Redundant Number Systems for Reconfigurable Arithmetic Units

Arnaud Tisserand

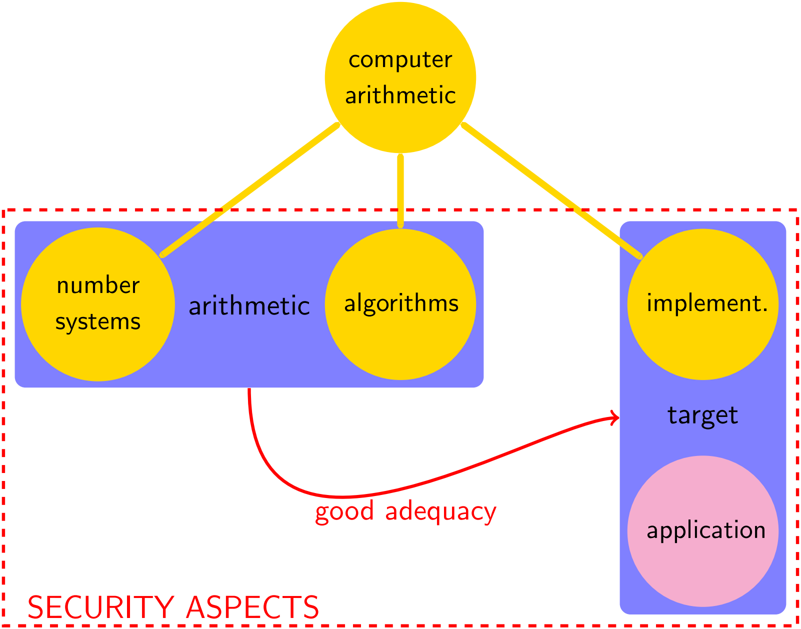
IRISA, CNRS – Univ. Rennes 1
CAIRN Group

CryptArchi
Prague, June 24–27, 2009

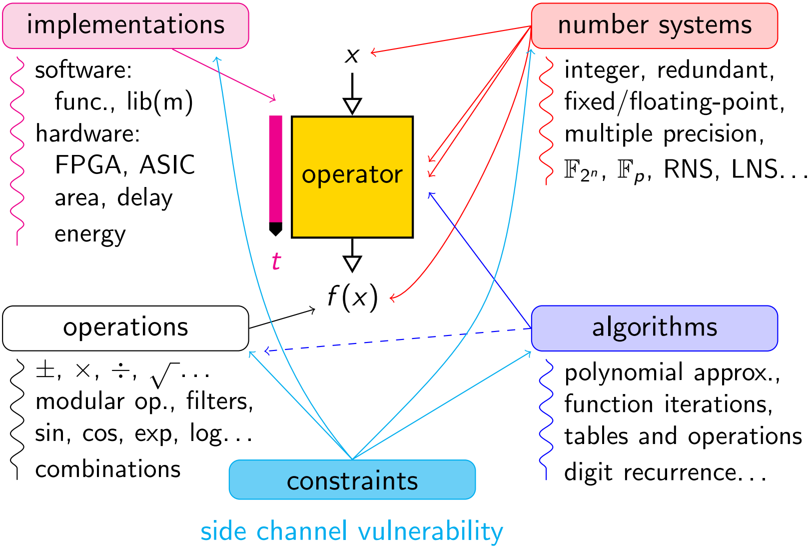
Plan

- Introduction
- Number Systems Examples
- Representation and Coding of Numbers in Hardware
- Reconfigurable Arithmetic Units for Crypto
- Future Prospects

Introduction to Computer Arithmetic



Arithmetic Operators Design



Practical Problems in Computer Arithmetic

- limited support in design tools

  software: integer, floating-point, libraries (standard problems)
  hardware: integer, fixed-point, a few IP blocs (standard problems)

- validation
  verification of the correctness of a program (function, library, hardware bloc, circuit) at design time

- test
  verification of the correctness of an implementation

Our goals:

- automatic generation of low-level descriptions (C and/or VHDL)
- (long term) include new arithmetic types and primitives in design tools (compilers, CAD tools)

(Basic) Positional Number Systems

$$A = \sum_{i=0}^{n-1} a_i \beta^i$$

- β is the radix or base (an integer and non-negative value here)
- i is the rank or index of digit a_i, β^i its weight
- n is the length of the representation (i.e. the number of digits)
- digits a_i are elements of the digit set $\mathcal{D}$ (here a set of consecutive integers)
- impact of the digit set size $|\mathcal{D}|$ and the radix β
  - $|\mathcal{D}| < \beta$ some number cannot be represented
  - $|\mathcal{D}| = \beta$ all numbers can be represented by a unique representation
  - $|\mathcal{D}| > \beta$ all numbers can be represented and some of them have several representations, the system is redundant

Booth Recoding

In 1951, Booth proposed to increase the number of 0s in the multiplier operand using the digit set $\{-1 = \overline{1}, 0, 1\}$

Recoding based on the identity:
$2^{i+k} + 2^{i+k-1} + 2^{i+k-2} + \cdots + 2^{i} = 2^{i+k+1} - 2^{i}$

Example: the integer 60 is represented by $00111100 = 01000\overline{1}00$

The recoding replaces strings of 1s by a representation with more 0s

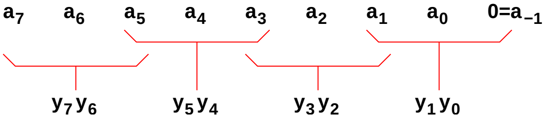
But, in some cases, this basic method leads to more 1 (or $\overline{1}$)!

Example: the value 01010101 is recoded to $\overline{1}1\overline{1}1\overline{1}1\overline{1}1$

$\Longrightarrow$ modified Booth's recoding

Modified Booth's Recoding

Idea: do not recode isolated 1 but only strings of 1

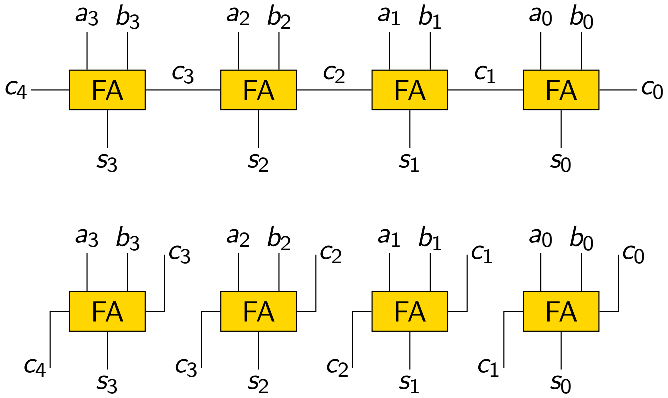


a_i	a_{i-1}	a_{i-2}	y_i	y_{i-1}	meaning	operation
0	0	0	0	0	string of 0s	$+0$
0	0	1	0	1	end of a string of 1s	$+B$
0	1	0	0	1	isolated 1	$+B$
0	1	1	1	0	end of a string of 1s	$+2B$
1	0	0	$\overline{1}$	0	beginning of a string of 1s	$-2B$
1	0	1	1	$\overline{1}$	isolated 0	$-B$
1	1	0	0	$\overline{1}$	beginning of a string of 1s	$-B$
1	1	1	0	0	middle of a string of 1s	$+0$

Improvement: leads to a n-product with $\lfloor n/2 \rfloor + 1$ additions and shifts at most

Carry-Save Representation

Simple (but efficient) idea: save carries instead of propagating them

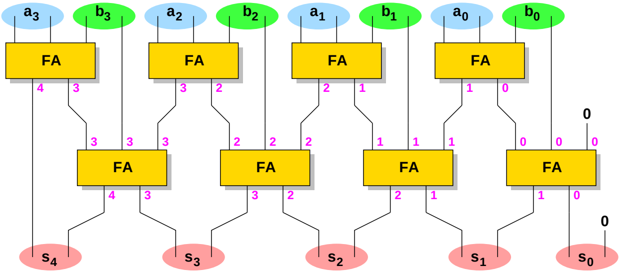


Widely used in multipliers.

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 $a_{i,s} \in \{0, 1\}$

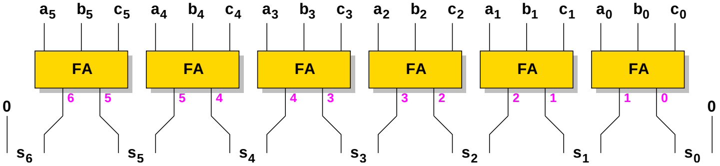
$$A = \sum_{i=0}^{n-1} a_i 2^i = \sum_{i=0}^{n-1} (a_{i,c} + a_{i,s}) 2^i$$



A carry-save addition is performed with the delay of 2 FA cells ($T = 0(1)$)

Carry-Save Trees

Example with 3 inputs: A, B and C



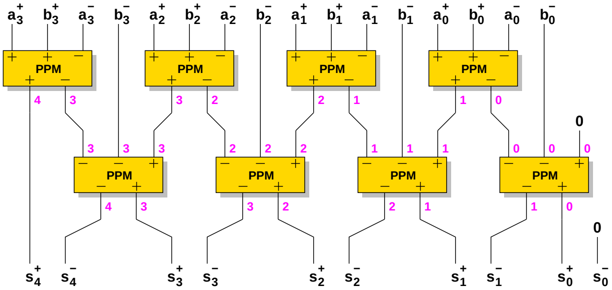
Carry-save reduction tree: $n(h)$ non-redundant inputs can be reduced by a h-level carry-save tree where $n(h) = \lfloor 3n(h-1)/2 \rfloor$ and $n(0) = 2$

h	1	2	3	4	5	6	7	8	9	10	11
$n(h)$	3	4	6	9	13	19	28	42	63	94	141

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 $a_i^- \in \{0, 1\}$

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



A borrow-save addition is performed with the delay of 2 PPM cells ($T = 0(1)$)

PPM Cell

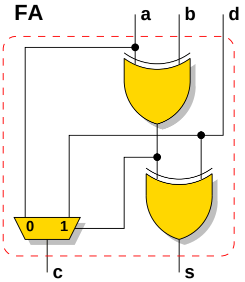
a^+	b^+	d^-	c^+	s^-
0	0	0	0	0
0	0	1	0	1
0	1	0	1	1
0	1	1	0	0
1	0	0	1	1
1	0	1	0	0
1	1	0	1	0
1	1	1	1	1

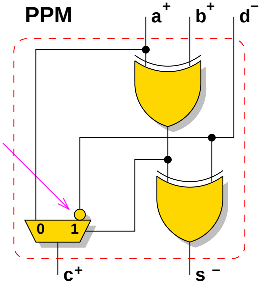
Arithmetic equation:

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

Logic equation:

$$s = a^+ \oplus b^+ \oplus d^-$$
$$c = a^+ b^+ + a^+ \overline{d^-} + b^+ \overline{d^-}$$





Borrow-Save Example for ECC Protection Against Fault Attacks

J. Francq, J.-B. Rigaud, P. Manet, A. Tria, A. Tisserand
Error Detection for Borrow-Save Adders Dedicated to ECC Unit
5th Workshop on Fault Detection and Tolerance in Cryptography (FDTC) 2008

Incorporation of a fault detection method based on parity-preserving logic gates in some parts of an elliptic curve unit implemented using borrow-save representation.

design	area [μm^2]	latency [ns]
original	3,096,103	8.38
borrow-save protected	4,270,313	19.96
overhead	$\times 1.38$	$\times 2.38$

Signed Digit Redundant Number Systems

In 1961, Avizienis suggested to represent numbers in radix β with digits in $\{-\alpha, -\alpha+1, \ldots, 0, \ldots, \alpha-1, \alpha\}$ instead of $\{0, 1, 2, \ldots, \beta-1\}$ with $\alpha \leq \beta - 1$

Using this representation, if $2\alpha + 1 > \beta$ some numbers have several possible representation at the bit level. For instance, the value 2345 (in the standard representation) can be represented in radix 10 with digits in $\{-5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5\}$ by the values 2345, 235(-5) or 24(-5)(-5)

Such a representation is said redundant

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

Are Redundant Number Systems New?

not really...

105.

Calculs numériques. — *Sur les moyens d'éviter les erreurs dans les calculs numériques.*

C. R., t. XI, p. 789 (16 novembre 1840).

Les nombreux exemples que l'on pourrait citer d'erreurs commises, quelquefois par des calculateurs fort habiles, dans la réduction des formules en nombres, doivent faire rechercher avec soin les moyens de vérifier l'exactitude des résultats numériques auxquels on se trouve conduit par une suite d'opérations déterminées. Or, pour que l'on

⋮

ces deux espèces de chiffres, il faudra remplacer chaque suite continue de chiffres négatifs, situés immédiatement l'un après l'autre, par le complément arithmétique de cette suite, et diminuer d'une unité le chiffre positif qui la précède. Ainsi, par exemple, on aura

$$1\bar{1}=9, \qquad 1\bar{2}1=81,$$
$$10\bar{2}\bar{4}5\bar{3}12\bar{4}\bar{2}=976471158.$$

A. Cauchy
Sur les moyens d'éviter les erreurs dans les calculs numériques
Comptes rendus de l'Academie des Sciences, pp 431–442, nov. 1840, Oeuvre complètes (tome 5, série 1), extrait n. 105.

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

Redundant representation based on a sum of mixed 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: $127 = 108 + 16 + 3 = 72 + 54 + 1 = \dots$

	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

DBNS (2/3)

Smallest $x > 0$ requiring n terms in DBNS:

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	?	

Theorem: Every positive integer x can be represented as a sum or difference of at most $O(\log x / \log \log x)$ terms

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

DBNS (3/3)

Application:

$314159 = 2^4 3^9 + 2^8 3^1 - 1$
$[314159]P = [2^4 3^9]P + [2^8 3^1]P - P$

cost: 12 DBL + 10 TPL + 2 ADD

$314159 = 2^4 3^9 - 2^0 3^6 - 3^3 - 3^2 - 3 - 1$
$[314159]P = 3(3(3(3^3([2^4 3^3]P - P) - P) - P) - P$

cost: 4 DBL + 9 TPL + 5 ADD

Goal: expansions with $a_1 \geq a_2 \geq \ldots a_n \geq 0$, $b_1 \geq b_2 \geq \cdots \geq b_n \geq 0$,

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

We compute $[x]P$ in a Horner-like fashion (reuse partial results)

Representation and Coding of Numbers in Hardware (1/2)

Standard coding for one bit $b \in \{0, 1\}$:

- $1 \Longrightarrow b = 1$
- $0 \Longrightarrow b = 0$

Dual-rail coding of one bit $b \in \{0, 1\}$:

- $(r_1, r_0) = (1, 0) \Longrightarrow b = 1$
- $(r_1, r_0) = (0, 1) \Longrightarrow b = 0$

Triple-rail coding of one borrow-save digit $b \in \{-1, 0, 1\}$:

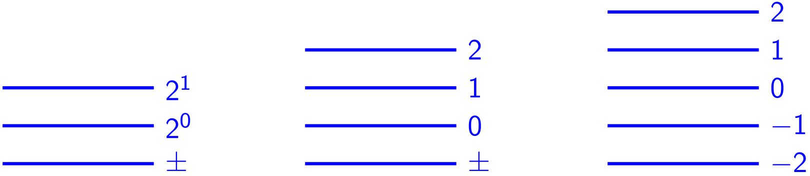
- $(r_{-1}, r_0, r_1) = (0, 0, 1) \Longrightarrow b = 1$
- $(r_{-1}, r_0, r_1) = (0, 1, 0) \Longrightarrow b = 0$
- $(r_{-1}, r_0, r_1) = (1, 0, 0) \Longrightarrow b = -1$

Goal: transitions number (then activity) is the same for all logical transitions between digits

Overhead: silicom area and local storage

Representation and Coding of Numbers in Hardware (2/2)

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



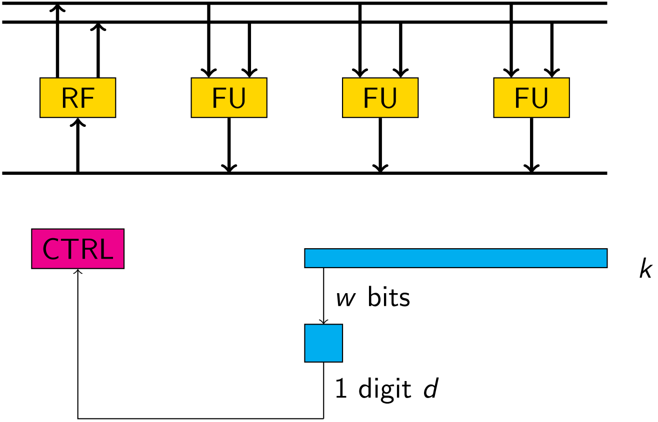
Is the overhead huge? NO!

Example: radix-4 ECC 200 unit $\Longrightarrow$ 20-30% area, -5–10% delay

Logical depth is limited to a very few gates (all nets are exclusives)

Reconfigurable Arithmetic Units for Crypto

Current work on ECC units with reconfigurable recoding of the key (k scalar for $[k]P$ scalar multiplication)



Future Prospects

- Evaluate higher radices (8, 16) and their impact on performances
- Measure resistance against SCA
- Evaluate the reconfiguration type for internal key recoding
  - frequency of the digit set modifications
  - random changes or small FSM
- Distribute VHDL sources for collaborations
- Work of arithmetic units for fault injection protection

The end, some questions ?

Contact:

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

Thank you