Combinational Circuit Verification using Strong Nullstellensatz
Overcoming the Complexity of Gröbner Bases for Efficient Verification, and Verification of 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, 2021 - onwards
What we have learnt so far...

Theorem (Weak Nullstellensatz)

Let $\overline{F}$ be an algebraically closed field. Given ideal

$J \subset \overline{F}[x_1, \ldots, x_n], V_{\overline{F}}(J) = \emptyset \iff J = \overline{F}[x_1, \ldots, x_n] \iff 1 \in J \iff \text{reducedGB}(J) = \{1\}$.

Theorem (The Strong Nullstellensatz)

Over an algebraically closed field $I(V(J)) = \sqrt{J}$
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 \( J_m = \langle f_{\text{spec}}, f_1, \ldots, f_s, f_m \rangle \)
- 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.
The second approach is based on Ideal Membership in $I(V(J))$

- 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.
- Why? Because $f_{spec}$ may vanish on the variety $V_{\mathbb{F}_q}(J)$
- So, $f \in I(V_{\mathbb{F}_q}(J))$. Remember $I(V(J))$?

$I(V)$

Let $J = \langle f_1, \ldots, f_s \rangle \subset \mathbb{F}[x_1, \ldots, x_n]$. Then:

$I(V(J)) = \{ f \in \mathbb{F}[x_1, \ldots, x_n] : f(a) = 0 \ \forall a \in V(J) \}$

- $I(V(J))$ is the set of all polynomials that vanish on $V(J)$
- If $f$ vanishes on $V(J)$, then $f \in I(V(J))$
Nullstellensatz over $\mathbb{F}_q$

**Theorem (Weak Nullstellensatz over $\mathbb{F}_{2^k}$)**

Let ideal $J = \langle f_1, \ldots, f_s \rangle \subset \mathbb{F}_{2^k}[x_1, \ldots, x_n]$ be an ideal. Let $J_0 = \langle x_1^{2^k} - x_1, \ldots, x_n^{2^k} - x_n \rangle$ be the ideal of all vanishing polynomials. Then:

$$V_{\mathbb{F}_{2^k}}(J) = \emptyset \iff V_{\mathbb{F}_{2^k}}(J + J_0) = \emptyset \iff \text{reducedGB}(J + J_0) = \{1\}$$

**Theorem ($J + J_0$ is radical)**

Over Galois fields $\sqrt{J + J_0} = J + J_0$, i.e. $J + J_0$ is a radical ideal.

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

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

Gates as polynomials

\[ F_2 \subset F_{2^k} \]

Ideal \( J \):

\[
\begin{align*}
z_0 &= s_0 + s_3; \quad \Rightarrow \quad f_1 : z_0 + s_0 + s_3 \\
s_0 &= a_0 \cdot b_0; \quad \Rightarrow \quad f_2 : s_0 + a_0 \cdot b_0 \\
&\cdots
\end{align*}
\]

\[
A + a_0 + a_1 \alpha; \quad B + b_0 + b_1 \alpha; \quad Z + z_0 + z_1 \alpha
\]

Ideal \( J_0 \):

\[
\begin{align*}
z_0^2 - z_0, s_0^2 - s_0, \\
&\cdots
\end{align*}
\]

\[
A^{2^k} - A, B^{2^k} - B, \\
Z^{2^k} - Z
\]
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 $J \subset \mathbb{F}_q[x_1, \ldots, x_n]$, Complexity $GB(J + J_0) : q^{O(n)}$ (Single exponential)

- Improving Buchberger’s algorithm:
  - Improve term ordering (heuristics)
  - Get to all $S(f, g) \rightarrow^+_G 0$ quickly; i.e. arrive at a GB quickly (hard to predict)
  - Improve the implementation of polynomial division; ideas proposed by Faugère in the $F_4$ algorithm
Complexity of Gröbner Basis and Term Orderings

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

Recall, S-polynomial depends on term ordering:

$$S(f, g) = \frac{L}{\text{lt}(f)} \cdot f - \frac{L}{\text{lt}(g)} \cdot g; \quad L = \text{LCM}(\text{lm}(f), \text{lm}(g))$$
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.
Recall Buchberger’s theorem

The set \( G = \{g_1, \ldots, g_t\} \) is a Gröbner basis if and only if for all pairs \((f, g) \in G\), \( S(f, g) \xrightarrow{G} + 0 \)

- If we can make leading monomials of all pairs \( \text{lm}(f), \text{lm}(g) \) relatively prime, then all \( \text{Spoly}(f, g) \) 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 $J + J_0 = \langle f_1, f_2, \ldots, f_s, x_1^q - x_1, \ldots, x_n^q - x_n \rangle$
- It turns out 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: $f_i = x_i + \text{tail}(f_i) = x_i + P$
- Vanishing polynomials $x_i^q - x_i$ with same variable $x_i$
- Only pairs to consider: $S(f_i, x_i^q - x_i)$ 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 $S(f_i = x_i + P, \ x_i^q - x_i)$:

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

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

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 $F \cup F_0 = \{ f_1, \ldots, f_s, \ x_i^q - x_i, \ldots, x_n^q - x_n \}$ is itself a Gröbner basis due to the reverse topological term order derived from the circuit!
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 $x_i \not\in$ 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 $G = \{f_1, \ldots, f_s, x_P^2 - x_{PI}\}$ 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 $\text{tail}(f_i)$
  - 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 $f_j | P(x_j)$
  - 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.**

<table>
<thead>
<tr>
<th>Solver</th>
<th>Word size of the operands $k$-bits</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>8</td>
</tr>
<tr>
<td>MiniSAT</td>
<td>22.55</td>
</tr>
<tr>
<td>CryptoMiniSAT</td>
<td>7.17</td>
</tr>
<tr>
<td>PrecoSAT</td>
<td>7.94</td>
</tr>
<tr>
<td>PicoSAT</td>
<td>14.85</td>
</tr>
<tr>
<td>Yices</td>
<td>10.48</td>
</tr>
<tr>
<td>Beaver</td>
<td>6.31</td>
</tr>
<tr>
<td>CVC</td>
<td><strong>TO</strong></td>
</tr>
<tr>
<td>Z3</td>
<td>85.46</td>
</tr>
<tr>
<td>Boolector</td>
<td>5.03</td>
</tr>
<tr>
<td>SimplifyingSTP</td>
<td>14.66</td>
</tr>
<tr>
<td>ABC</td>
<td>242.78</td>
</tr>
<tr>
<td>BDD</td>
<td>0.10</td>
</tr>
</tbody>
</table>
Experimental Results: Correctness Proof

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.

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

Why does Compute-GB (*SINGULAR*) run out of memory?
Limitations of RTTO-based GB-reduction

For XOR logic:

\[ f_1 : z + f + d \]
\[ f_2 : f + e + c \]
\[ f_3 : e + b + a \]

The reduction procedure \( z \stackrel{f_1,f_2,f_3}{\longrightarrow} r \) will be computed as follows:

1. \( z \stackrel{z+f+d}{\longrightarrow} f + d \)
2. \( (f + d) \stackrel{f+e+c}{\longrightarrow} e + d + c \)
3. \( (e + d + c) \stackrel{e+b+a}{\longrightarrow} d + c + b + a \)
Limitations of GB-Reduction: OR-gates explode

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:

1. \( z \xrightarrow{z+fd+f+d} fd + f + d \)
2. \( (fd + f + d) \xrightarrow{f+ec+e+c} f + edc + ed + dc + d; \)
   \( (f + edc + ed + dc + d) \xrightarrow{f+ec+e+c} edc + ed + ec + e + dc + d + c \)
3. \( (edc + ed + ec + e + dc + d + c) \xrightarrow{e+ba+b+a} + \\
   dcba + dcb + dca + dba + dc + db + da + d + cba + cb + ca + c + ba + b + a \)
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
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 coefficients 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 $\mathbb{Q}$

\[
\begin{align*}
z &= \neg a \quad \rightarrow \quad z &= 1 - a \quad \rightarrow \quad z - 1 + a \\
z &= a \wedge b \quad \rightarrow \quad z &= a \cdot b \quad \rightarrow \quad z - a \cdot b \\
z &= a \vee b \quad \rightarrow \quad z &= a + b - a \cdot b \quad \rightarrow \quad z - a - b + ab \\
z &= a \oplus b \quad \rightarrow \quad z &= a + b - 2 \cdot a \cdot b \quad \rightarrow \quad z - a - b + 2ab
\end{align*}
\]

- 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 $f_{\text{spec}} \pmod{J + J_0}$ under RTTO

$$Z = 8z_3 + 4z_2 + 2z_1 + z_0$$

$$= 8x_1x_2x_3 + (4x_1 + 4x_2x_3 - 8x_1x_2x_3)$$

$$+ (2x_2 + 2x_3 - 4x_2x_3) + x_4$$

$$= 4x_1 + 2x_2 + 2x_3 + x_4$$

**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 $f_{\text{spec}} \xrightarrow{J+J_0} + 0$. 
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!