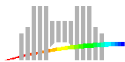


# Accélérateurs matériels reconfigurables

*RAIM 2009*

**Florent de Dinechin**

projet Arénaire, ENS-Lyon/INRIA/CNRS/Université de Lyon



# Un FPGA en 2010

Un FPGA en 2010

Digital signal processing

Cryptographie

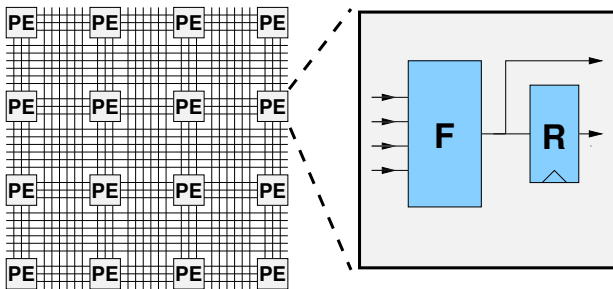
Virgule flottante sur mesure

Conclusion

# Field-programmable gate arrays

Au départ c'était cela:

## Fine-Grain Parallel Architecture



- **F**: fonction booléenne arbitraire à  $k$  entrées ( $k = 3, 4, 5, 6$ )
- **R**: un bit de mémoire
- Fonctionnalité et routage **configurables**

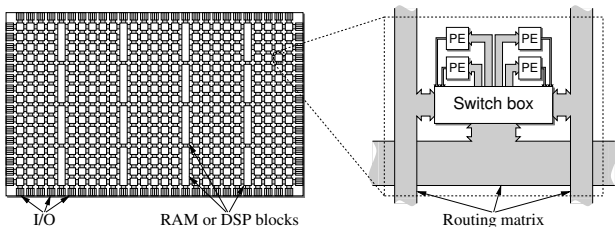
Modèle d'exécution: le **circuit**

# Historique 1990-2010

## La capacité brute suit la loi de Moore

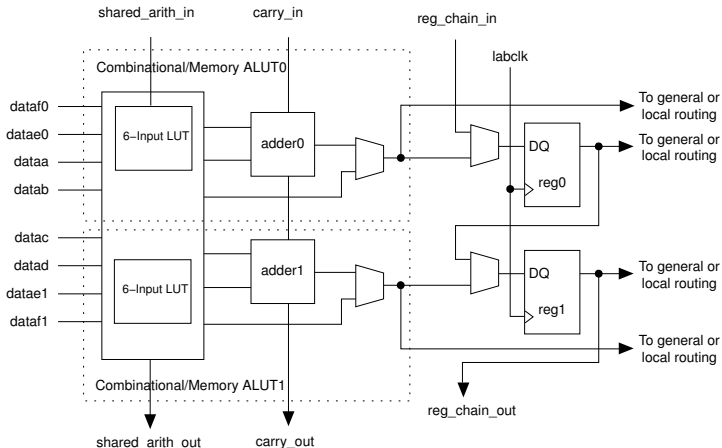
Augmentation de la granularité pour alléger le routage

- le PE
  - est passé de 4 entrées à 6 ou 7,
  - et est regroupé par 2 puis 4 puis 8 puis ...
- Apparition de blocs mémoires de quelques Kbits
- Apparition de blocs multiplieurs 18x18 bits
  - bientôt incluant un accumulateur de 48 bits (blocs DSP)
  - regroupés par 4 dans les derniers Altera



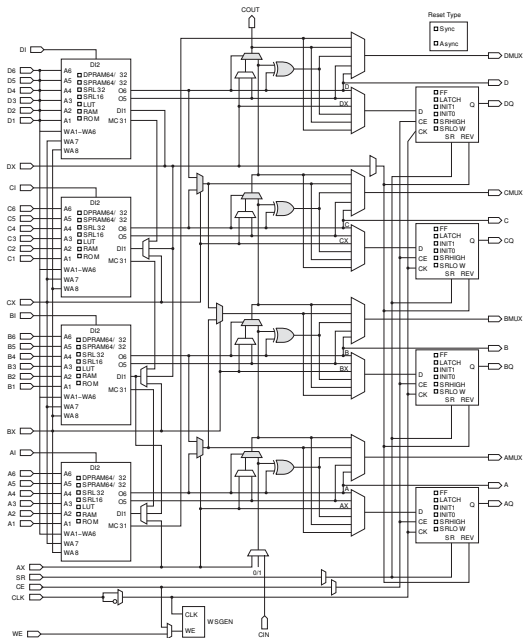
# Dans la vie, il faut savoir faire des additions

## L'ALM (Arithmetic/Logic Module) de l'Altera Stratix IV



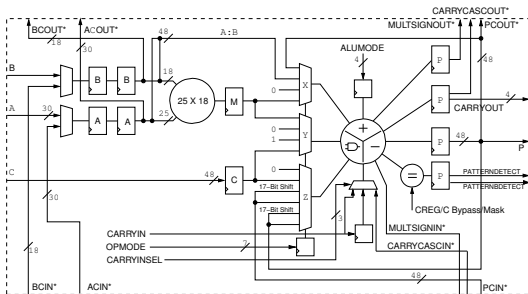
Dessin coupé de la doc. Oui, les 6-input LUT ont 4 entrées seulement. Il y a un truc.

# La Slice du Xilinx Virtex-5

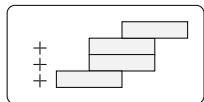


# Dans la vie, il faut savoir faire des multiplications

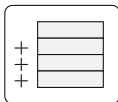
## Le bloc DSP du Virtex-5



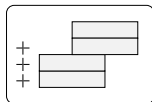
## Les configurations du bloc DSP de l'Altera Stratix IV



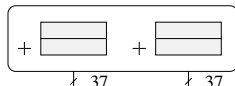
72



38



55



37

37

# Les FPGA chers en 2009

## Virtex-5 XC5VSX240T

6-input LUT + reg	149,760
DSP blocks (18x25 bits)	1056
RAM 36Kb	516

## Stratix IV EP4SGX530

4-input LUT + reg	414,960
DSP blocks (4 x 18x18 bits )	256
RAM 9Kb	1,280
RAM 144Kb	64

(je parlerai des entrées/sorties plus tard)

## Quant à programmer ces choses-là

- On ne programme pas, on **construit un circuit**
- Compilation pouvant atteindre plusieurs jours
- Debuggage horrible

### Un jour mon compilateur C viendra

- Les compilos (d'un sur-ensemble d'un sous-ensemble du C) quittent le monde académique
- ... pour arriver dans des startups
- Altera en distribue un avec ses outils.

En attendant, voyons quelques success-stories d'accélération avec de l'arithmétique dedans.

# Digital signal processing

Un FPGA en 2010

Digital signal processing

Cryptographie

Virgule flottante sur mesure

Conclusion

## Juste un exemple

Le projet TCHATER: 40Gb/s sur de “vieilles” fibres optiques

- Échantillonnage du signal optique sur 5 bit à 20 GHz
- puis environ 500–1000 opérations par échantillon

Besoin de **10–20 TOP/s**

Démonstrateur: une carte accélératrice incluant 4 FPGAs

- un pentium+SSE, en 8 bits, fait dans les 50 GOPs/cœur en crête.
- Mettez m'en donc 200.

## Avec un PC ça le ferait pas

### Heureusement

- les données sont sur 5 bits seulement
- il y a plein de multiplications par des constantes

### Exemple: le premier FPGA

- Un filtre FIR complexe de profondeur 128
- implémenté dans le domaine de Fourier
  - FFT: que des multiplications par des constantes, tabulées
  - Puis 256 multiplications complexes dans 256 blocs DSP
  - puis une FFT inverser
- Précisions intermédiaires taillées au plus juste

Mais bon, 6 mois de travail pour programmer un FIR...

# Cryptographie

Un FPGA en 2010

Digital signal processing

Cryptographie

Virgule flottante sur mesure

Conclusion

## Pourquoi c'est une bonne idée

... Parce que l'arithmétique dont on a besoin n'est pas supportée en hardware par le processeur.

## Computation of the Tate pairing

$$\hat{e} : E(\mathbb{F}_{p^m})[\ell] \times E(\mathbb{F}_{p^m})[\ell] \rightarrow \mu_\ell \subseteq \mathbb{F}_{p^{km}}^\times$$

- ▶ Arithmetic over  $\mathbb{F}_{p^m}$ :
  - polynomial basis:  $\mathbb{F}_{p^m} \cong \mathbb{F}_p[x]/(f(x))$
  - $f(x)$ , degree- $m$  polynomial irreducible over  $\mathbb{F}_p$
- ▶ Arithmetic over  $\mathbb{F}_{p^{km}}^\times$ :
  - tower-field representation
  - only arithmetic over the underlying field  $\mathbb{F}_{p^m}$
- ▶ Operations over  $\mathbb{F}_{p^m}$ :
  - $O(m)$  additions / subtractions
  - $O(m)$  multiplications
  - $O(m)$  Frobenius maps ( $a \mapsto a^p$ , i.e. squarings or cubings)
  - 1 inversion
- ▶ A first idea: an all-in-one unified operator:
  - shared resources
  - scalable architecture

## Speed-up typique: 50

Beuchat et al., *Multi-core Implementation of the Tate Pairing over Supersingular Elliptic Curves*, in Cryptology and Network Security 2009.

- Ils ont fait d'abord une version FPGA.
- Ensuite ils ont écrit une version software
  - optimisée avec amour (utilisant SSE etc)
  - au point d'être digne d'être publiée
- Dans cette publication, ils se tirent dans le pied en admettant un speed-up de 50 sur le FPGA.

On trouve des speedup de 1000 dans la littérature, mais alors la version soft n'a pas reçu la même attention que la version hard.

## Il faut aussi amener les données

- bande passante interne suffisante
- bande passante extérieure du FPGA: de l'ordre de 1Tb/s crête pour les plus gros
  - 1000 broches échangeant à 600MHz
  - une centaine capable d'échanger à 5Gb/s chacune
- cartes accélératrices : échange de données à qq Gb/s avec la mémoire/le processeur



# Virgule flottante sur mesure

Un FPGA en 2010

Digital signal processing

Cryptographie

Virgule flottante sur mesure

Conclusion

## Qui a la plus grosse?

*Peak performance (en français, intox ou pipô)*

Family Chip	Xeon 5160	Cell PS3	NVIDIA 9600 GT	AMD AMD 9270	Virtex5 LX330T
Techno	65 nm	65 nm	65 nm	55 nm	65 nm
Clock	3 GHz	3.2 GHz	1.625 GHz	750 MHz	200 MHz
DP GFlops	12	10.5	-	240	11.4
SP GFlops	24	204.8	312	1200	33
Power (W)	80	135	59–96	160–220	20–30

Nachiket Kapre, André DeHon, *Performance Comparison of Single-Precision SPICE Model-Evaluation on FPGA, GPU, Cell, and multi-core Processors*, in FPL'2009

Deux questions:

- Quid de la performance réelle pour une application donnée?
- Quid du temps de développement pour l'obtenir?

# When are FPGAs really good at floating-point ?

Basic operations:  $+$ ,  $-$ ,  $\times$  ?

- ⊖ Highly optimized FPU in the processor
- ⊖ Each operator **10x slower** in an FPGA
- ⊕ Massive parallelism on an FPGA
- FPGA matches PC, but not GPGPU, Cell, ...

Double-precision floating-point logarithm?

- Pentium core: 130 cycles @ 3GHz, not pipelined: **23 MFPLog/s**
- FPGA: one log per cycle @ 208MHz: **208 MFPLog/s**
- Chip vs chip: 4-8 Pentium cores vs 15-30 FPLog/FPGA:

**40x speed-up**

J. Detrey, F. de Dinechin, and X. Pujol. Return of the hardware floating-point elementary function. In **18th Symposium on Computer Arithmetic**. IEEE, 2007.  
(Numbers obtained using the FPLog in FloPoCo 1.15.1)

### Un peu de *green marketing* appris à FPL 2009

- Un FPGA consomme 5 fois moins qu'un Pentium
- Donc tout *speedup* pas terrible peut être transformé en un *energy efficiency improvement* qui est 5 fois meilleur
- Exemple: *Computing double-precision logarithms in FPGAs is 200x more energy-efficient on an FPGA.*

Moi je préfère rouler à vélo.

# The FloPoCo project: Not your neighbour's FPU

## Useful operators that would not be economical in a processor

- Elementary functions (sine, exponential, logarithm...)
- Algebraic functions ( $\frac{x}{\sqrt{x^2 + y^2}}$ , polynomials, ...)
- Compound functions ( $\log_2(1 \pm 2^x)$ ,  $e^{-Kt^2}$ , ...)
- Floating-point sums, dot products, sums of squares
- Specialized operators: constant multipliers, squarers, ...
- Complex arithmetic
- LNS arithmetic
- Decimal arithmetic
- Interval arithmetic
- ...
- Oh yes, basic operations, too.

# Optimization opportunities?

## A floating-point operator for

$$x^2 + y^2 + z^2$$

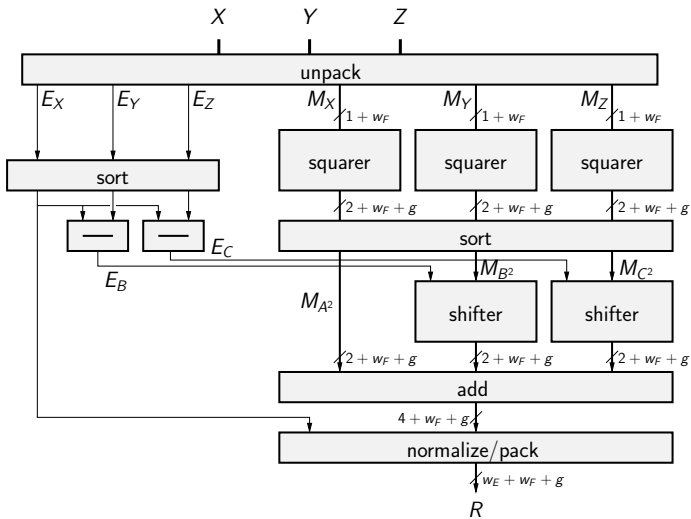
(not a toy example but a useful building block)

- A square is simpler than a multiplication
- $x^2$ ,  $y^2$ , and  $z^2$  are positive:
  - one half of your FP adder is useless
- Accuracy can be improved:
  - 5 rounding errors in the floating-point version
  - $(x^2 + y^2) + z^2$  : asymmetrical

## The FloPoCo Recipe

- build a fixed-point architecture (FPGAs are good at fixed point)
- but keep the FP interface

# Optimization opportunities (2)



## Optimization opportunities (3)

A few results for floating-point sum-of-squares on Virtex4:

Simple Precision	area	performance
LogiCore classic	1282 slices, 20 DSP	43 cycles @ 353 MHz
FloPoCo classic	1188 slices, 12 DSP	29 cycles @ 289 MHz
FloPoCo custom	453 slices, 9 DSP	11 cycles @ 368 MHz

Double Precision	area	performance
FloPoCo classic	4480 slices, 27 DSP	46 cycles @ 276 MHz
FloPoCo custom	1845 slices, 18 DSP	16 cycles @ 362 MHz

- all performance metrics improved, FLOP/s/area more than doubled
- Plus: custom operator more accurate, and symmetrical

## Making development easier

- An arithmetic operation is a **function** (in the mathematical sense)
  - few well-typed inputs and outputs
  - no memory or side effect
- An operator is **the implementation** of such a function
  - IEEE-754 FP standard:  $\text{operator}(x) = \text{rounding}(\text{operation}(x))$

### Operators are a simple class of circuit

- direct acyclic graphs
  - may be pipelined without control or state machine
- easy to test
  - stateless
  - expected result is mathematically-defined

FloPoCo is also a **framework** that builds on this simplicity.

# Le bonheur du chercheur en arithmétique

- Une terre vierge
  - Ces opérateurs n'ayant aucun sens dans un processeur, ils n'ont pas été étudiés
- qui s'étend à perte de vue
  - Un papier sur la multiplication par 17, puis un sur la multiplication par 42, puis...
- et en plus en prétendant être vraiment utile
  - j'ignore si l'accélération de virgule flottante sur FPGA a de l'avenir
  - mais si elle en a, c'est en faisant comme nous

## Une approche alternative

### Floating-Point Datapath Compiler (Langhammer, Altera)

- Entrée: un morceau de C *straight line*
- Sortie: un pipeline d'opérateurs

### Presque la même recette que nous

- Étendre les multiplications de mantisse 24x24 à 36x36
  - 4 DSP 18x18
- Normaliser les additions moins souvent en ajoutant des “bits d’overflow”
- Économiser
  - les arrondis intermédiaires
  - les normalisations suivies de dénormalisations
  - le traitement des cas spéciaux

Bien sûr, on s’assied sur les standards (C99 et IEEE-754).

# Conclusion

Un FPGA en 2010

Digital signal processing

Cryptographie

Virgule flottante sur mesure

Conclusion

## La malédiction du FPGA

- Potentiel indéniable pour des calculs qu'un processeur fait mal
- Dès que ces calculs seront vraiment utiles, l'extension SSE n+1 les supportera
- Les FPGA condamnés à vivre dans des niches?

## Deux vrais challenges

- Rendre les FPGA sympas à programmer
- Faire payer leur R&D par les *gamerz* comme les GPU

## Soyons optimistes

Mercredi dernier j'ai reçu cela:

(...) We are a startup company focussed on supercomputing with FPGAs. (...) Unlike any other FPGA company I've ever known, we aren't burning through some investors' money. Every time we hire someone it's because a client gave us money for an FPGA accelerated application. In short, I think we're it, the FPGA computing company that will finally do something rather than talk about something.

# Thank you for your attention



F. de Dinechin, C. Klein and B. Pasca, **Generating high-performance custom floating-point pipelines.** In *Field Programmable Logic and Applications*. IEEE, 2009.



F. de Dinechin and B. Pasca, **Large multipliers with fewer DSP blocks.** In *Field Programmable Logic and Applications*. IEEE, 2009.



F. de Dinechin, B. Pasca, O. Creț, and R. Tudoran, **An FPGA-specific approach to floating-point accumulation and sum-of-products.** In *Field-Programmable Technologies*. IEEE, 2008, pp. 33–40.



N. Brisebarre, F. de Dinechin, and J.-M. Muller, **Integer and floating-point constant multipliers for FPGAs.** In *Application-specific Systems, Architectures and Processors*. IEEE, 2008, pp. 239–244.



J. Detrey and F. de Dinechin. **Floating-point trigonometric functions for FPGAs.** In *Intl Conference on Field-Programmable Logic and Applications*, pages 29–34. IEEE, Aug. 2007.



J. Detrey and F. de Dinechin. **Parameterized floating-point logarithm and exponential functions for FPGAs.** *Microprocessors and Microsystems*, 31(8):537–545, Jun. 2007. Elsevier.



J. Detrey, F. de Dinechin, and X. Pujol. **Return of the hardware floating-point elementary function.** In *18th Symposium on Computer Arithmetic*, pages 161–168. IEEE, June 2007.



J. Detrey and F. de Dinechin, **Table-based polynomials for fast hardware function evaluation.** In *Application-specific Systems, Architectures and Processors*. IEEE, 2005, pp. 328–333.

# Accélération de fouille de données

Pompé sur le site d'XtremeData, qui vend un calculateur spécialisé (dbX)

- Built for high performance ad hoc "Database Analytics"
  - Operational Reporting
  - Ad-Hoc Data Exploration
- Discriminators versus competition
  - Best performance per dollar
  - Best performance per watt
  - Best performance per square foot
- Business benefits
  - Deeper wider data exploration – 10x data sizes
  - Cycle time compression – 1/10th of the time
  - Increase analyst productivity

## Un peu de technique

- Un FPGA prend la place d'un processeur sur une carte bipro
  - 10x performance, disent-ils
  - 1/3 de la conso (mais alors, pourquoi a-t-il un radiateur deux fois plus gros?)
- Compilation presse-bouton possible, car domaine d'application restreint
- Opérations de base souvent très simples (trop pour le processeur)

Combien vont-ils en vendre?