Formal Verification of Sequential Galois Field Arithmetic Circuits using Algebraic Geometry

Xiaojun Sun[1], Priyank Kalla[1], Tim Pruss[1], Florian Enescu[2]

[1]Electrical and Computer Engineering, University of Utah, Salt Lake City, USA
[2]Mathematics & Statistics, Georgia State University, Atlanta, USA

{**xiaojuns**, kalla, tpruss}@ece.utah.edu, fenescu@gsu.edu

Session 12.2 – Long Presentation

Motivation and Basic Idea

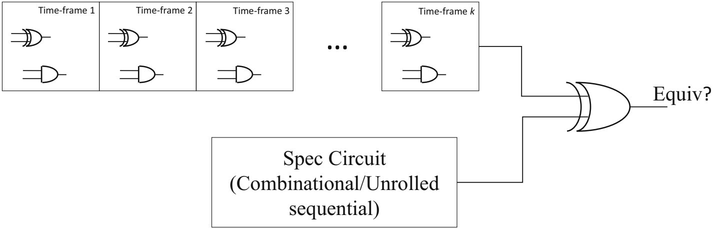
- Target problems
  - Given:
    - A Galois field (GF) and normal basis representation
    - A word level specification polynomial
    - A sequential implementation of polynomial computation
  - Aim: perform property checking on the sequential implementation
- Focus
  - Analyze and abstract the function of sequential GF arithmetic circuits
  - Implicit word-level finite state machine (FSM) traversal
- Motivation
  - Data flow = word level info
  - Conventional techniques are bit-level
  - Gröbner basis theory can assist bit-to-word conversion
  - Word level $\rightarrow$ implicit $\rightarrow$ efficient!

Outline

- Background application
- Preliminaries
    - Field, polynomial ideal
    - Abstraction using Gröbner basis
    - Normal basis representation
- Methodology
    - Basic algorithm
    - Improving our approach
- Experiment results
- Conclusion

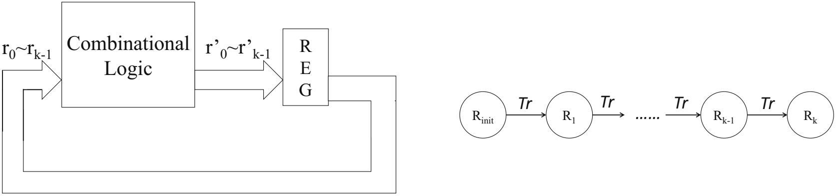
- Cryptography: polynomial computation over $\mathbb{F}_{2^k}$
  - Algebraic nature (GF) of the computation (polynomial)
  - Datapath size k: very large
- Verification of sequential circuits is needed
  - Verify sequential GF circuits designed using normal basis
  - Complicated circuits need to be verified
  - Sequential circuits bounded by k clock cycles
  - Example: [Reyhani-Masoleh and Hasan, *Sequential normal basis multipliers*, Trans on Computer, 2005]

Our approach vs Conventional approach



- Conventional: explicitly unroll k time-frames
  - Bit-blasting!
- New: Implicitly unroll the finite state machine (FSM)
  - Update the spec poly for k times ($R = \mathcal{F}(A, B)$) when unrolling
  - $R_1 = \mathcal{F}(A_{init}, B_{init})$, $R_2 = \mathcal{F}(A_1, B_1) = \mathcal{F}^2(A_{init}, B_{init})$, $\cdots$, $R_k = \mathcal{F}^k(A_{init}, B_{init}) = A_{init} \cdot B_{init}$

Illustration of implicit unrolling



(a) (b)

- Model: restricted Moore finite state machine
  - Some sequential arithmetic circuits will give results after running for k clock cycles
  - The initial operands are preloaded in register files
- State transitions on this model:

$$R_k = Tr(R_{k-1}) = Tr(Tr(\cdots Tr(R_{init}) \cdots)) = Tr^k(R_{init})$$

Galois Field Overview

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

- Commutative Ring with unity, associate, distributive laws
- Closure property: $+, -, \times$, inverse $(\div)$

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
- $P(\alpha) = 0$, $\alpha =$Primitive element

- Normal basis representation: $A(a_0, \ldots, a_{k-1}) = \sum_{i=0}^{k-1} a_{n(i)}\beta^{2^i}$
- Normal element: $\beta = \alpha^t$
- Squaring of elements represented in normal bases can be implemented simply by a cyclic right-shift operation.

Example

For $a, b \in \mathbb{F}_{2^k}, (a+b)^2 = a^2 + b^2$. Applying this rule for element squaring:

$$\begin{aligned} B =& b_0\beta + b_1\beta^2 + b_2\beta^4 + \cdots + b_{k-1}\beta^{2^{k-1}} \\ B^2 =& b_0^2\beta^2 + b_1^2\beta^4 + b_2^2\beta^8 + \cdots + b_{k-1}^2\beta^{2^k} \\ =& b_{k-1}\beta + b_0\beta^2 + b_1\beta^4 + \cdots + b_{k-2}\beta^{2^{k-1}} \end{aligned}$$

as $\beta^{2^k} = \beta$ by applying Fermat's little theorem to $\mathbb{F}_{2^k}$, and $b_i^2 = b_i$. It is an 1-bit cyclic right-shift $\rightarrow$ implemented efficiently with sequential circuit

SPEC: $R = A_{init} \cdot B_{init} \pmod{P(\alpha)}$ after k clock cycles

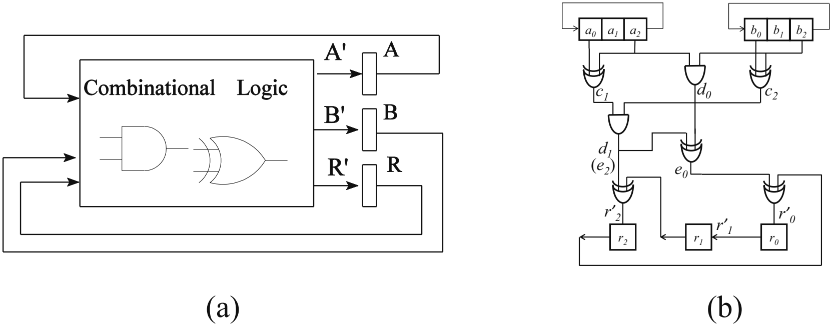


Figure : A 3-bit RH-SMPO and its Moore FSM model

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 $f_1, f_2, \dots, f_s \in \mathbb{F}_q[x_1, \dots, x_d]$. Let

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

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

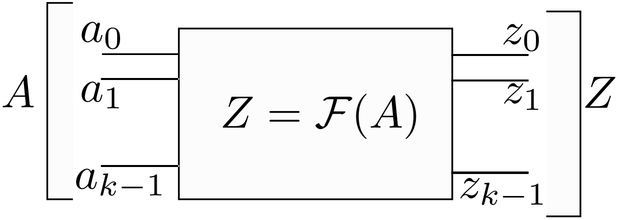
- Different generators can generate the same ideal
- $\langle f_1, \cdots, f_s \rangle = \cdots = \langle g_1, \cdots, g_t \rangle$
- Some generators are a "better" representation of the ideal
- A (reduced)**Gröbner basis** is a "canonical" representation of an ideal

Given $F = \{f_1, f_2, \cdots, f_s\}$, Compute a GB (using Buchberger's algorithm) $G = \{g_1, g_2, \cdots, g_t\}$, such that $I = \langle F \rangle = \langle G \rangle$

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

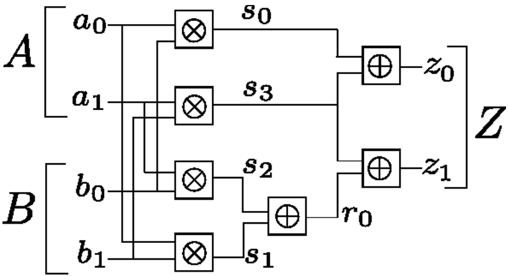
- GB computation requires a term order
- Let ideal $I = \langle f_1, f_2, f_3 \rangle$ where
  - $f_1 = x^2 + y + z - 1$
  - $f_2 = x + y^2 + z - 1$
  - $f_3 = x + y + z^2 - 1$
- The Gröbner basis of I with lex order $(x > y > z)$ is
  - $g_1 = x + y + z^2 - 1$
  - $g_2 = y^2 - y - z^2 + z$
  - $g_3 = 2yz^2 + z^4 - z^2$
  - $g_4 = z^6 - 4z^4 + 4z^3 - z^2$
- g_2, g_3 and g_4: only contain variables y and z
  - Eliminates variable $x \Leftrightarrow \exists_x$ in Boolean formula!
- g_4: only contains the variable $z \rightarrow$ eliminates x and y

Abstraction Term Ordering[Pruss et al, *Abstraction using GB*, DAC'14]



- Impose a lex term order $>$ on the polynomial ring such that circuit-variables including $a_0, \dots, a_{k-1}, z_0, \dots, z_{k-1} > Z > A$.
- This elimination term order $>$: **Abstraction Term Order** (ATO).
- Compute a Gröbner basis G of ideal $(J + J_0)$ using $>$
  - G will contain a polynomial of the form $Z + \mathcal{F}(A)$ $(Z = \mathcal{F}(A))$
  - $Z = \mathcal{F}(A)$ is a unique, canonical, polynomial representation of C over $\mathbb{F}_q$

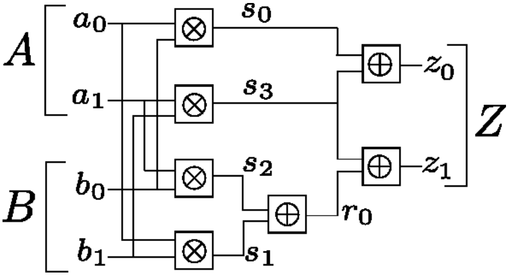
Abstraction Term Order Example



$(z_0 > z_1 > r_0 > s_0 > s_3 > s_1 > s_2 > a_0 > a_1 > b_0 > b_1 > Z > A > B)$

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

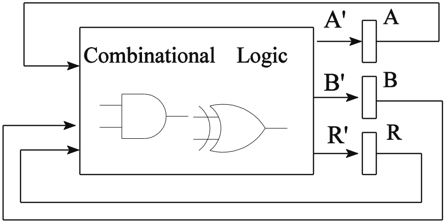
$$J = \langle f_1, \ldots, f_{10} \rangle \quad + \quad J_0 = \langle \textbf{vanishing poly } x^q - x \rangle$$



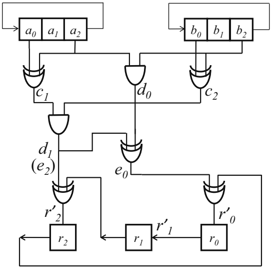
$(z_0 > z_1 > r_0 > s_0 > s_3 > s_1 > s_2 > a_0 > a_1 > b_0 > b_1 > Z > A > B)$
Compute the Gröbner basis, G, of $\{J + J_0\}$ with respect to abstraction term ordering $>$. $G = \{g_1, \ldots, g_{14}\}$

$$g_1 : B^4 + B; \quad g_2 : b_0 + b_1\alpha + B; \quad g_3 : a_0 + a_1\alpha + A; \quad g_4 : A^4 + A;$$
$$g_5 : s_0 + s_1\alpha + s_2(\alpha + 1) + Z; \quad g_6 : r_0 + s_1 + s_2; \quad g_7 : z_1 + r_0 + s_3$$
$$g_7 : z_0 + z_1\alpha + Z; \quad \mathbf{g_9 : Z + A * B}; \quad g_{10} : b_1 + B^2 + B; \quad g_{11} : a_1 + A^2 + A$$
$$g_{12} : s_3 + a_1 b_1; \quad g_{13} : s_2 + a_1 b_1\alpha + a_1 B; \quad g_{14} : s_1 + a_1 b_1\alpha + b_1 A$$

SPEC: $R = A_{init} \cdot B_{init} \pmod{P(\alpha)}$ after k clock cycles



(a)



(b)

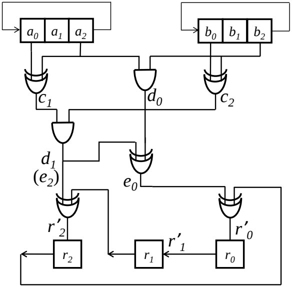
Figure : A 3-bit RH-SMPO and its Moore FSM model

ALGORITHM 1: Abstraction via implicit unrolling for Sequential GF circuit verification

Input: Circuit polynomial ideal J, vanishing ideal J_0, initial state ideal $R(=0), \mathcal{G}(A_{init}), \mathcal{H}(B_{init})$

$from_0(R, A, B) = \langle R, \mathcal{G}(A_{init}), \mathcal{H}(B_{init}) \rangle$;
$i = 0$;
repeat
    $i \leftarrow i + 1$;
    $G \leftarrow$ GB($\langle J + J_0 + from_{i-1}(R, A, B) \rangle$) with ATO;
    $to_i(R', A', B') \leftarrow G \cap \mathbb{F}_{2^k}[R', A', B', R, A, B]$;
    $from_i \leftarrow to_i(\{R, A, B\} \setminus \{R', A', B'\})$;
until $i == k$;
return $from_k(R_{final})$

Experiment on 3-bit RH-SMPO



- The elimination ideal (first iteration):

$$
\begin{aligned}
J = & d_0 + b_2 \cdot a_2, c_1 + a_0 + a_2, c_2 + b_0 + b_2, d_1 + c_1 \cdot c_2, \\
& e_0 + d_0 + d_1, e_2 + d_1, r'_0 + r_2 + e_0, r'_1 + r_0, r'_2 + r_1 + e_2, \\
& A + a_0\beta + a_1\beta^2 + a_2\beta^4, B + b_0\beta + b_1\beta^2 + b_2\beta^4, \\
& R + r_0\beta + r_1\beta^2 + r_2\beta^4, R' + r'_0\beta + r'_1\beta^2 + r'_2\beta^4;
\end{aligned}
$$

Experiment on 3-bit RH-SMPO(2)

- $J_0 = \langle x_i^2 - x_i, X^q - X \rangle$
- $from_0 = \{R, A_{init} + a_0\beta + a_1\beta^2 + a_2\beta^4, B_{init} + b_0\beta + b_1\beta^2 + b_2\beta^4\}$

ALGORITHM 2: Abstraction via implicit unrolling for Sequential GF circuit verification

Input: Circuit polynomial ideal J, vanishing ideal J_0, initial state ideal $R(=0), \mathcal{G}(A_{init}), \mathcal{H}(B_{init})$

$from_0(R, A, B) = \langle R, \mathcal{G}(A_{init}), \mathcal{H}(B_{init})\rangle$;

$i = 0$;

repeat

  $i \leftarrow i + 1$;

  $G \leftarrow$GB($\langle J + J_0 + from_{i-1}(R, A, B)\rangle$) with ATO;

  $to_i(R', A', B') \leftarrow G \cap \mathbb{F}_{2^k}[R', A', B', R, A, B]$;

  $from_i \leftarrow to_i(\{R, A, B\} \setminus \{R', A', B'\})$;

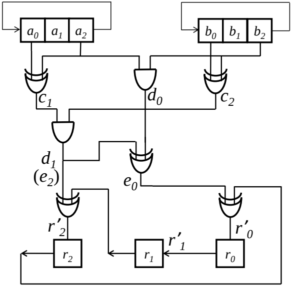
until $i == k$;

return $from_k(R_{final})$

- $J_0 = \langle x_i^2 - x_i, X^q - X \rangle$
- $from_0 = \{R, A_{init} + a_0\beta + a_1\beta^2 + a_2\beta^4, B_{init} + b_0\beta + b_1\beta^2 + b_2\beta^4\}$ $(\beta = \alpha^3)$
- $to_1 : R' + (\alpha^2)A_{init}^4 B_{init}^4 + (\alpha^2 + \alpha)A_{init}^4 B_{init}^2 + (\alpha^2 + \alpha)A_{init}^4 B_{init} + (\alpha^2 + \alpha)A_{init}^2 B_{init}^4 + (\alpha^2 + \alpha + 1)A_{init}^2 B_{init}^2 + (\alpha^2)A_{init}^2 B_{init} + (\alpha^2 + \alpha)A_{init} B_{init}^4 + (\alpha^2)A_{init} B_{init}^2$
- $from_1 = \{R' + (\alpha^2)A_{init}^4 B_{init}^4 + (\alpha^2 + \alpha)A_{init}^4 B_{init}^2 + (\alpha^2 + \alpha)A_{init}^4 B_{init} + (\alpha^2 + \alpha)A_{init}^2 B_{init}^4 + (\alpha^2 + \alpha + 1)A_{init}^2 B_{init}^2 + (\alpha^2)A_{init}^2 B_{init} + (\alpha^2 + \alpha)A_{init} B_{init}^4 + (\alpha^2)A_{init} B_{init}^2, A_{init} + a_2\alpha^3 + a_0\alpha^6 + a_1\alpha^{12}, B_{init} + b_2\alpha^3 + b_0\alpha^6 + b_1\alpha^{12}\}$
- $\cdots$
- After 3 iterations: $to_3 = \{R' + A_{init} B_{init},\ A_{init} + a_0'\alpha^3 + a_1'\alpha^6 + a_2'\alpha^{12},\ B_{init} + b_0'\alpha^3 + b_1'\alpha^6 + b_2'\alpha^{12}\}$

Refined Abstraction Term Ordering (RATO)[Pruss et al, *Abstraction using GB*, DAC'14]

- Computing GB: high computational complexity
- Buchberger's algorithm simplified w/ special term order
- reverse-topological term order: only input variables left in remainder



Example

Elimination ideal under RATO:

$$\begin{aligned} J = & d_0 + b_2 \cdot a_2, c_1 + a_0 + a_2, c_2 + b_0 + b_2, d_1 + c_1 \cdot c_2, \\ & e_0 + d_0 + d_1, e_2 + d_1, r'_0 + r_2 + e_0, r'_1 + r_0, r'_2 + r_1 + e_2, \\ & a_0\alpha^3 + a_1\alpha^6 + a_2\alpha^{12} + A, b_0\alpha^3 + b_1\alpha^6 + b_2\alpha^{12} + B, \\ & r_0\alpha^3 + r_1\alpha^6 + r_2\alpha^{12} + R, r'_0\alpha^3 + r'_1\alpha^6 + r'_2\alpha^{12} + R'; \end{aligned}$$

Example

$$r_0'\alpha^3 + r_1'\alpha^6 + r_2'\alpha^{12} + R' \xrightarrow{J}_+$$

$$(\alpha+1)r_1 + r_2 + \alpha b_1 a_1 + (\alpha^2+\alpha)b_1 a_2 + \alpha^2 b_1 A + (\alpha^2+\alpha)b_2 a_1 + \alpha b_2 a_2 + (\alpha^2+\alpha+1)b_2 A + \alpha^2 a_1 B + (\alpha^2+\alpha+1)a_2 B + R' + (\alpha+1)R + (\alpha+1)AB$$

- In [Pruss et al, *Abstraction using GB*, DAC'14], the authors did not address the problem when there are bit-level input variables in the remainder.
- Improve abstraction using RATO:

$$A = a_0\beta + a_1\beta^2 + \cdots + a_{k-1}\beta^{2^{k-1}}$$

$$\implies a_0 = \mathcal{F}_0(A), a_1 = \mathcal{F}_1(A), \ldots$$

Improve abstraction using RATO

- Given word definition $A = a_0\beta + a_1\beta^2 + \cdots + a_{k-1}\beta^{2^{k-1}}$, by squaring:

$$\begin{array}{rcl} A & = & a_0\beta + a_1\beta^2 + \cdots + a_{k-1}\beta^{2^{k-1}} \\ A^2 & = & a_0\beta^2 + a_1\beta^4 + \cdots + a_{k-1}\beta^{2\cdot 2^{k-1}} \\ \vdots & \vdots & \vdots \\ A^{2^{k-1}} & = & a_0\beta^{2^{k-1}} + a_1\beta^{2^{k-1}\cdot 2} + \cdots + a_{k-1}\beta^{2^{2(k-1)}} \end{array}$$

- Write in matrix form:

$$\begin{pmatrix} \beta & \beta^2 & \cdots & \beta^{2^{k-1}} \\ \beta^2 & \beta^4 & \cdots & \beta^{2^{k-1}\cdot 2} \\ \vdots & \vdots & \ddots & \vdots \\ \beta^{2^{k-1}} & \beta^{2^{k-1}\cdot 2} & \cdots & \beta^{2^{2(k-1)}} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{k-1} \end{pmatrix} = \begin{pmatrix} A \\ A^2 \\ \vdots \\ A^{2^{k-1}} \end{pmatrix}$$

Improve abstraction using RATO (2)

Example

Solve system of equations using Gaussian elimination:

$$\begin{pmatrix} \alpha^3 & \alpha^6 & \alpha^{12} \\ \alpha^6 & \alpha^{12} & \alpha^{24} \\ \alpha^{12} & \alpha^{24} & \alpha^{48} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} A \\ A^2 \\ A^4 \end{pmatrix}$$

Results are:

$$\begin{cases} a_0 = (\alpha^2 + \alpha + 1)A^4 + (\alpha^2 + 1)A^2 + (\alpha + 1)A \\ a_1 = (\alpha + 1)A^4 + (\alpha^2 + \alpha + 1)A^2 + (\alpha^2 + 1)A \\ a_2 = (\alpha^2 + 1)A^4 + (\alpha + 1)A^2 + (\alpha^2 + \alpha + 1)A \end{cases}$$

Similarly replace $b_0, b_1, b_2, r_0, r_1, r_2$ with B, R

Final approach

ALGORITHM 3: Abstraction via implicit unrolling for Sequential GF circuit verification

Input: Circuit polynomial ideal J, vanishing ideal J_0, initial state ideal $R(=0), \mathcal{G}(A_{init}), \mathcal{H}(B_{init})$

$from_0(R, A, B) = \langle R, \mathcal{G}(A_{init}), \mathcal{H}(B_{init}) \rangle$;
$i = 0$;
repeat
    $i \leftarrow i + 1$;
    $f_o \xrightarrow{J+J_0+from_{i-1}(R,A,B)}_+ f_r$ under RATO ;
    $to_i(R', A', B') \leftarrow f_r(\{R', A', B'\} \setminus \{r_0, \ldots, r_{k-1}, a_0, \ldots, a_{k-1}, b_0, \ldots, b_{k-1}\}$;
    $from_i \leftarrow to_i(\{R, A, B\} \setminus \{R', A', B'\})$;
until $i == k$;
return $from_k(R_{final})$

Table : Run-time (seconds) for verification of bug-free and buggy RH-SMPO using our approach

Operand size k	33	51	65	81	89	99
#variables	4785	11424	18265	28512	34354	42372
#polynomials	3630	8721	13910	21789	26255	32373
#terms	13629	32793	52845	82539	99591	122958
Runtime(bug-free)	112.6	1129	5243	20724	36096	67021
Runtime(buggy)	112.7	1129	5256	20684	36120	66929

- Experimented performed using SINGULAR
- Note: Using conventional methods we cannot verify any multipliers with 23+ bits datapath

Conclusion

- Succeed to verify large GF arithmetic circuits based on k-cycle unrolling
- Provide a way to simplify GB computation
- Has the potential to be applied to the verification of other sequential circuits

Singular code available: ece.utah.edu/∼xiaojuns/codes.html