Error Detection for Borrow-Save Adders Dedicated to ECC Unit

Julien Francq, Jean-Baptiste Rigaud, Pascal Manet, Assia Tria, Arnaud Tisserand

Ecole Nationale Supérieure des Mines SAINT-ETIENNE

cea leti

Laboratoire d'Informatique de Robotique et de Microélectronique de Montpellier LIRMM

August 10, 2008

Part 1

Introduction

Elliptic Curves

- **Definition.** An elliptic curve over a finite field $\mathbb{F}_p$ (with p prime) is the set of points $(x, y) \in E$

$$E : y^2 = x^3 + ax + b \ \cup \ \{\infty\}$$

- **Fact.** $E(\mathbb{F}_p)$: additive group
  - Neutral element: ∞
  - Group operation: addition ($\oplus$) ["chord-and-tangent" law]
- **Addition formulæ.** Let $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$, $P_1 \oplus P_2 = (x_3, y_3)$ where

$$x_3 = \lambda^2 - x_1 - x_2 \text{ and } y_3 = (x_1 - x_3)\lambda - y_1$$

$$\text{with } \lambda = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1}, \text{ if } P_1 \neq \pm P_2 \text{ [Addition]} \\ \dfrac{3x_1^2 + a}{2y_1}, \text{ if } P_1 = P_2 \text{ [Doubling]} \end{cases}$$

Elliptic Curve Cryptography (ECC)

- [Miller, CRYPTO 1986], [Koblitz, MC, 1987]

Elliptic Curve Discrete Logarithm Problem (ECDLP)

Let $\mathbb{G} = \langle P \rangle = \{\infty, P, [2]P, \cdots, [n-1]P\} \subseteq E(\mathbb{F}_p)$,
with $n = \text{ord}_E(P)$ prime.
Given points $P,\ Q \in \mathbb{G}$, compute d such that
$Q = [d]P = \underbrace{P \oplus P \oplus \cdots \oplus P}_{d \text{ times}}$.

- ECDLP is intractable if elliptic curve parameters $(p,\ E,\ P,\ n)$ are carefully chosen.
- RSA-1024 $\simeq$ ECC-160

Fault Attacks and Countermeasures

- Attacks on ECC
  - Side-channel attacks
  - Fault attacks $[E \rightarrow \hat{E}$, where $n' = ord_{\hat{E}}(P/\hat{P}) < n]$
- Standard countermeasures for fault attacks protect $Q = [d]P$
- $\rightarrow$ Modular arithmetic's layer must be also protected...
- ...by using parity-preserving logic gates!
- Outline:

Fault Attacks and Countermeasures

- Attacks on ECC
  - Side-channel attacks
  - Fault attacks $[E \rightarrow \hat{E}$, where $n' = ord_{\hat{E}}(P/\hat{P}) < n]$
- Standard countermeasures for fault attacks protect $Q = [d]P$
- $\rightarrow$ Modular arithmetic's layer must be also protected...
- ...by using parity-preserving logic gates!
- Outline:
  - Parity-Preserving Logic Gates (PPLGs)

Fault Attacks and Countermeasures

- Attacks on ECC
  - Side-channel attacks
  - Fault attacks $[E \rightarrow \hat{E}$, where $n' = ord_{\hat{E}}(P/\hat{P}) < n]$
- Standard countermeasures for fault attacks protect $Q = [d]P$
- $\rightarrow$ Modular arithmetic's layer must be also protected...
- ...by using parity-preserving logic gates!
- Outline:
  - Parity-Preserving Logic Gates (PPLGs)
  - Implementing parity-preserving logic circuits

Fault Attacks and Countermeasures

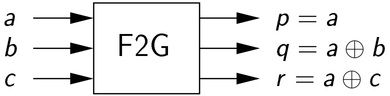
- Attacks on ECC
  - Side-channel attacks
  - Fault attacks $[E \rightarrow \hat{E}$, where $n' = ord_{\hat{E}}(P/\hat{P}) < n]$
- Standard countermeasures for fault attacks protect $Q = [d]P$
- $\rightarrow$ Modular arithmetic's layer must be also protected...
- ...by using parity-preserving logic gates!
- Outline:
  - Parity-Preserving Logic Gates (PPLGs)
  - Implementing parity-preserving logic circuits
  - Implementation results

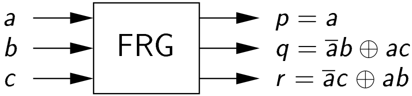
Part 2

PPLGs

PPLGs

- [Fredkin *et al.*, TP, 1982], [Feynman, ON, 1985], [Parhami, ACSSC 2006]
- Feynman double-gate (F2G), Fredkin gate (FRG)





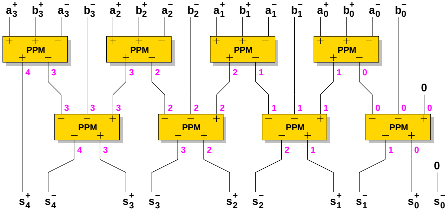
Part 3

Implementing Parity-Preserving Logic Circuits

Why Borrow-Save for Modular Arithmetic?

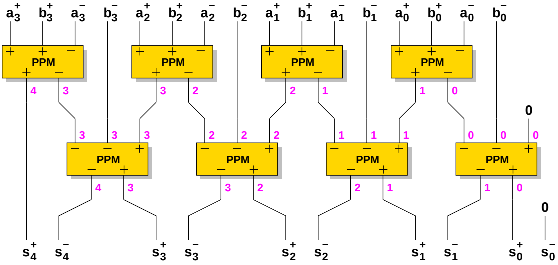
- Compute $Q = [d]P$ efficiently $\rightarrow$ efficient $+/-, \times, x^{-1}$
- $\Rightarrow$ Addition must be efficiently implemented!
  - $\rightarrow$ Borrow-Save Addition (BSA) without carry-propagation
- $A = (a_{l-1} \cdots a_1 a_0)_{BS}$ where $a_i \in \{-1, 0, 1\}$ are coded on 2 bits a_i^+ and a_i^- such that $a_i = a_i^+ - a_i^-$ and

$$A = \sum_{i=0}^{l-1} a_i 2^i = \sum_{i=0}^{l-1} (a_i^+ - a_i^-) 2^i$$



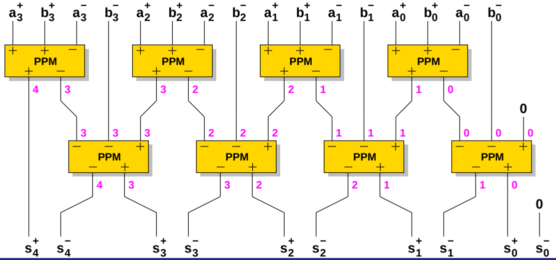
Implementing Parity-Preserving Circuits (1/3)

- 1.Choose the protected part of the circuit.
  - How many output bits are protected at a time, what is the protection level? / Performance
  - $\rightarrow$ 2 BSA output bits (s_{i+1}^{-}, s_i^{+})
  - 5 BSA input bits are concerned: a_i^{+}, b_i^{+}, a_i^{-}, b_i^{-}, c_i^{+}



Implementing Parity-Preserving Circuits (2/3)

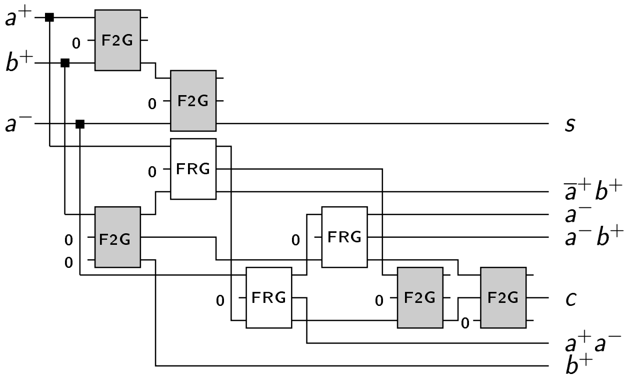
- 2.Get the corresponding logic equations.
  - Sufficiently simple circuit
  - 1^{st} row of PPMs $\left\{ \begin{array}{l} c_i^- = a_i^+ \oplus b_i^+ \oplus a_i^- \\ c_i^+ = a_{i-1}^+ . b_{i-1}^+ + a_{i-1}^+ . \overline{a_{i-1}^-} + b_{i-1}^+ . \overline{a_{i-1}^-} \end{array} \right.$
  - 2^{nd} row of PPMs $\left\{ \begin{array}{l} s_{i+1}^- = c_i^- . b_i^- + c_i^- . \overline{c_i^+} + b_i^- . \overline{c_i^+} \\ s_i^+ = c_i^- \oplus b_i^- \oplus c_i^+ \end{array} \right.$



Implementing Parity-Preserving Circuits (3/3)

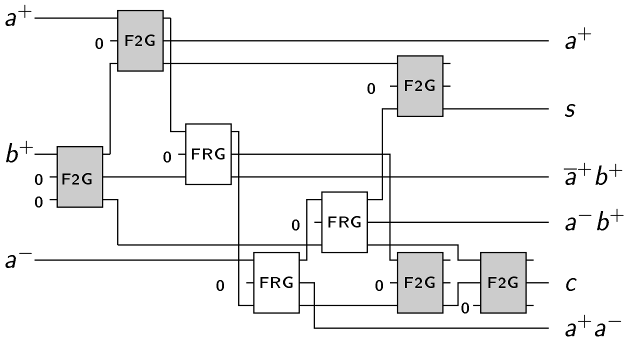
- **3.Transform the logic equations in Galois field.** (using $f + g = f \oplus g \oplus f.g$)
  - 1^{st} row of PPMs $\begin{cases} c_i^- = a_i^+ \oplus b_i^+ \oplus a_i^- \\ c_i^+ = a_{i-1}^+.b_{i-1}^+ \oplus a_{i-1}^+.\overline{a_{i-1}^-} \oplus b_{i-1}^+.\overline{a_{i-1}^-} \end{cases}$
  - 2^{nd} row of PPMs $\begin{cases} s_{i+1}^- = c_i^-.b_i^- \oplus c_i^-.\overline{c_i^+} \oplus b_i^-.\overline{c_i^+} \\ s_i^+ = c_i^- \oplus b_i^- \oplus c_i^+ \end{cases}$
- **4.Implement these equations thanks to PPLGs.**

PPM1 Cell



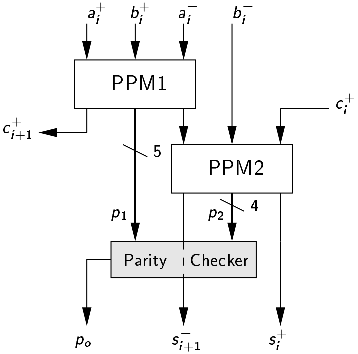
- Triplication: F2G($a = x, b = 0, c = 0,\ p = x, q = x, r = x$)
- "Garbage bits"

PPM2 Cell



- Triplication: F2G($a = x, b = 0, c = 0,\ p = x, q = x, r = x$)
- "Garbage bits"

Elementary Cell of our Fault-Tolerant BSA



$p_o = \beta_1 + \beta_2 + \beta_3$

With: $\beta_1 = a_i^+ \oplus b_i^+ \oplus a_i^- \oplus c_i^-$, $\beta_2 = a_i^+ \oplus b_i^+ \oplus a_i^- \oplus p_1 \oplus c_{i+1}^+$,

$\beta_3 = c_i^+ \oplus b_i^- \oplus c_i^- \oplus p_2 \oplus s_{i+1}^- \oplus s_i^+$

Part 4

Implementation Results

Performance

Architecture	Area (μm^2)	Latency (ns)
BSA-160 w/o EDC	134,440	1.39
BSA-160 with EDC	698,157	5.69
Overhead	**x5.2**	**x4.1**

Architecture	Area (μm^2)	Latency (ns)
ALU-160 w/o EDC	3,096,103	8.38
ALU-160 with EDC	4,270,313	19.96
Overhead	**x1.4**	**x2.4**

- ALU mainly consists in 2 BSAs, MUXs, shifts and registers
- Synthetized in C35 CORELIB technology using Design Vision

Evaluation of the Detection Capabilities

Number of faulty bits	**1 bit**	**2 bits**
Detected faults	80.0%	86.8%
Unfaulty computations	14.3%	2.1%
Undetected faults	**5.7%**	**11.1%**

Part 5

Conclusion and future works

Conclusion and future works

- Fault-tolerant elliptic curve cryptoprocessor unit...
- ...using parity-preserving logic gates.
- Protecting only borrow-save adders implies an acceptable area overhead (+40%)...
- ...but a less acceptable latency overhead (+140%).
- Improvement by using optimized PPLGs or reversible gates (e.g., Toffoli and Peres gate)
- Protect control logic

Affiliation

- ENSM-SE/SGC

  - Secure Embedded Systems And Microelectronics team
- LIRMM
  - ARITH team
    - Design of secured arithmetical operators
- Technological Bricks for Reinforcing Security project
  - Partners: CEA-LETI, Gemalto, Smart Packaging Solutions