Word-Level Multi-Fix Rectifiability of Finite Field Arithmetic Circuits

isQED® Symposium 2021

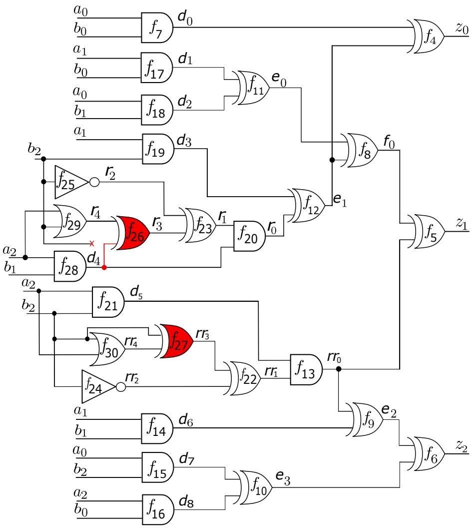
Vikas Rao[1], Irina Ilioaea[2], Haden Ondricek[1], Priyank Kalla[1], and Florian Enescu[3]

[1]Electrical & Computer Engineering, University of Utah
[2]Department of Mathematics, Louisiana State University Shreveport
[3]Mathematics & Statistics, Georgia State University

Outline

- Problem Description
- Motivation and Application
- Preliminaries
- Multi-Fix setup
    - Mathematical challenges
- Rectifiability check
- Experimental results
- Conclusion and Future work

Problem Description: Multi-error logic rectification



A faulty implementation of a 3-bit modulo multiplier

Problem Description

- Agnostic to the fault model, check for rectification at particular targets
  - Single-fix Rectification (SFR)
    - Correct circuit by changing function at a single net
- In a general setting, SFR might not be desired or may not exist
  - Multi-fix Rectification (MFR)
    - Correct circuit by changing functions at multiple nets
    - Contribution: Multi-fix rectifiability setup and check

Preliminaries: Finite field basics

- Fields - set of elements over which operations $(+, \cdot, /)$ can be performed
  - Ex. $\mathbb{R}, \mathbb{Q}, \mathbb{C}$
- Finite fields (Galois fields) - Finite set of elements
  - Ex. $\mathbb{F}_q$, where $q = p^n,\ p\ =\ prime,\ n \in \mathbb{Z}_{\geq 1}$
    - When $n = 1$, $\mathbb{F}_p = \mathbb{Z}_p \pmod{p}$
    - With $p = 2$, $\mathbb{F}_2 = \mathbb{B} = \{0, 1\}$
  - On circuits, $p = 2$, $n =$ data-operand width
- Hardware cryptography extensively based on $\mathbb{F}_{2^n}$ (we use $\mathbb{F}_{2^n}$)

Preliminaries: Modeling circuit polynomials over $\mathbb{F}_{2^n}$

- Boolean logic gates in $\mathbb{F}_2$ $(\mathbb{F}_2 \subset \mathbb{F}_{2^n})$. Over $\mathbb{F}_2$, $-1 = +1 \pmod 2$

$$
\begin{aligned}
z &= \ \sim a && \implies z + a + 1 && \pmod 2 \\
z &= a \wedge b && \implies z + a \cdot b && \pmod 2 \\
z &= a \vee b && \implies z + a \cdot b + a + b && \pmod 2 \\
z &= a \oplus b && \implies z + a + b && \pmod 2
\end{aligned}
$$

- Word-level polynomials [γ = primitive element of $\mathbb{F}_{2^n}$]

$$Output : Z + z_0 + \gamma \cdot z_1 + \cdots + \gamma^{n-1} \cdot z_{n-1}$$

$$Input : A + a_0 + \gamma \cdot a_1 + \cdots + \gamma^{n-1} \cdot a_{n-1}$$

Problem Statement and Objective

- A multivariate specification polynomial $f \in \mathbb{F}_{2^n}$
  - n is the operand width
  - Ex. $Z = A \cdot B \pmod{P_n(x)}$ over $\mathbb{F}_{2^n}$
- A faulty circuit implementation C for specification f
  - Model gates as polynomials over $\mathbb{F}_{2^n}$
- A primitive polynomial $P_n(x)$ used to construct $\mathbb{F}_{2^n}$
  - $\mathbb{F}_{2^n}$ constructed as $\mathbb{F}_{2^n} = \mathbb{F}_2[x] \pmod{P_n(x)}$
  - Let γ be one of the roots of $P_n(x)$, i.e. $P_n(\gamma) = 0$
- A set of m targets from C (modeled over $\mathbb{F}_{2^m}$)
- Check if C is rectifiable at these m targets

Algebraic Geometry: Ideals

- Let $R = \mathbb{F}_{2^n}[x_1, \ldots, x_d, Z]$
  - $\{f_1, \ldots, f_s\} \in R$

- In our context
  - $x_1, \ldots, x_d$: Variables (nets of the circuit)
  - Z: bit-vector representation for variables
  - $f_1, \ldots, f_s$: Polynomials from the circuit (logic gate relations)

- $J = \langle F \rangle = \langle f_1, \ldots, f_s \rangle \subseteq R$
  - $\{h_1 f_1 + \cdots + h_s f_s : \; h_i \in R\}$
  - Polynomials $f_1, \ldots, f_s$: *basis* or *generators* of J

- Vanishing Ideal: $J_0 = \langle F_0 \rangle = \langle x_1^2 + x_1, \ldots, x_d^2 + x_d, Z^{2^n} + Z \rangle$
  - Restrict solutions to x_i in $\mathbb{F}_2$, and solutions to Z in $\mathbb{F}_{2^n}$

Algebraic Geometry: Varieties

- $J = \langle F \rangle = \langle f_1, \ldots, f_s \rangle \subseteq \mathbb{F}_{2^n}[x_1, \ldots, x_d, Z]$
- Let $\boldsymbol{a} = (a_1, \ldots, a_d) \in \mathbb{F}_{2^n}^d$ *s.t.* $f_1(\boldsymbol{a}) = \cdots = f_s(\boldsymbol{a}) = 0$

$$V(J) = \text{Set of all } \{\boldsymbol{a}\} \text{ s.t. } \begin{cases} f_1(\boldsymbol{a}) = 0, \\ f_2(\boldsymbol{a}) = 0, \\ \vdots \\ f_s(\boldsymbol{a}) = 0 \end{cases}$$

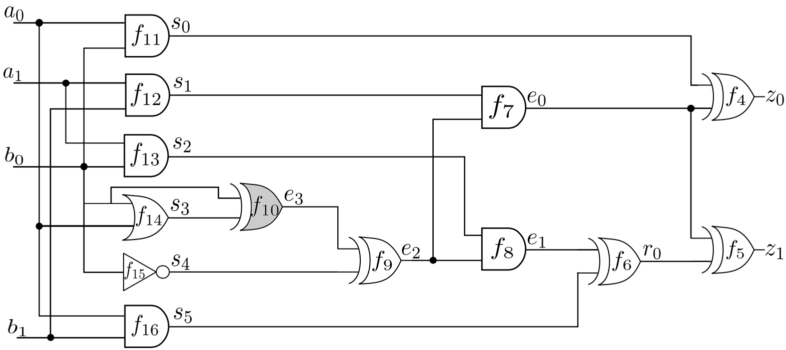
- $V(J)$ correspond to function mappings (Truth tables)

Gröbner Basis and Ideal membership

- An ideal $J = \langle f_1, \ldots, f_s \rangle \subseteq R$ can have many generators.
  - $J = \langle p_1, \ldots, p_m \rangle = \cdots = \langle g_1, \ldots, g_t \rangle$
  - Gröbner Basis (GB) is one such set with special properties

- Let $J = \langle f_1, \ldots, f_s \rangle = \langle g_1, \ldots, g_t \rangle$ and $G = GB(J) = \{g_1, \ldots, g_t\}$.
  - G is a Gröbner basis of $J \iff \forall f \in J, f \xrightarrow{g_1, \ldots, g_t}_+ 0$
  - Ideal membership: Let f be a polynomial in R:
    - if $f \xrightarrow{g_1, \ldots, g_t}_+ 0$, then f is a member of J.

- Circuit designed over $\mathbb{F}_{2^n} = \mathbb{F}_{2^2} (n = 2)$ using irreducible polynomial $P_n(x) = P_2(x) = x^2 + x + 1$ with $P_2(\gamma) = 0$



A 2-bit faulty modulo multiplier implementation.

- Denote polynomial $f : Z + A \cdot B$ as the design specification.
- Impose RTTO $>$

$f_1 : Z + z_0 + \gamma \cdot z_1;$
$f_2 : A + a_0 + \gamma \cdot a_1;$
$f_3 : B + b_0 + \gamma \cdot b_1;$
$f_4 : z_0 + s_0 + e_0;$
$f_5 : z_1 + e_0 + r_0;$
$f_6 : r_0 + e_1 + s_5;$
$f_7 : e_0 + s_1 e_2;$
$f_8 : e_1 + s_2 e_2;$
$f_9 : e_2 + e_3 + s_4;$
$f_{10} : e_3 + b_0 + s_3;$
$f_{11} : s_0 + a_0 b_0;$
$f_{12} : s_1 + a_1 b_1;$
$f_{13} : s_2 + a_1 b_0;$
$f_{14} : s_3 + a_0 + b_0 + a_0 b_0;$
$f_{15} : s_4 + b_0 + 1;$
$f_{16} : s_5 + a_0 b_1;$

- $F = \{f_1, \ldots, f_{16}\}$, $F_0 = \{a_0^2 - a_0, a_1^2 - a_1, b_0^2 - b_0, b_1^2 - b_1\}$
- Ideal Membership Test: $f \xrightarrow{F,F_0}_+ \gamma^1 \cdot (a_0 a_1 b_1 b_0 + a_0 a_1 b_1 + a_1 b_1 b_0 + a_1 b_0) + \gamma^0 \cdot (a_0 a_1 b_1 b_0 + a_0 a_1 b_1 + a_1 b_1 b_0)$

SFR Application: Rectification Check

1. Rectification check at net e_3: $W = \{e_3\}$
   - $J_1 = \langle F_1 \rangle$, where $F_1 = \{f_1, \ldots, f_{10} = e_3 + 0, \ldots, f_{16}\}$
   - $J_2 = \langle F_2 \rangle$, where $F_2 = \{f_1, \ldots, f_{10} = e_3 + 1, \ldots, f_{16}\}$
2. Compute r_1 and r_2:
   - $r_1 = f \xrightarrow{J_1, J_0}_+ (\gamma + 1)a_1 b_1 b_0 + (\gamma + 1)a_1 b_1$
   - $r_2 = f \xrightarrow{J_2, J_0}_+ (\gamma + 1)a_1 b_1 b_0 + (\gamma)a_1 b_0$
3. Single-fix rectification possible iff $V(r_1) \cup V(r_2) = \mathbb{F}_{2^3}^{|X_{PI}|} = V(J_0)$
   - Compute $G = GB(r1 \cdot r2, J_0)$ and check if $G = J_0$
   - In this example, target e_3 admits SFR

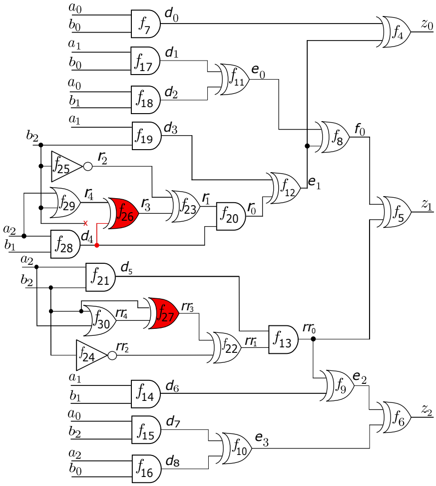
Unified framework motivation

- Single-fix is a special case of MFR with $m = 1$
  - Rectification patch modeled over $\mathbb{F}_{2^m} = \mathbb{F}_{2^1} = \mathbb{F}_2$
  - Circuit modeled over $\mathbb{F}_{2^n}$
    - Since $m = 1$ divides n, $\mathbb{F}_2 \subset \mathbb{F}_{2^n}$, $\forall n \in \mathbb{Z}_{>1}$
- For Multi-fix, since $m > 1$, $\mathbb{F}_{2^m}$ might not be contained in $\mathbb{F}_{2^n}$
  - Ex. $\mathbb{F}_{2^2}$ is not contained in $\mathbb{F}_{2^3}$, $m = 2, n = 3$
- Need a higher composite field $\mathbb{F}_{2^k}$ such that
  - $\mathbb{F}_{2^m} \subset \mathbb{F}_{2^k}$ and $\mathbb{F}_{2^n} \subset \mathbb{F}_{2^k}$
  - What are the mathematical challenges?
  - What primitive polynomial $P_K(x)$ should be used for constructing $\mathbb{F}_{2^k}$

Multi-fix Rectification: Prior work

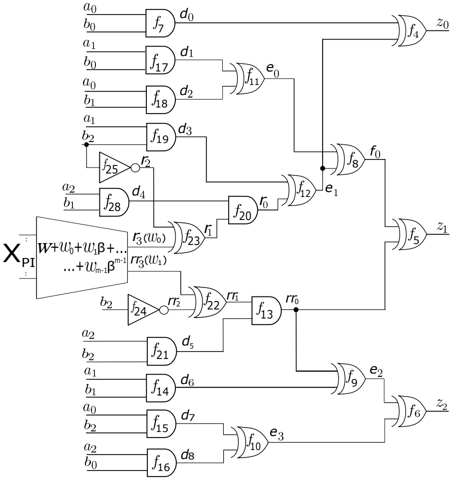
- Craig interpolation and/or iterative SAT solving [*Huang. et al*, DAC'11][*Huang. et al*, DATE'12]
  - Iteratively and incrementally patch the circuit
  - Compute multiple partial single-fix functions at the given m targets
- Resource aware ECO patch generation [*Jiang. et al*, DAC'18][*Mishchenko. et al*, DAC'18] [*Fujita. et al*, ISCAS'19]
- Approaches infeasible on arithmetic circuits
- Symbolic sampling technique [*Jiang. et al*, DAC'19]
  - Enumerate rectification points functionally and match the circuitry of patches implicitly
  - Scalability achieved by modeling computations in symbolic sampling domain

Application: Multi-fix Rectification



A faulty implementation of a 3-bit (n=3) Mastrovito multiplier

Application: Word-level representation



Patch function modeled as a 2-bit-vector word (m=2)

$$W = \{r_3, rr_3\} = r_3 + \beta \cdot rr_3, \ (w_0 = r_3, w_1 = rr_3).$$

MFR Notation: Field setup

- Circuit with data-path size n modeled over $\mathbb{F}_{2^n}$
  - $\mathbb{F}_{2^n}$ is constructed as $\mathbb{F}_{2^n} = \mathbb{F}_2[x] \pmod{P_n(x)}$
    - $P_n(x) \in \mathbb{F}_2[x]$ is a given degree-n primitive polynomial; $P_n(\gamma) = 0$
  - The word-level polynomials for Z, A are modeled as:
    - $f_Z : Z + \sum_{i=0}^{n-1} \gamma^i z_i,\ \ f_A : A + \sum_{i=0}^{n-1} \gamma^i a_i$
- Patch size m modeled over $\mathbb{F}_{2^m}$
  - $\mathbb{F}_{2^m}$ is constructed as $\mathbb{F}_{2^m} = \mathbb{F}_2[x] \pmod{P_m(x)}$
    - We select a degree-m primitive polynomial $P_m(x) \in \mathbb{F}_2[x]$; $P_m(\beta) = 0$
  - The word-level polynomial for W is modeled as:
    - $f_w : W + \sum_{i=0}^{m-1} \beta^i w_i$
    - $\{w_0, \ldots, w_{m-1}\} \subset \{x_1, \ldots, x_d\}$

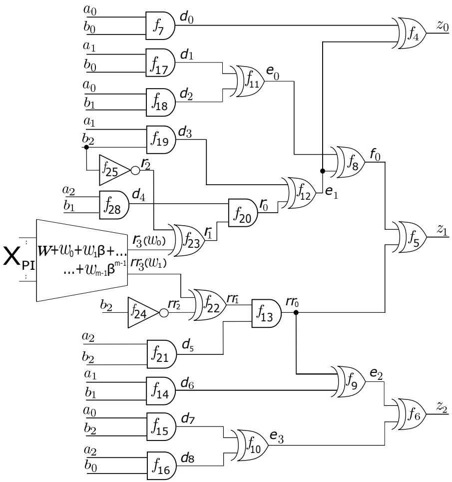
MFR Challenges: $\mathbb{F}_{2^k}$ and $P_k(x)$

- Smallest k is $LCM(n, m)$
  - $\mathbb{F}_{2^k} \supset \mathbb{F}_{2^n}$ and $\mathbb{F}_{2^k} \supset \mathbb{F}_{2^m}$
  - $\mathbb{F}_{2^k}$ is constructed as $\mathbb{F}_{2^k} = \mathbb{F}_2[x] \pmod{P_k(x)}$
    - $P_k(x)$ is a degree-k primitive polynomial; $P_k(\alpha) = 0$
- Mathematical challenge: Given $P_n(x)$ and $P_m(x)$, compute $P_k(x)$ such that $P_n(\gamma) = P_m(\beta) = P_k(\alpha) = 0$
  - $\gamma = \alpha^{(2^k-1)/(2^n-1)} = \alpha^{\lambda}$
  - $\beta = \alpha^{(2^k-1)/(2^m-1)} = \alpha^{\mu}$
- Solved using factorization of univariate polynomials over finite fields

Contribution: Computing $P_k(x)$

- Obtain UPFs of $P_n(x^\lambda)$ and $P_m(x^\mu)$ in $\mathbb{F}_2[x]$
- Then, $\exists P_k(x) \in \mathbb{F}_2[x]$ as a common factor of $P_n(x^\lambda)$ and $P_m(x^\mu)$, such that:
  - $P_k(x)$ is a degree-k primitive polynomial in $\mathbb{F}_2[x]$ with $P_k(\alpha) = 0$

Application: Word-level representation



Patch function modeled as a 2-bit-vector word (m=2)

$$W = \{r_3, rr_3\} = r_3 + \beta \cdot rr_3, \; (w_0 = r_3, w_1 = rr_3).$$

Application: Computing $P_k(x)$

- $P_3(x) = x^3 + x + 1, \;\; P_2(x) = x^2 + x + 1, \;\; \gamma = al^9, \;\; \beta = \alpha^{21}$
- Composite field: $k = LCM(2,3) = 6$
  - $UPF(P_3(x^9)) = (x^9)^3 + (x^9) + 1 = (x^6 + x^5 + x^2 + x + 1)(x^6 + x^5 + 1)(x^6 + x^4 + x^3 + x + 1)(x^6 + x^4 + x^2 + x + 1)(x^3 + x + 1)$;
  - $UPF(P_2(x^{21})) = (x^{21})^2 + (x^{21}) + 1 = (x^6 + x^5 + x^2 + x + 1)(x^6 + x^5 + 1)(x^6 + x^4 + x^3 + x + 1)(x^6 + x^5 + x^3 + x^2 + 1)(x^6 + x^5 + x^4 + x + 1)(x^6 + x + 1)(x^6 + x^3 + 1)$;
  - We choose $P_6(x) = x^6 + x^5 + 1$ as the required $P_k(x)$.

MFR Notation: Incorrect $P_k(x)$

- Note that if we incorrectly choose $P_k(x) = x^6 + x^3 + 1$
- For its root α, we have

$$
\begin{aligned}
\alpha^6 + \alpha^3 + 1 &= 0 \\
(\alpha^3)(\alpha^6 + \alpha^3 + 1) &= 0 \text{ (multiply by } \alpha^3\text{)} \\
\alpha^9 + \alpha^6 + \alpha^3 &= 0 \\
\gamma + 1 &= 0
\end{aligned}
\tag{1}
$$

- However, $\gamma \neq 1$, as γ is a primitive element of $\mathbb{F}_{2^n}$
- Selecting arbitrary $P_k(x)$ leads to erroneous results

MFR Notation: Word-level Formulation steps

- Modify field to $\mathbb{F}_{2^k}$ and compute $P_k(x)$
- Update ring by adding word-level target representation W
- Construct a polynomial set F' as follows:
  - Start with $F' = F$
  - Remove polynomials with w_i's as leading terms
  - Substitute for w_i's the respective word-level polynomials
  - Add $f_w : W + \sum_{i=0}^{m-1} \beta^i w_i$

MFR Application: Word-level Formulation

- 2-bit rectification patch over the 3-bit circuit can be performed over the field $\mathbb{F}_{2^6}$
  - Field $\mathbb{F}_{2^6} = \mathbb{F}_2[X] \pmod{P_6(X)}$
- Update polynomial set F to F' as:

$$F' = \{f_1, \dots, f_3, f'_4, f'_5, f_6, f'_7, f'_8, f_9, f_w, f_{11}, f_{13} \dots, f_{20}\}$$

$$\begin{aligned} f'_4 &: z_0 + (\beta W^2 + \beta^2 W) + d_0; & f'_5 &: z_1 + f_0 + (W^2 + W); \\ f'_7 &: f_0 + (\beta W^2 + \beta^2 W) + e_1; & f'_8 &: e_2 + (W^2 + W) + d_6; \\ f_w &: W + e_0 + \beta d_5; & \beta &= \alpha^{21}; \gamma = \alpha^9; \end{aligned}$$

MFR Contribution: Rectification Check

- Multi-fix rectification at target W
  - Construct the following ideals:
    - $J_i = \langle F_i' \rangle = \{f_1', \dots, f_w = W + \delta(i), \dots, f_s'\}$:$1 \leq i \leq 2^m$, $\delta(0) = 0, \delta(1) = 1, \delta(2) = \beta, \dots, \delta(2^m) = \beta^{2^m-2}$
  - Performing the reductions for all $1 \leq i \leq 2^m$:
    - $f \xrightarrow{F_i', F_0}_+ r_i$
  - Let $V_{\mathbb{F}_q}(r_i)$ denote the varieties of the respective r_i's
  - Multi-fix rectification exists at target W:

$$\textbf{if and only if } \bigcup_{i=1}^{2^m} V_{\mathbb{F}_q}(r_i) = \mathbb{F}_q^{|X_{PI}|} = V(J_0)$$

MFR Application: Rectification Check

- Constructing the J_i ideals:
  - $J_1 = \langle F'_1 \rangle$, where $F'_1[f_w] = W + \delta(1) = W$,
  - $J_2 = \langle F'_2 \rangle$, where $F'_2[f_w] = W + \delta(2) = W + 1$,
  - $J_3 = \langle F'_3 \rangle$, where $F'_3[f_w] = W + \delta(3) = W + \beta$,
  - $J_4 = \langle F'_4 \rangle$, where $F'_4[f_w] = W + \delta(4) = W + \beta^2$
- Reducing the specification $f : Z + A \cdot B$ modulo these ideals, we get:
  - $rem_1 = f \xrightarrow{F'_1 \cup F'_0}_+ \alpha^{27}(a_2 b_1 b_2) + \alpha^{36}(a_2 b_2)$
  - $rem_2 = f \xrightarrow{F'_2 \cup F'_0}_+ \alpha^{27}(a_2 b_1 b_2 + a_2 b_1) + \alpha^{36}(a_2 b_2)$
  - $rem_3 = f \xrightarrow{F'_3 \cup F'_0}_+ \alpha^{27}(a_2 b_1 b_2)$
  - $rem_4 = f \xrightarrow{F'_4 \cup F'_0}_+ \alpha^{27}(a_2 b_1 b_2 + a_2 b_1)$
- Compute $GB(r_1 \cdot r_2 \cdot r_3 \cdot r_4, F_0) = F_0$
- Target W with nets r_3 and rr_3 admits MFR

- A polynomial which can be computed to rectify the circuit
  - $W = a_2 b_1 b_2 + \beta \cdot a_2 b_2$
  - $r_3 = (a_2 \wedge b_1 \wedge b_2), \quad rr_3 = (a_2 \wedge b_2)$

Focus: Finite Field Arithmetic Circuits

- Applications:
  - RSA, ECC, Error correcting codes, RFID, etc.
    - Crypto-system bugs can leak secret keys [*Biham. et al*, Crypto'08]
    - RFID tag cloning could cause counterfeiting [*Batina. et al*, Security'09]
  - Large datapath sizes in ECC crypto systems
    - In $\mathbb{F}_{2^n}$, $n = 163, 233, 283, 409, 571$ (NIST standard)
- Rectification Motivation:
  - Synthesize sub-functions as opposed to complete redesign
  - Automated debugging

MFR Experiments: SINGULAR Implementation

Table: Word-level multi-fix rectifiability check against word level specification. Time is in seconds; rows marked '*' indicates $m \nmid n$; Benchmark = Mastrovito architecture, n = Datapath Size, #Gates = No. of gates, K = 10^3, m = patch size, k = encompassing composite field size, PF = time for polynomial factorization and computing minpoly for the composite field, RC = time for rectification check

n	#Gates	m	k	PF	RC
12	0.45K	2	12	NA	0.4
16	0.8K	2	16	NA	3.2
*16	0.8K	3	48	–	–
*20	0.0	3	60	–	–
32	2.8K	2	32	NA	184
48	6.4K	3	48	NA	–
64	11.2K	2	64	NA	–

MFR Experiments: Custom software

Table: Word-level multi-fix rectifiability check against word level specification. Time is in seconds; Benchmark = Mastrovito architecture, n = Datapath Size, #Gates = No. of gates, K = 10^3, m = word length of patch function, k = encompassing composite field size (degree of primpoly used), PF = time for polynomial factorization and computing minpoly for the composite field, PBS = PolyBori setup (ring declaration/poly collection/spec collection), VF = time for verification, MFS = Multi-fix check setup, MFRC = time for multi-fix rectification check, TE = Total execution time

n	**#Gates**	m	k	**PF**	**PBS**	*VF*	*MFS*	*MFRC*	*TE*
12	0.45K	2	12	$<$0.01	$<$0.01	$<$0.01	$<$0.01	$<$0.01	$<$0.01
12	0.45K	3	12	$<$0.01	$<$0.01	$<$0.01	$<$0.01	$<$0.01	$<$0.01
16	0.8K	2	16	$<$0.01	$<$0.01	$<$0.01	$<$0.01	$<$0.01	$<$0.01
16	0.8K	3	48	$<$0.01	$<$0.01	$<$0.01	$<$0.01	$<$0.01	$<$0.01
32	2.8K	2	32	$<$0.01	0.1	$<$0.01	$<$0.01	$<$0.01	0.15
64	11.2K	2	64	$<$0.1	0.5	$<$0.01	$<$0.01	0.2	0.9
96	24.5K	2	96	$<$0.1	1.4	0.1	$<$0.01	$<$0.01	1.7
128	43.2K	2	128	$<$0.3	3.1	0.3	$<$0.1	$<$0.1	3.6
163	69.8K	2	326	$<$0.4	6.2	2.0	$<$0.1	0.4	7.5
233	119K	2	466	$<$1	13.0	0.9	0.15	$<$0.1	14.3
283	190K	2	566	$<$2	39.0	2.1	0.2	$<$0.1	41.3
409	384K	2	818	$<$2	190	3.5	0.5	0.1	195.4
571	827K	2	1042	$<$3	2170	9.1	1.1	$<$0.1	2183

Rectification function computation

- SFR of finite field arithmetic circuits [*Rao. et al*, FMCAD'18][*Rao. et al*, IWLS'18]
  - Quantification based computation
  - Alternate to Craig Interpolation
- Currently addressing function computation at a word-level for finite field arithmetic circuits:
  - Rectification function computation at multiple nets in terms of primary inputs [Due notification GLSVLSI'21]
    - Define and formulate existence of don't cares
    - Devise algorithms to explore don't cares for logic optimization
  - Formulate rectification setup in terms of internal nets of the circuit.
    - Explore word-level don't care formulation in terms of internal nets.
  - Extend the multi-fix approach to integer arithmetic circuits and address the associated challenges.

THANK YOU

Email: vikas.k.rao@utah.edu