Verification of Finite Field Arithmetic Circuits using Ideal Membership Testing

Overcoming the Complexity of Gröbner Bases for Efficient Verification of Finite Field and Integer Multipliers

Priyank Kalla

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

Nov 22, 2023 - onwards

Verification using Nullstellensatz over $\mathbb{F}_q$

We have two approaches to verify circuits using the Nullstellensatz

- Verify circuits using the miter model
- Construct a miter, and apply the **Weak Nullstellensatz**
- Construct ideal $J_m = \langle f_{spec}, f_1, \ldots, f_s, f_m \rangle$
- Polynomials $f_1, \ldots, f_s$ are the polynomials from the circuit
- J_0 = ideal of all vanishing polynomials
- Circuit $\equiv$ Spec if and only if $GB(J_m + J_0) = \{1\}$.

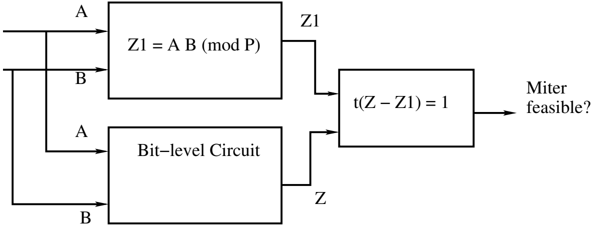


Figure: The equivalence checking setup: miter.

Second Approach to Verification: Using Ideal Membership

- The **second** approach is based on Ideal Membership
- It uses the concepts of Radical Ideals and the Strong Nullstellensatz
- Today, I will teach the procedure, with an intuitive explanation of the formulation. I will give the proof of correctness of the procedure next week onwards, as it requires the understanding of radical ideals.

Verification Formulation using Ideal Membership

- We are given a finite field $\mathbb{F}_{2^k}$, for a given k.
- Given a spec polynomial f_{spec} and an implementation circuit C
- Derive ideal $J = \langle f_1, \ldots, f_s \rangle$, where $\{f_1, \ldots, f_s\}$ are polynomials from the given circuit C
- It is NOT sufficient to check if $f_{spec} \in J$.
- It is **necessary and sufficient** to check if $f_{spec} \in J + J_0$, where $J_0 =$ ideal of all vanishing polynomials.

Verification Setup and Formulation

Verification Formulation for Finite Field Multipliers

- Setup the verification formulation over the polynomial ring $R = \mathbb{F}_q[x_1, \ldots, x_n], q = 2^k$.
- Let $f_{spec} : Z - A \cdot B$ be the specification polynomial.
- From the given circuit implementation C, derive the polynomials from the gates of the circuit $\{f_1, \ldots, f_s\}$.
- Let ideal $J = \langle F \rangle = \langle f_1, \ldots, f_s \rangle \subseteq R$.
- For each variable, create the ideal of vanishing polynomials $J_0 = \langle Z^q - Z, A^q - A, B^q - B, x_1^2 - x_1, \ldots, x_n^2 - x_n \rangle >$.
- Then, the circuit C implements $f_{spec} \iff$ (if and only if) $f_{spec} \in (J + J_0) \iff f_{spec} \xrightarrow{GB(J+J_0)}_+ 0$.

Compute $G = GB(J + J_0) = \{g_1, \ldots, g_t\}$, and divide f_{spec} by $G = \{g_1, \ldots, g_t\}$, and see if the remainder is 0.

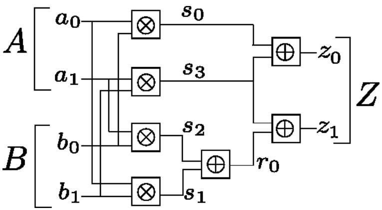
Verification Formulation: The Mathematical Problem

- 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 $f_1 = f_2 = \cdots = f_s = 0$?
- Can the spec f be written as a combination of $f_1, \ldots, f_s$ and J_0?
- Is $f \xrightarrow{GB(J+J_0)}_+ 0$?

Example Formulation



Gates as polynomials
$\mathbb{F}_2 \subset \mathbb{F}_{2^k}$:
Ideal J :

$$z_0 = s_0 + s_3; \;\mapsto\; f_1 : z_0 + s_0 + s_3$$
$$s_0 = a_0 \cdot b_0; \;\mapsto\; f_2 : s_0 + a_0 \cdot b_0$$
$$\vdots$$
$$A + a_0 + a_1\alpha;\; B + b_0 + b_1\alpha;\; Z + z_0 + z_1\alpha$$

Ideal J_0:
$z_0^2 - z_0, s_0^2 - s_0,$
$\vdots$

$A^{2^k} - A, B^{2^k} - B,$
$Z^{2^k} - Z$

Complexity of Gröbner Basis

- Complexity of Gröbner basis
  - Degree of polynomials in G is bounded by $2(\frac{1}{2}d^2 + d)^{2^{n-1}}$ [1]
  - Doubly-exponential in n and polynomial in the degree d
- This is the complexity of the GB problem, not of Buchberger's algorithm – that's still a mystery
- For $J \subset \mathbb{F}_q[x_1, \dots, x_n]$, Complexity $GB(J + J_0) : q^{O(n)}$ (Single exponential)
- Improving Buchberger's algorithm:
  - Improve term ordering (heuristics)
  - Get to all $S(f, g) \xrightarrow{G}_+ 0$ quickly; i.e. arrive at a GB quickly (hard to predict)
  - Improve the implementation of polynomial division; ideas proposed by *Faugére* in the F_4 algorithm

- 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:

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

Effect of Term Orderings on Buchberger's Algorithm

The Product Criteria

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

LEX: $x_0 > x_1 > x_2 > x_3$

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

LEX: $x_3 > x_2 > x_1 > x_0$

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

"Obviate" Buchberger's algorithm... really?

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

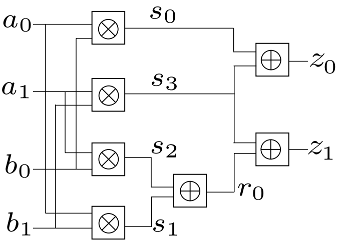
Product Criteria and Gröbner Bases

Recall Buchberger's theorem

The set $G = \{g_1, \ldots, g_t\}$ is a Gröbner basis **iff** for all pairs $(f, g) \in G, \; S(f, g) \xrightarrow{G}_+ 0$

- If we can make leading monomials of all pairs $lm(f), lm(g)$ relatively prime, then all $Spoly(f, g)$ reduce to 0
- This would imply that the polynomials already constitute a Gröbner basis
- No need to compute a GB, may be able to circumvent the GB complexity issues
- Can a term order be derived that makes leading monomials of all polynomials relatively prime?
  - For an "acyclic" circuit, make the gate output variable x_i greater than all variables x_j that are inputs to the gate

For Circuits, such an order can be derived



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

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

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

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

- Perform a Reverse Topological Traversal of the circuit, order the variables according to their reverse topological levels
- LEX with $Z > \{A > B\} > \{z_0 > z_1\} > \{r_0 > s_0 > s_3\} > \{s_1 > s_2\} > \{a_0 > a_1 > b_0 > b_1\}$
- This makes every gate output a leading term, and $\{f_1, \ldots, f_{10}\}$ is a Gröbner basis

This term order also renders a Gröbner Basis of $J + J_0$

Using the 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 (these polynomials also have relatively prime leading terms)
- But we have to compute a Gröbner Basis of $J + J_0 = \langle f_1, f_2 \ldots, f_s, \quad x_1^q - x_1, \ldots, x_n^q - x_n \rangle$
- It turns out that $\{f_1, f_2 \ldots, f_s, \quad x_1^q - x_1, \ldots, x_n^q - x_n\}$ is a Gröbner basis!!
- From our circuit: $f_i = {\color{red}x_i} + tail(f_i) = {\color{red}x_i} + P$
- Vanishing polynomials ${\color{red}x_i^q} - x_i$ with same variable x_i
- Only pairs to consider: $S(f_i, \quad x_i^q - x_i)$ in Buchberger's Algorithm
- All other pairs will have relatively prime leading terms, which will reduce to 0 modulo G

This term order renders a Gröbner basis by construction

So, let us compute $S(f_i = x_i + P, \;\; x_i^q - x_i)$:

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

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

Since $P^q - P$ is a vanishing polynomial, $P^q - P \in J_0$ and $P^q - P \xrightarrow{J_0}_+ 0$

Conclusion: The set of polynomials
$F \cup F_0 = \{f_1, \dots, f_s, \;\; x_i^q - x_i, \dots, x_n^q - x_n\}$ is itself a Gröbner basis due to the reverse topological term order derived from the circuit!

Our Minimal Gröbner Basis

Conclusion:

- Our term order makes $G = \{f_1, \ldots, f_s, x_1^q - x_1, \ldots, x_n^q - x_n\}$ a Gröbner Basis
- This GB$(J + J_0)$ can be further simplified (made minimal)
  - Two types of polynomials: $f_i = x_i + P, \; g_i = x_i^q - x_i$
  - Primary inputs bits are never a leading term of any polynomial
  - Primary inputs are not the output of any gate
- For $x_i \notin$ primary inputs, $f_i = x_i + P$ divides $x_i^q - x_i$; remove $x_i^q - x_i$
- Keep $J_0 = \langle x_i^2 - x_i : x_i \in \text{primary input bits}\rangle$

Our term order makes $G = \{f_1, \ldots, f_s, \;\; x_{PI}^2 - x_{PI}\}$ a minimal Gröbner basis by construction!

Verify the circuit only by a reduction: $f \xrightarrow{G}_+ 0$?

Our Overall Approach

- Given the circuit, perform reverse topological traversal
- Derive the term order to represent the polynomials for every gate, call it the Reverse Topological Term Order (RTTO) $>$
- The set: $\{F, F_0\} = \{f_1, \ldots, f_s, \;\; x_i^2 - x_i : x_i \in X_{PI}\}$ is a minimal Gröbner Basis
- Obtain: $f \xrightarrow{F,F_0}_+ r$
- If $r = 0$, the circuit is verified 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

Move the complexity to that of Polynomial Division

Is this Magic? Or have I told you the full story?

- Reduce x^n modulo $\langle x + P \rangle$, how many cancellations?
  - Requires raising P to the n^{th} power
  - P is the $tail(f_i)$
  - Depending upon n, this can become complicated
- **Reduce** this **minimal** GB $G = \{F, F_0\}$, what does it look like?
  - $f_i = x_i + \text{tail}(f_i)$, where $\text{tail}(f_i) = P(x_j), x_i > x_j$
  - There exists $f_j = x_j + \text{tail}(f_j)$, where $f_j \mid P(x_j)$
  - All non-PI variables x_j can be canceled in this reduction
  - Reduction results in GB G with only primary input variables, potentially explosive

This approach should work for specification polynomials f with low degree terms

Experiments: Correctness Proof, Miter Mastrovito v/s Montgomery Multipliers

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

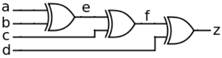
	Word size of the operands k-bits		
Solver	8	12	16
MiniSAT	22.55	*TO*	*TO*
CryptoMiniSAT	7.17	16082.40	*TO*
PrecoSAT	7.94	*TO*	*TO*
PicoSAT	14.85	*TO*	*TO*
Yices	10.48	*TO*	*TO*
Beaver	6.31	*TO*	*TO*
CVC	*TO*	*TO*	*TO*
Z3	85.46	*TO*	*TO*
Boolector	5.03	*TO*	*TO*
SimplifyingSTP	14.66	*TO*	*TO*
ABC	242.78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

Verify a specification polynomial f against a circuit C by performing the test $f \xrightarrow{J+J_0}_+ 0$?

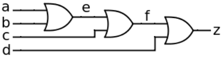
Table: Verify bug-free and buggy Mastrovito multipliers. SINGULAR computer algebra tool used for division.

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

Why does Compute-GB (SINGULAR) run out of memory?



(a) XOR logic



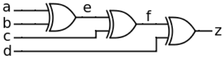
(b) OR logic

For XOR logic:

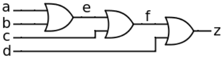
$$f_1 : z + f + d \quad f_2 : f + e + c \quad f_3 : e + b + a$$

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

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



(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 the same ideal membership approach to verify integer multipliers
- Consider a 2-bit (integer multiplier) circuit. Prove that it is an integer multiplier! Or prove that it is buggy.

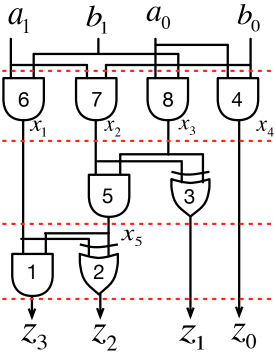


Figure: Integer multiplier circuit

Integer Arithmetic Verification Model

- What is the spec?
- Output word: $z_0 + 2z_1 + 4z_2 + 8z_3$, z_i are bits $\{0, 1\}$
- Input words: $a_0 + 2a_1$, $b_0 + 2b_1$.
- $f_{spec} : z_0 + 2z_1 + 4z_2 + 8z_3 = (a_0 + 2a_1)(b_0 + 2b_1)$
- In polynomial form: $f_{spec} : z_0 + 2z_1 + 4z_2 + 8z_3 - (a_0 + 2a_1)(b_0 + 2b_1)$
- Note f_{spec} has cofficients in $\mathbb{Z}$, but $\mathbb{Z}$ is NOT a field, so we cannot apply Nullstellensatz!
- Trick: Model the problem over $\mathbb{Q}[x_1, \ldots, x_n]$, BUT, use the same RTTO order (important)
- How to model Boolean logic gates over $\mathbb{Q}$?

Model Logic Gates over $\mathbb{Q}$

$$
\begin{aligned}
z = \neg a \mapsto &\quad z = 1 - a \quad \mapsto z - 1 + a \\
z = a \wedge b \mapsto &\quad z = a \cdot b \quad \mapsto z - a \cdot b \\
z = a \vee b \mapsto &\quad z = a + b - a \cdot b \quad \mapsto z - a - b + ab \\
z = a \oplus b \mapsto &\quad z = a + b - 2 \cdot a \cdot b \quad \mapsto z - a - b + 2ab
\end{aligned}
$$

- This requires that every variable take binary values: $a^2 = a$ or $J_0 = \langle a^2 - a, b^2 - b, \ldots, z^2 - z \rangle$
- Construct ideal J from logic gates, add bit-level vanishing polynomials J_0
- What is the leading term of polynomials in J under RTTO?
- Gate output is the leading term, and leading coefficient $= 1$
- Divide by $lc(f) = 1$, division will NEVER produce fractions!

Verification f_{spec} (mod $J + J0$) under RTTO

$$
\begin{aligned}
Z =& 8z_3 + 4z_2 + 2z_1 + z_0 \\
=& \underline{8x_1x_2x_3} + (4x_1 + \underbrace{4x_2x_3} - \underline{8x_1x_2x_3}) \\
& + (2x_2 + 2x_3 - \underbrace{4x_2x_3}) + x_4 \\
=& 4x_1 + 2x_2 + 2x_3 + x_4
\end{aligned}
$$

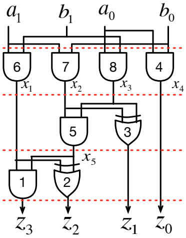


Figure: Integer multiplier circuit

- Ring $R = 0, (z_3, z_2, z_1, z_0, x_5, x_1, x_2, x_3, x_4, a_0, a_1, b_0, b_1), lp$;
- Circuit is an integer multiplier if $f_{spec} \xrightarrow{J+J_0}_+ 0$.

In Conclusion

The Key to Success in Design Automation

- Build algorithms and techniques on solid theoretical foundations
- Use all of the mathematical tools at your disposal
- Make sure to exploit circuit structure
- Develop domain-specific implementations
- That's what SAT, BDDs, AIGs do too!

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