Arithmetic Level Countermeasures for ECC Coprocessor

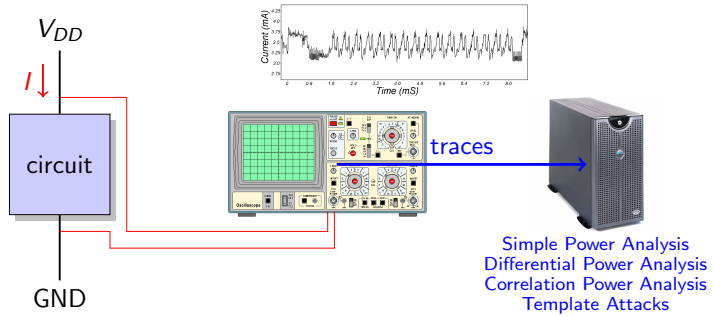
Arnaud Tisserand, Thomas Chabrier, Danuta Pamula

CNRS, IRISA laboratory, CAIRN research team

Claude Shannon Institute Workshop on Coding & Cryptography
May 17-18, 2010, UCC

Power Consumption Analysis

General principle: measure the current I in the circuit



Notations: V_{DD} power supply (5, 3, 2.5, 1.2, 0.9 V), GND ground

Similar attacks: electromagnetic radiations (EMR) and timing analysis

Countermeasures

Prevent attacks by using:

- additional protection block(s)
- modification(s) of the original circuit (i.e. secure version)

Examples:

- electrical shielding
- use uniform computation durations
- use uniform power consumption
- add noise (e.g. useless instructions/computations)
- circuit reconfiguration at runtime
  - modify the datapath
  - modify the representation of values
  - modify the computation algorithms

Our solution: arithmetic level protection(s)

Arithmetic Operators for Cryptography

Values are elements of:

- prime finite field $\mathbb{F}_p$ (p is a large prime)
- extensions of the binary field $\mathbb{F}_{2^m}$
- extensions of small fields $\mathbb{F}_{p^m}$ (e.g.: $p = 3$)

Typical sizes for public-key cryptography:

- RSA $\Longrightarrow$ 1024 to 8192 bits
- ECC $\Longrightarrow$ 160 to 600 bits

Operations:

- addition, subtraction
- multiplication
- multiplication by a constant
- inversion
- exponentiation

Addition Modulo M

Inputs:

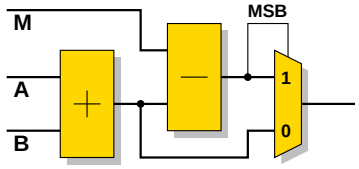
$$A, B \in \{0, 1, 2, 3, \ldots, M-1\}$$

Output[1]:

$$(A + B) \bmod M$$

Algorithm:

$$(A + B) \bmod M = \begin{cases} A + B & \text{if } A + B < M \\ A + B - M & \text{if } A + B \geq M \end{cases}$$



[1] $0 \leq A + B \leq 2M - 2$

Addition Modulo $2^n - 1$

Inputs:

$$A, B \in \{0, 1, 2, 3, \ldots, 2^n - 2\}$$

Output:

$$(A + B) \bmod (2^n - 1)$$

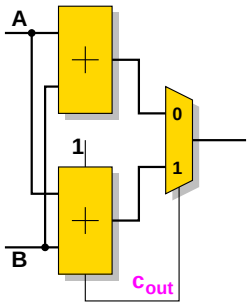
Basic method:

$$(A + B) \bmod (2^n - 1) = \begin{cases} A + B & \text{if } A + B < 2^n - 1 \\ \underbrace{A + B - (2^n - 1)}_{A+B+1} & \text{if } A + B \geq 2^n - 1 \end{cases}$$

Problem: the test $A + B \geq 2^n - 1$ is costly

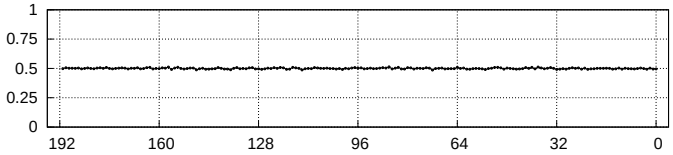
Addition Modulo $2^n - 1$: improved version

$$(A + B) \bmod (2^n - 1) = \begin{cases} A + B & \text{if } A + B + 1 < 2^n \\ A + B + 1 & \text{if } A + B + 1 \geq 2^n \end{cases}$$

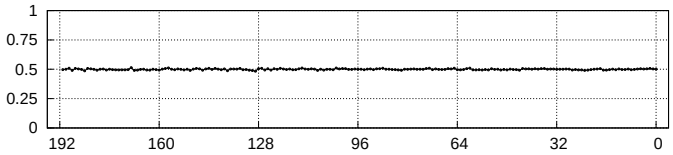


Activity in $\mathbb{F}_p$ Arithmetic Operators (1/2)

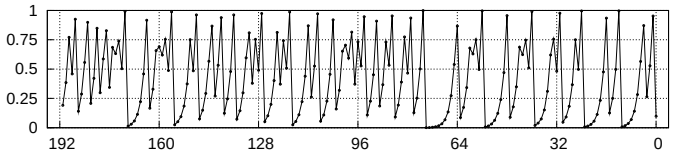
$a + b$



$a \times b$

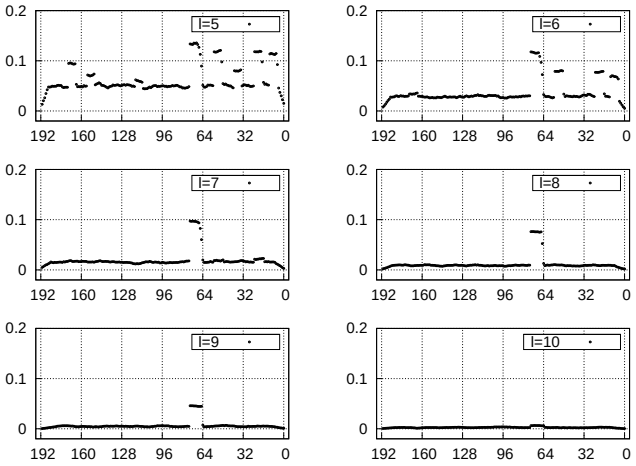


$a + cst$



a and b are random elements of $\mathbb{F}_p$, cst is a constant (curve parameter)

Activity in $\mathbb{F}_p$ Arithmetic Operators (2/2)



Activity profiles for l-bit windows (same transitions for all the bits of the window)

Modular Exponentiation for RSA

Algorithm: *square and multiply*

Inputs: x, $d = (d_{m-1} \dots d_1 d_0)_2$
Output: $y = x^d$

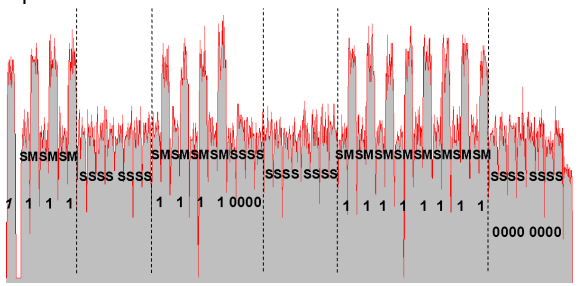
```
1  R ⟵ 1
2  i ⟵ m − 1
3  while (i ≥ 0) do
4      R ⟵ R²                    square
5      if (d_i = 1) then
6          R ⟵ R × x             multiply
7      endif
8      i ⟵ i − 1
9  endwhile
10 return R
```


Square and multiply is Weak!

Attack: SPA
Difference at each loop iteration:

- $d_i = 1 \Longrightarrow$ square and multiply
- $d_i = 0 \Longrightarrow$ square only

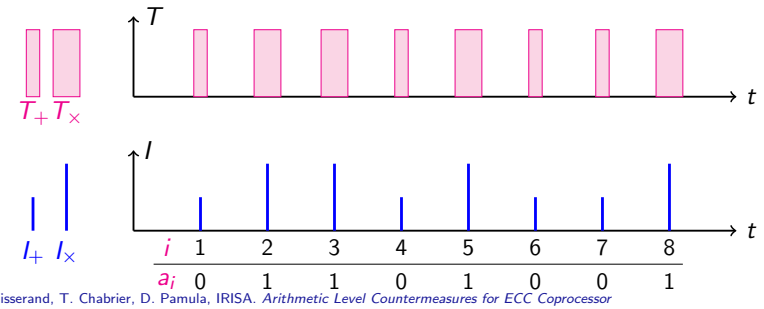
Trace example:



Differences & External Signature

An algorithm has a current signature and a time signature:

```
r = c0
for i from 1 to n do
  if a_i = 0 then
    r = r + c1
  else
    r = r × c2
```



SPA Countermeasure: *Square and multiply always*

Inputs: x, $d = (d_{m-1} \dots d_1 d_0)_2$
Output: $y = x^d$

---

1 $R \longleftarrow 1$
2 $i \longleftarrow m-1$
3 **while** $(i \geq 0)$ **do**
4 $\quad R_1 \longleftarrow R^2$ *square*
5 $\quad R_2 \longleftarrow R_1 \times x$ *multiply*
6 $\quad$ **if** $(d_i = 1)$ **then**
7 $\quad\quad R \longleftarrow R_2$
8 $\quad$ **else**
9 $\quad\quad R \longleftarrow R_1$
10 $\quad$ **endif**
11 $\quad i \longleftarrow i-1$
12 **endwhile**
13 **return** R

ECC: Scalar Multiplication

This is the main operation for ECC

Inputs: P a point of the curve E, a large integer $k = \sum_{i=0}^{n-1} k_i 2^i$
Output: the point $Q = [k]P = \underbrace{P + P + P + \dots + P}_{k \text{ times}}$

Basic algorithm: *double-and-add*

1: $Q \longleftarrow P$
2: **for** i **from** n-2 **to** 0 **do**
3: $\quad Q \longleftarrow 2P$
4: $\quad$ **if** $k_i = 1$ **then** $Q \longleftarrow Q + P$

Same problem: weak for SPA!

Countermeasure: Key Recoding

Recoding: w-NAF (*non-adjacent form*)
With

$$k = \sum_{i=0}^{n-1} k_i 2^i, \quad k_i \in \{0, 1\}$$

use k with digits in "windows" of w bits

$$|k_i| < 2^{w-1}$$

Example:

$$\begin{array}{rlllllllll} k = 267 = & (& 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 &)_2 \\ & (& 1 & 0 & 0 & 0 & 1 & 0 & \overline{1} & 0 & \overline{1} &)_{2-NAF} \\ & (& 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 3 &)_{3-NAF} \\ & (& 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & \overline{5} &)_{4-NAF} \\ & (& 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 11 &)_{5-NAF} \end{array}$$

Cost: $n-1$ DBL and $\frac{n}{w+1}$ ADD

Notation: $\overline{d} = -d$ where d is a digit

Addition Chains (PhD thesis of Nicolas Méloni)

In scalar multiplication $[k]P$, only use point additions on the curve

- robust against SPA
- $ADD(P_1, P_2) = (P_1 + P_2, P_1)$ with P_1 and P_2 already computed
- problem find a short chain

Example: addition chains for $k = 113$

1	1	2	1	1	6	1	1	14	14	14	14	14	14	14	
1	2	3	5	6	7	13	14	15	29	43	57	71	85	99	113

1	1	1	1	4	5	5	14	14	19	47	
1	2	3	4	5	9	14	19	33	47	66	113

Collaboration with UCC code and crypto group (2006–2008)

Signed-Digit Redundant Number Systems

Avizienis 1961: radix β representation

- replace the digit set $\{0, 1, 2, \ldots, \beta - 1\}$
- by the digit set $\{-\alpha, -\alpha + 1, \ldots, 0, \ldots, \alpha - 1, \alpha\}$ with $\alpha \leq \beta - 1$

If $2\alpha + 1 > \beta$ some numbers have several possible representations

Example: radix $\beta = 10$, digits from the set $\mathcal{D} = \{\overline{9}, \ldots, \overline{1}, 0, 1, \ldots, 9\}$

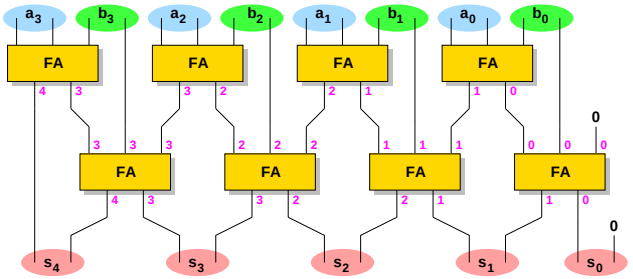
$$\begin{aligned} 2010 &= (2010)_{\beta,\mathcal{D}} \\ &= (21\overline{9}0)_{\beta,\mathcal{D}} \\ &= (3\overline{99}0)_{\beta,\mathcal{D}} \\ &= (1\overline{8}010)_{\beta,\mathcal{D}} \\ &= (1\overline{8}1\overline{9}0)_{\beta,\mathcal{D}} \\ &= \ldots \end{aligned}$$

In a redundant number system there is constant-time addition algorithm (without carry propagation) where all computations are done in parallel

Carry-Save Adder

In carry-save, the number A is represented in radix 2 using digits $a_i \in \{0, 1, 2\}$ coded by 2 bits such that $a_i = a_{i,c} + a_{i,s}$ where $a_{i,c} \in \{0, 1\}$ and $a_{i,s} \in \{0, 1\}$

$$A = \sum_{i=0}^{n-1} a_i 2^i = \sum_{i=0}^{n-1} (a_{i,c} + a_{i,s}) 2^i$$

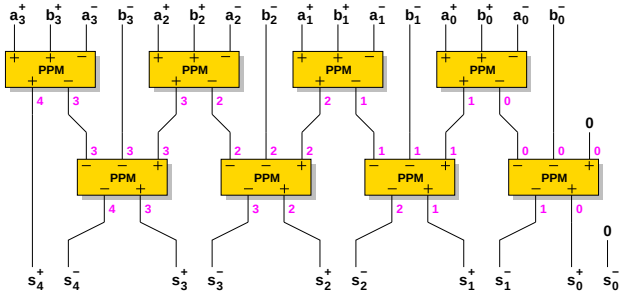


Carry-save addition: delay of 2 FA cells ($T = 0(1)$)

Borrow-Save Addition

In borrow-save, the number A is represented in radix 2 using digits $a_i \in \{-1, 0, 1\}$ coded by 2 bits such that $a_i = a_i^+ - a_i^-$ where $a_i^+ \in \{0, 1\}$ and $a_i^- \in \{0, 1\}$

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



Borrow-save addition: delay of 2 PPM cells ($T = 0(1)$)

PPM Cell

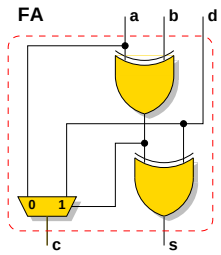
a^+	b^+	d^-	c^+	s^-
0	0	0	0	0
0	0	1	0	1
0	1	0	1	1
0	1	1	0	0
1	0	0	1	1
1	0	1	0	0
1	1	0	1	0
1	1	1	1	1

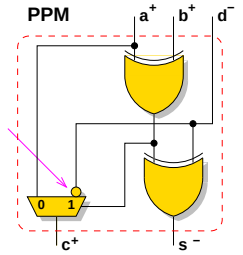
Arithmetic equation:

$$2c^+ - s^- = a^+ + b^+ - d^-$$

Logic equation:

$$\begin{aligned} s &= a^+ \oplus b^+ \oplus d^- \\ c &= a^+ b^+ + a^+ \overline{d^-} + b^+ \overline{d^-} \end{aligned}$$





Double-Base Number Systems (DBNS) (1/3)

Redundant representation based the sum of powers of 2 AND 3:

$$x = \sum_{i=1}^{n} x_i 2^{a_i} 3^{b_i}, \text{ with } x_i \in \{-1, 1\},\ a_i, b_i \geq 0$$

Example: $127 = 108 + 16 + 3 = 72 + 54 + 1 = \ldots$

	1	2	4	8	16
1					1
3	1				
9					
27			1		

	1	2	4	8
1	1			
3				
9				1
27		1		

Source: L. Imbert

Double-Base Number Systems (DBNS) (2/3)

Smallest $x > 0$ with n DBNS terms in its decomposition:

n	unsigned	signed
2	5	5
3	23	105
4	431	(4985)
5	18,431	?
6	3,448,733	
7	1,441,896,119	
8	?	

DBNS is a very sparse and redundant representation

Example: 127 has 783 DBNS representations among which 6 are canonic: $127 = (108 + 18 + 1) = (108 + 16 + 3) = (96 + 27 + 4) = (72 + 54 + 1) = (64 + 54 + 9) = (64 + 36 + 27)$

Double-Base Number Systems (DBNS) (3/3)

Application: ECC scalar multiplication
$314159 = 2^4 3^9 + 2^8 3^1 - 1$
$[314159]P = [2^4 3^9]P + [2^8 3^1]P - P$
cost: 12 DBL + 10 TPL + 2 ADD

$314159 = 2^4 3^9 - 2^0 3^6 - 3^3 - 3^2 - 3 - 1$
$[314159]P = 3(3(3(3^3([2^4 3^3]P - P) - P) - P) - P) - P$
cost: 4 DBL + 9 TPL + 5 ADD

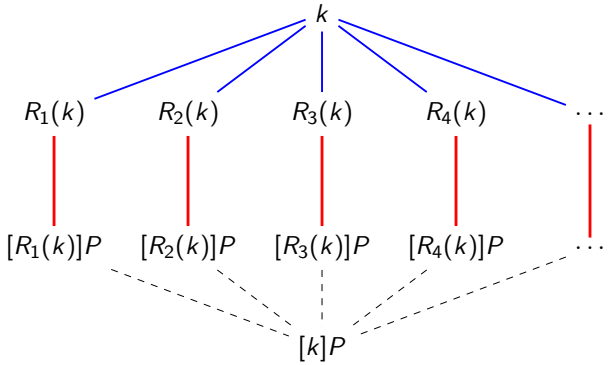
Recoding rules:

- $1 + 2 \longleftrightarrow 3$
- $1 + 3 \longleftrightarrow 4$
- $\ldots$

Protection at the Arithmetic Level

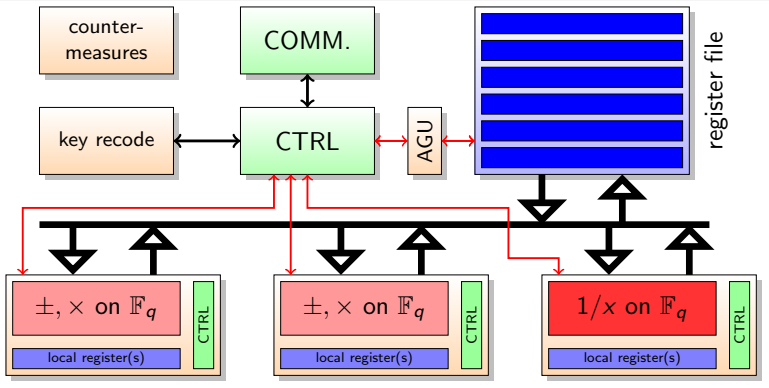
Redundant number system =

- a way to improve the performance of some operations
- a way to represent a value with different representations



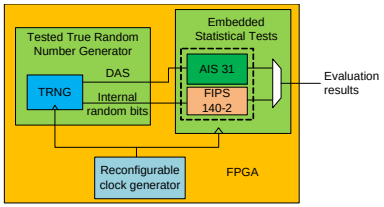
Proposed solution: use random redundant representations of k

Towards an ECC (co)Processor



- Functional units (FU): $\pm, \times, 1/x$ for $\mathbb{F}_p$ and $\mathbb{F}_{2^m}$, key recoding
- Memory: register file + internal registers in the FUs
- Control: operations (E and $\mathbb{F}_q$ levels) schedule, parameters management...

TRNG Circuits with On-Line Quality Evaluation

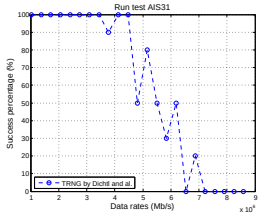


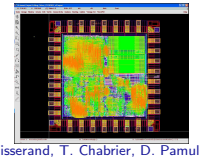
Quality of a TRNG depends on:

- type of TRNG
- target circuit (FPGA, ASIC,...)
- V_{dd}, EMR, temp., attacks...
- data rate (Mbit/s)

Objectives:

- TRNGs with embedded quality tests for security applications
- Comparison of various TRNGs
- Find optimal data rate of a TRNG





- ASIC: 130 nm circuit HCMOS9GP (CMP) V_1: June 09 (100% OK), V_2: 2010Q3
- FPGAs: Xilinx, Altera, Actel

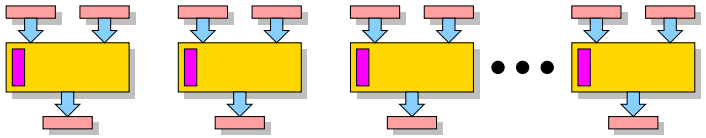
Residue Number System (RNS)

- Base $\mathcal{B} = (m_1, m_2, \ldots, m_k)$ of k relatively prime moduli
- Size of the base: k

$$A = \{a_1, a_2, \ldots, a_k\}, \quad \forall i \ a_i = A \bmod m_i$$

Operations:

$$A \pm B = (|a_1 \pm b_1|_{m_1}, \ldots, |a_k \pm b_k|_{m_k})$$
$$A \times B = (|a_1 \times b_1|_{m_1}, \ldots, |a_k \times b_k|_{m_k})$$



Residue Number System: Example (1/2)

Base: $\mathcal{B} = (8, 7, 5, 3)$

Dynamic range: $M = 8 \times 7 \times 5 \times 3 = 840$, i.e., $0 \leq A < M$

A_{std}	A_{RNS}
0	[0, 0, 0, 0]
1	[1, 1, 1, 1]
2	[2, 2, 2, 2]
3	[3, 3, 3, 0]
4	[4, 4, 4, 1]
5	[5, 5, 0, 2]
6	[6, 6, 1, 0]
7	[7, 0, 2, 1]
8	[0, 1, 3, 2]

A_{std}	A_{RNS}
9	[1, 2, 4, 0]
10	[2, 3, 0, 1]
11	[3, 4, 1, 2]
12	[4, 5, 2, 0]
13	[5, 6, 3, 1]
14	[6, 0, 4, 2]
15	[7, 1, 0, 0]
16	[0, 2, 1, 1]
17	[1, 3, 2, 2]

A_{std}	A_{RNS}
18	[2, 4, 3, 0]
19	[3, 5, 4, 1]
20	[4, 6, 0, 2]
21	[5, 0, 1, 0]
22	[6, 1, 2, 1]
23	[7, 2, 3, 2]
24	[0, 3, 4, 0]
25	[1, 4, 0, 1]
26	[2, 5, 1, 2]

Residue Number System: Example (2/2)

Operands: $A = 6 = [6, 6, 1, 0]$ and $B = 16 = [0, 2, 1, 1]$

Addition:

- $(6 + 0) \bmod 8 = 6$
- $(6 + 2) \bmod 7 = 1$
- $(1 + 1) \bmod 5 = 2$
- $(0 + 1) \bmod 3 = 1$

Verification: $22 = [6, 1, 2, 1]$

Multiplication:

- $(6 \times 0) \bmod 8 = 0$
- $(6 \times 2) \bmod 7 = 5$
- $(1 \times 1) \bmod 5 = 1$
- $(0 \times 1) \bmod 3 = 0$

Verification: $96 = [0, 5, 1, 0]$

Residue Number System: Conversions

From standard to RNS:

$$\forall i \quad a_i = A \bmod m_i$$

From RNS to standard:
Using a constructing proof of the Chinese Remainder Theorem (CRT)

$$A = \sum_{i=1}^{k} a_i M_i |M_i^{-1}|_{m_i} \bmod M$$

where

- $M = \prod_{i=1}^{k} m_i$, $A < M$
- $M_i = M/m_i$
- $|M_i^{-1}|_{m_i}$ is the inverse of M_i modulo m_i

Residue Number System: Summary

Advantages:

- parallel addition/subtraction and multiplication
- no carry propagation (between the blocks)
- natural way to split large numbers $\Longrightarrow$ simple scheduling
- no order in the elements (RNS is not a positional number system)

Disadvantages:

- difficult comparison ($<$ and $>$)
- difficult division
- difficult sign test
- difficult magnitude computation

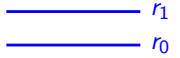
Circuit-Level Representations of Digits

Standard representation of a bit b:

- $V_{DD} \Longrightarrow b = 1$, GND $\Longrightarrow b = 0$

Dual-rail representation of a bit b:

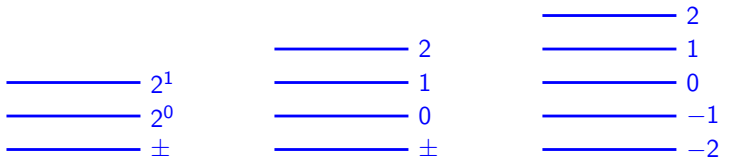
- $r_1 = V_{DD}$ $r_0 = GND \Longrightarrow b = 1$
- $r_1 = GND$ $r_0 = V_{DD} \Longrightarrow b = 0$



Benefit: same number of transitions for $0 \rightarrow 1$ and $1 \rightarrow 0$

Cost: larger area and memory

High-radix coding: radix 4 with digits in $\{-2, -1, 0, 1, 2\}$



Conclusion

- attacks are more and more efficient
- security is mandatory at all levels (specification, algorithm, operation, implementation)
- security = tradeoff between performances and robustness
- security = computer science + microelectronics + mathematics

Current research topics:

- redundant number systems
- non-positional number systems
- circuit reconfigurations (representations, algorithms)
- circuits with reduced activity variations
- links between scheduling and circuit activity
- design space exploration

The end, some questions ?

Contact:

- mailto:arnaud.tisserand@irisa.fr
- http://www.irisa.fr/prive/Arnaud.Tisserand/
- CAIRN Group http://www.irisa.fr/cairn/
- IRISA Laboratory, CNRS–INRIA–Univ. Rennes 1
  6 rue Kérampont, BP 80518, F-22305 Lannion cedex, France

Thank you