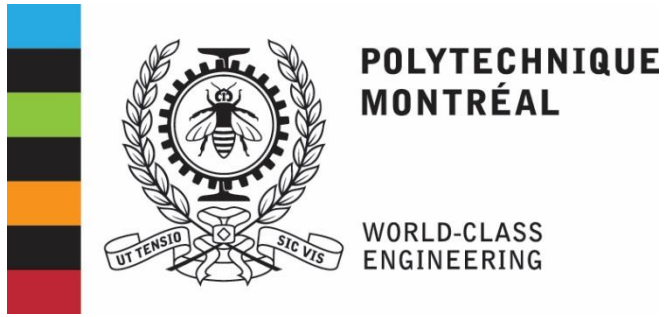
Application-Specific Instruction Set Processors
What, why, when and how?



Pierre Langlois, Eng., Ph.D.
Professor and Chair
Department of Computer and Software Engineering
Polytechnique Montréal

CMC Microsystems Seminar – 26 March 2018
Configure Your Research Platform: Infrastructure Needs for Embedded and Heterogeneous Computing

About me

- Naval engineering officer 1990-2005
- Ph.D. in Computer Engineering from Royal Military College in 2002
  - Received several fabrication grants from the *then* Canadian Microelectronics Corporation
  - Spent many hours with Bob Stevenson in CMC offices in Queen's testing my chips
- Prof. in RMC M&CS and ECE Departments 1999-2005
- Prof. in Polytechnique Montréal since 2005
- Chair of C&SE Department since 2016

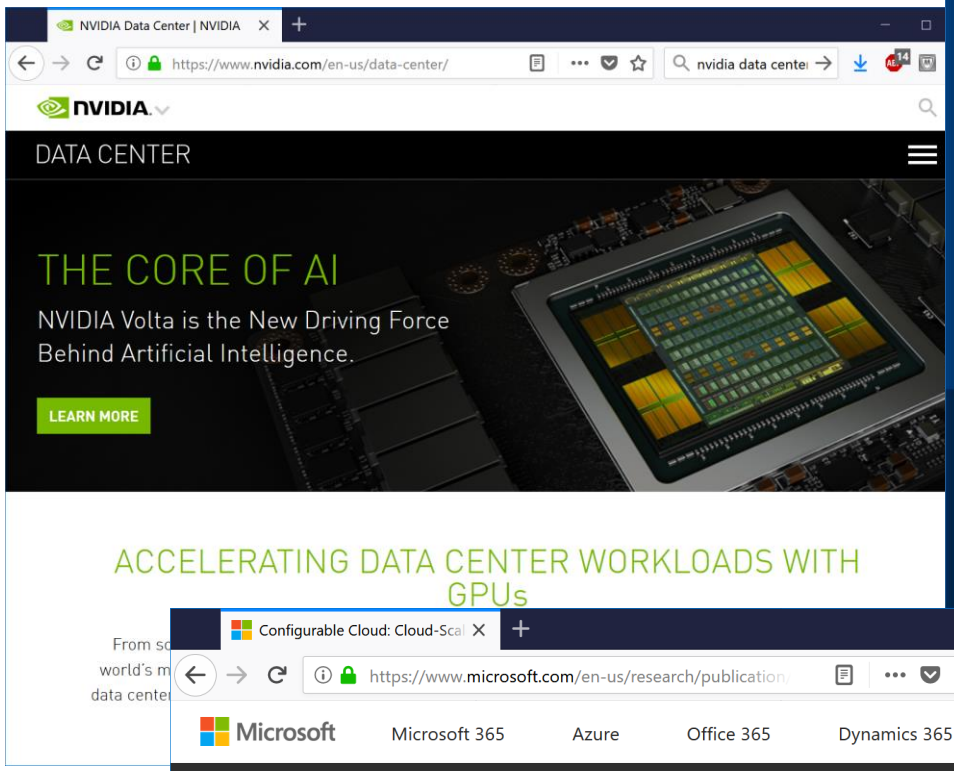
- Research on computer architecture, mostly on the implementation of signal, image and other processing
- Co-author or author of ~95 refereed papers
- Principal or co-advisor of ~35 research and ~40 non-research Highly Qualified Personnel (since 2002)

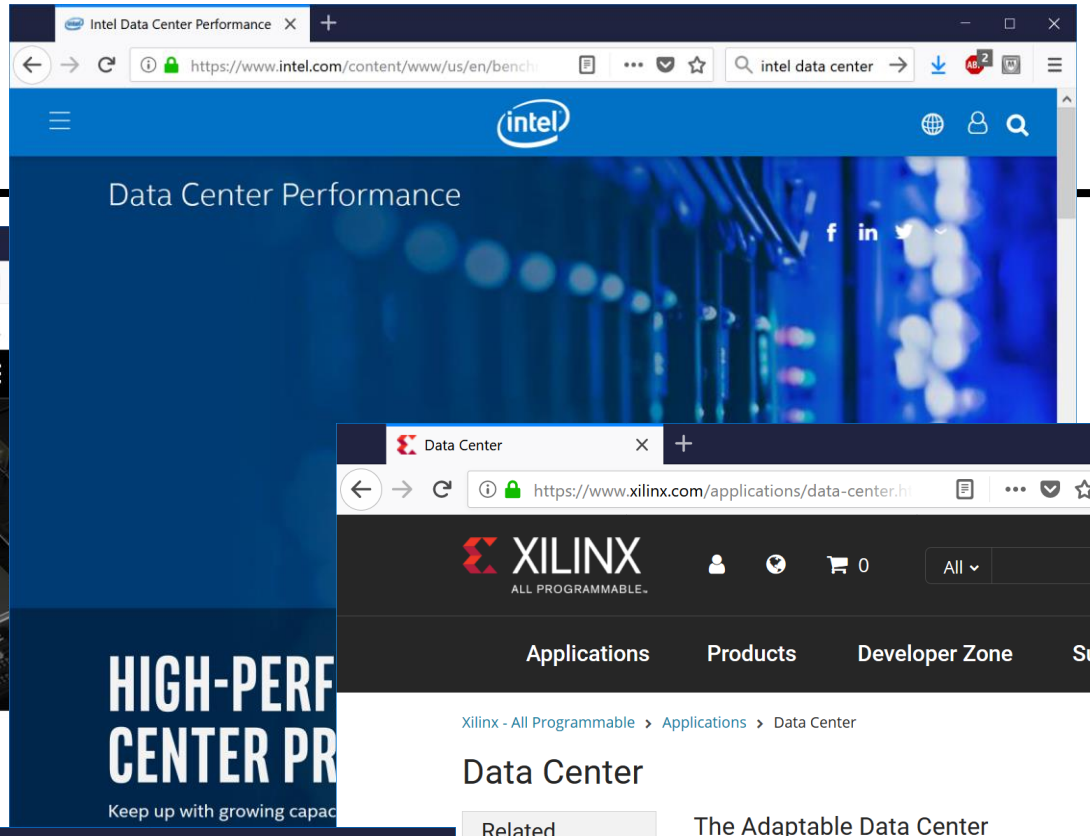
- Graduate course INF8505 : *Processeurs embarqués configurables* since 2008

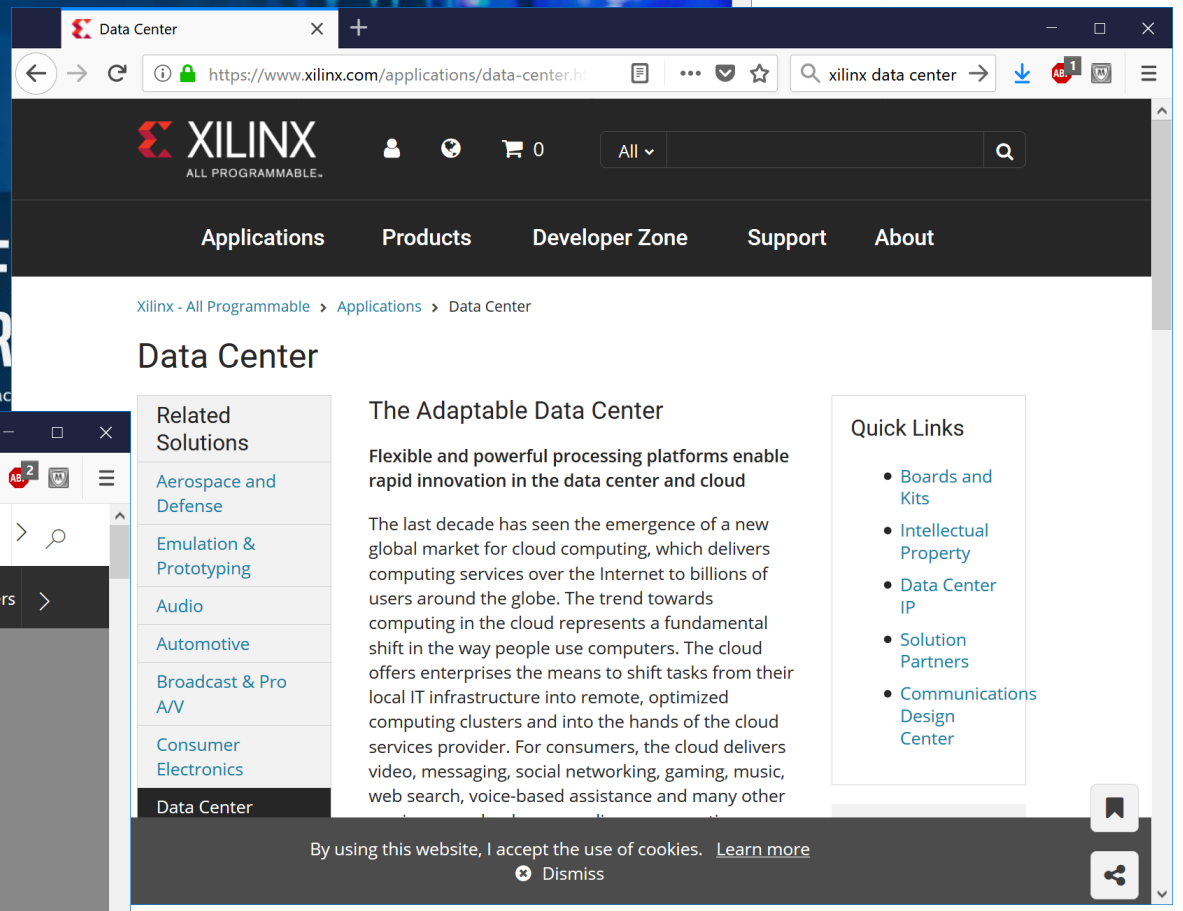
Outline

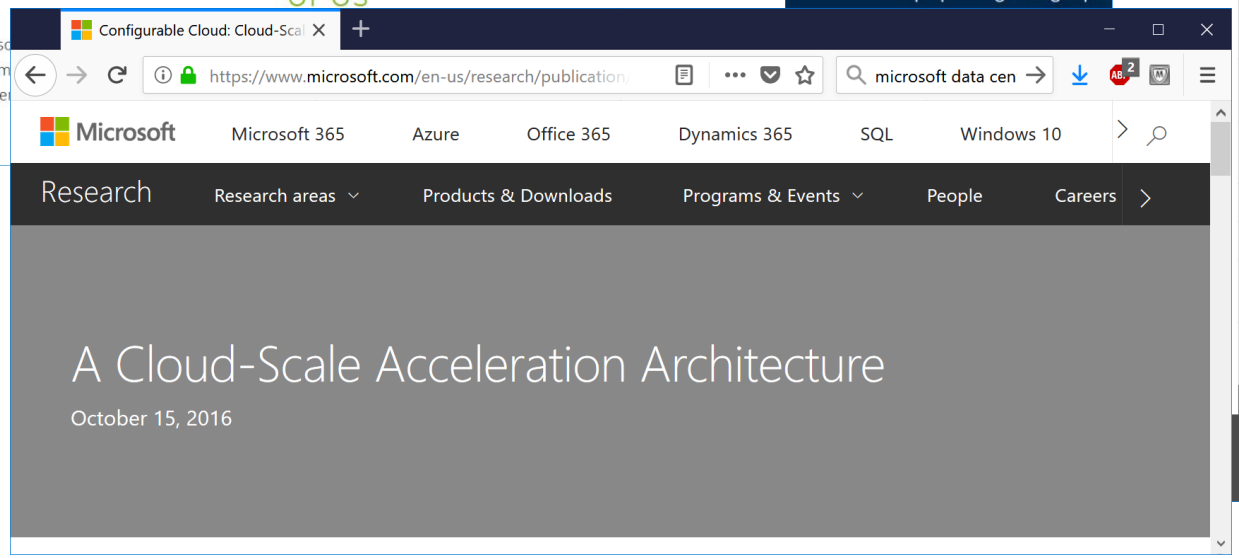
- The war is on : ACAP, ASIC, ASIP, CGRA, CPU, FPGA, GPU, SIMD, VLIW
- A few details about Application-Specific Instruction-set Processors (ASIPs)
  - Architecture
  - Instruction set
  - Memory hierarchy
  - Design tools
- Tools, tools and tools
- A few recent research results

The war is on


NVIDIA Data Center | NVIDIA
https://www.nvidia.com/en-us/data-center/
nvidia data center
NVIDIA
DATA CENTER
THE CORE OF AI
NVIDIA Volta is the New Driving Force Behind Artificial Intelligence.
LEARN MORE
ACCELERATING DATA CENTER WORKLOADS WITH GPUs

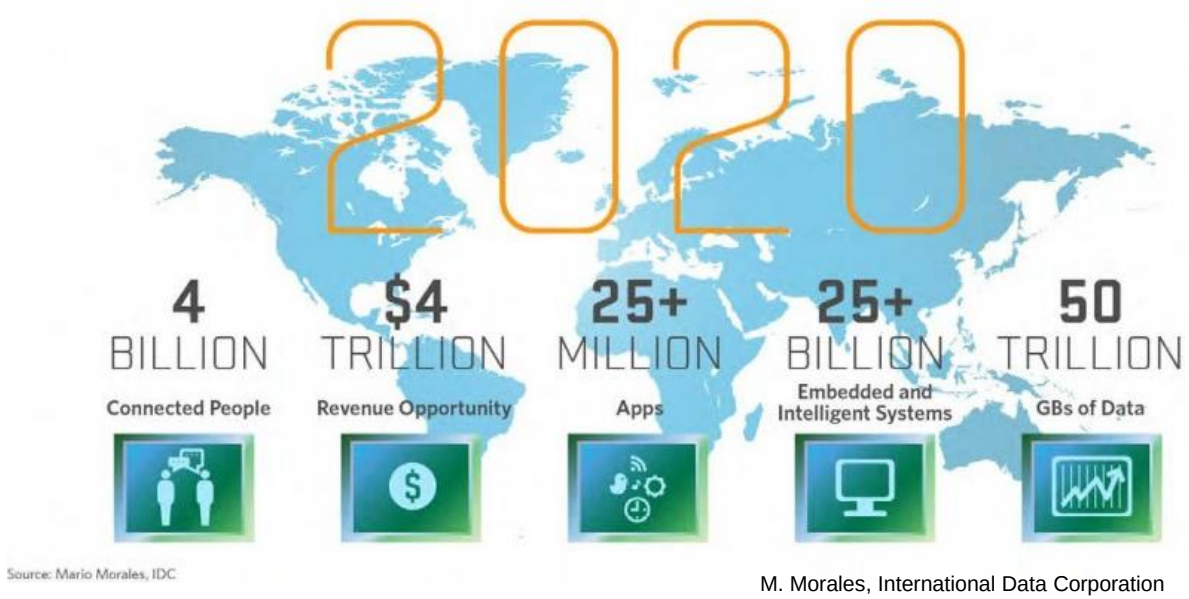

Intel Data Center Performance
https://www.intel.com/content/www/us/en/bench
intel data center
intel
Data Center Performance
HIGH-PERF
CENTER PR
Keep up with growing capac


Data Center
https://www.xilinx.com/applications/data-center.h
xilinx data center
XILINX
ALL PROGRAMMABLE
0
All
Applications
Products
Developer Zone
Support
About
Xilinx - All Programmable › Applications › Data Center
Data Center
Related Solutions
Aerospace and Defense
Emulation & Prototyping
Audio
Automotive
Broadcast & Pro A/V
Consumer Electronics
Data Center
The Adaptable Data Center
Flexible and powerful processing platforms enable rapid innovation in the data center and cloud
The last decade has seen the emergence of a new global market for cloud computing, which delivers computing services over the Internet to billions of users around the globe. The trend towards computing in the cloud represents a fundamental shift in the way people use computers. The cloud offers enterprises the means to shift tasks from their local IT infrastructure into remote, optimized computing clusters and into the hands of the cloud services provider. For consumers, the cloud delivers video, messaging, social networking, gaming, music, web search, voice-based assistance and many other
Quick Links
Boards and Kits
Intellectual Property
Data Center IP
Solution Partners
Communications Design Center
By using this website, I accept the use of cookies. Learn more
Dismiss


Configurable Cloud: Cloud-Scal
https://www.microsoft.com/en-us/research/publication
microsoft data cen
Microsoft
Microsoft 365
Azure
Office 365
Dynamics 365
SQL
Windows 10
Research
Research areas
Products & Downloads
Programs & Events
People
Careers
A Cloud-Scale Acceleration Architecture
October 15, 2016

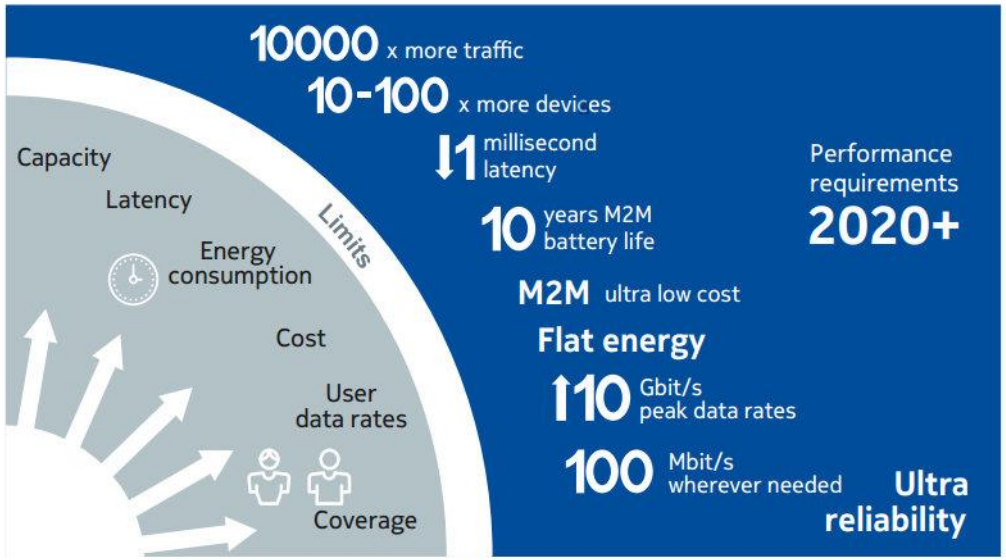
Quelques défis de l’internet des objets et des réseaux 5G

- Les chiffres de l’internet des objets pour 2020:
  - 4 G, 4 T, 25 M, 25 G et 50 T



M. Morales, International Data Corporation

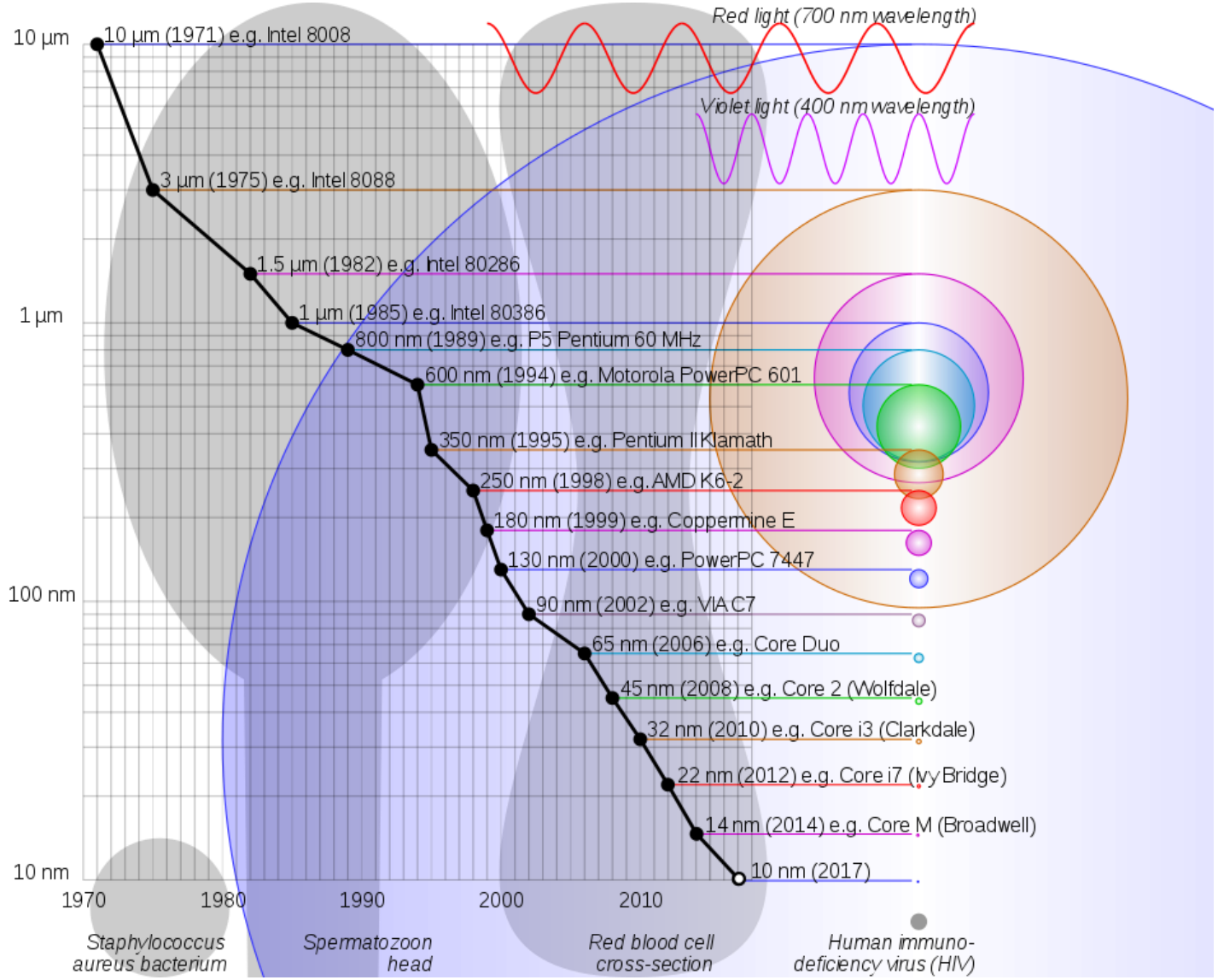
- Les chiffres pour les réseaux 5G en 2020+:
  - 100×, 1 ms, 10 ans, 10 G / 100 M



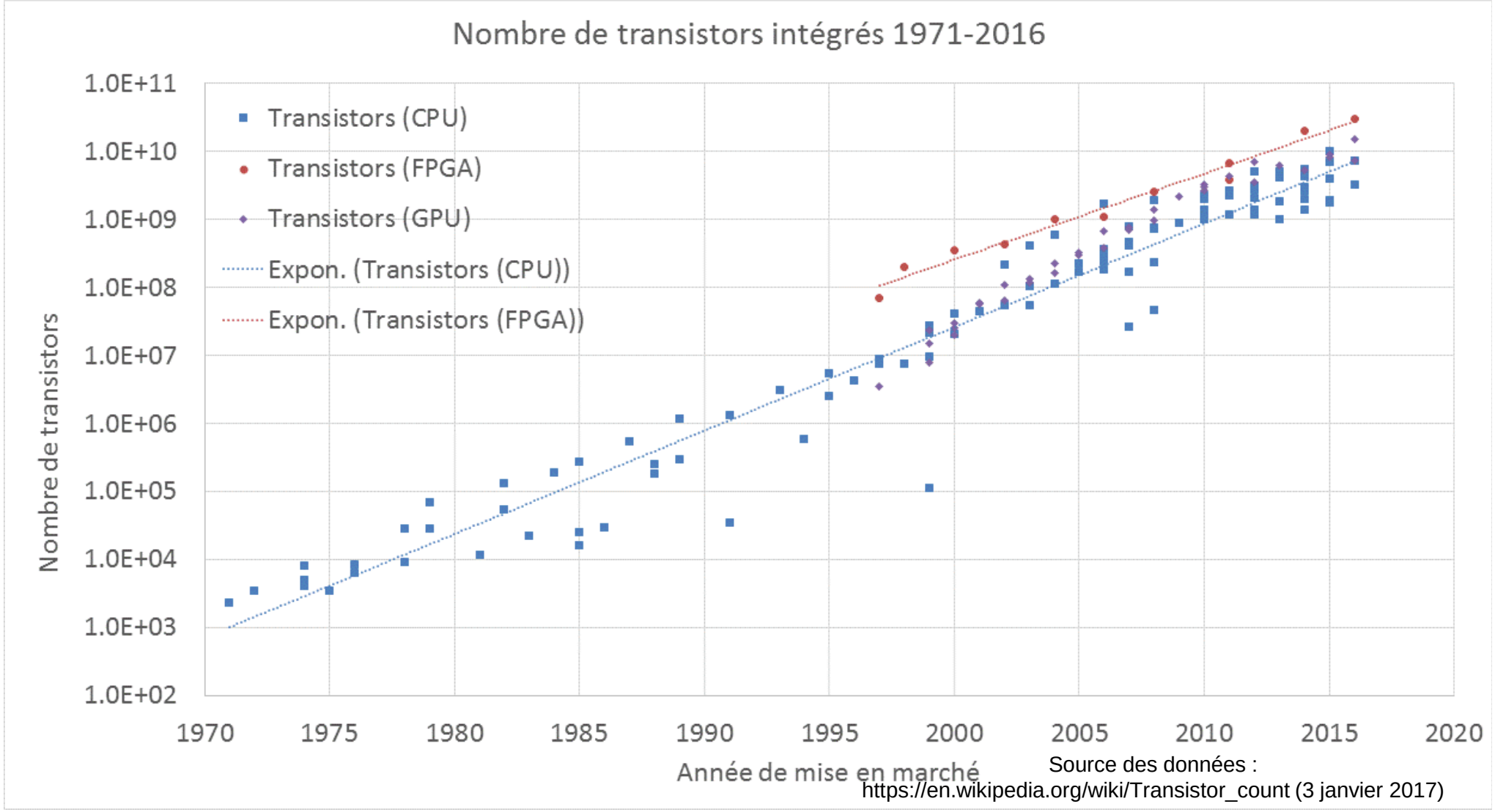
“What is 5G? 5G vs 4G & 3G”, TechnoDots, Mars 2016

Résolution du processus de fabrication

cheveu humain:
~70 µm

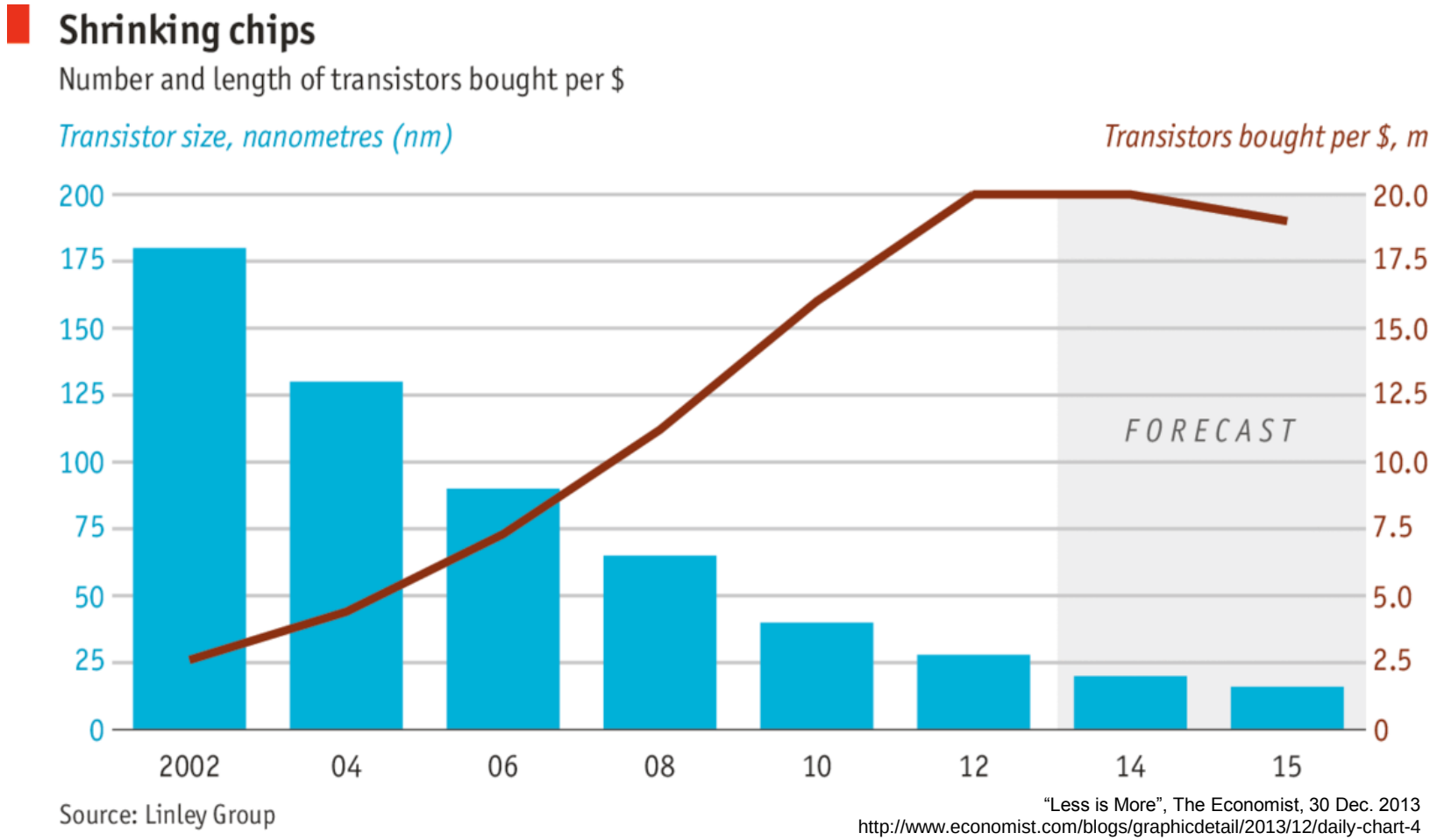


Nombre de transistors par puce et loi de Moore



Source des données :
https://en.wikipedia.org/wiki/Transistor_count (3 janvier 2017)

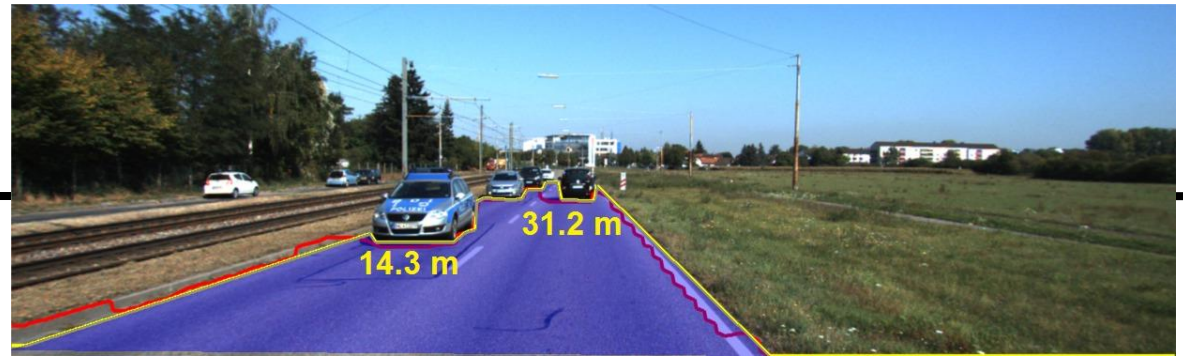
Coût par transistor



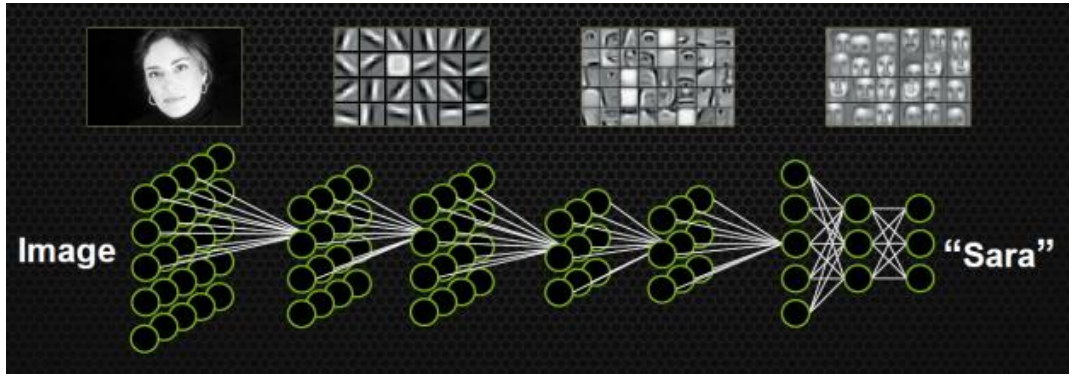
"Less is More", The Economist, 30 Dec. 2013
http://www.economist.com/blogs/graphicdetail/2013/12/daily-chart-4

Target applications
AI, Machine/Deep Learning, graphics

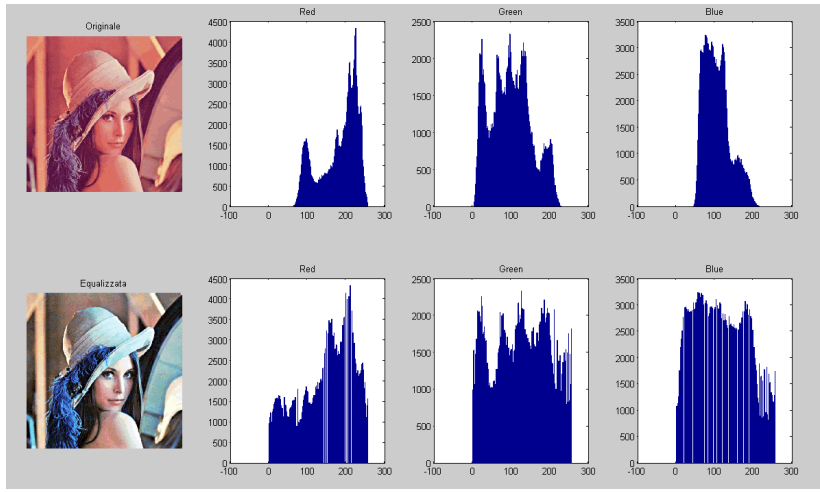
- High parallelism
- High throughput and low latency
- Simple operations
- Custom and irregular data types
- Von Neumann architectures seem like overkill
- Power, power, power
- Must be able to quickly adapt



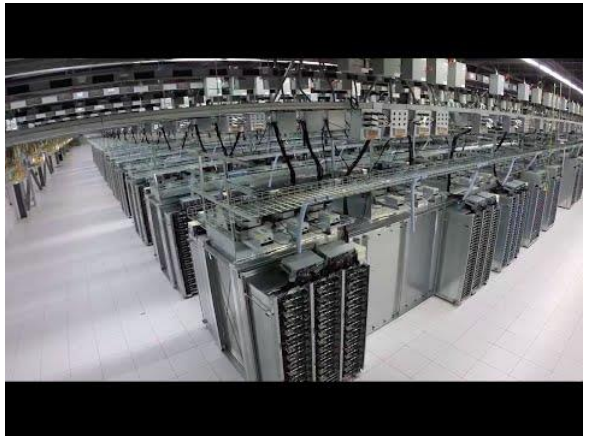
J. Yao et al., Estimating Drivable Collision-Free Space from Monocular Video, WACV 2015.



L. Brown, Accelerate Machine Learning with the cuDNN Deep Neural Network Library, NVIDIA, 2014.



Mathworks.com

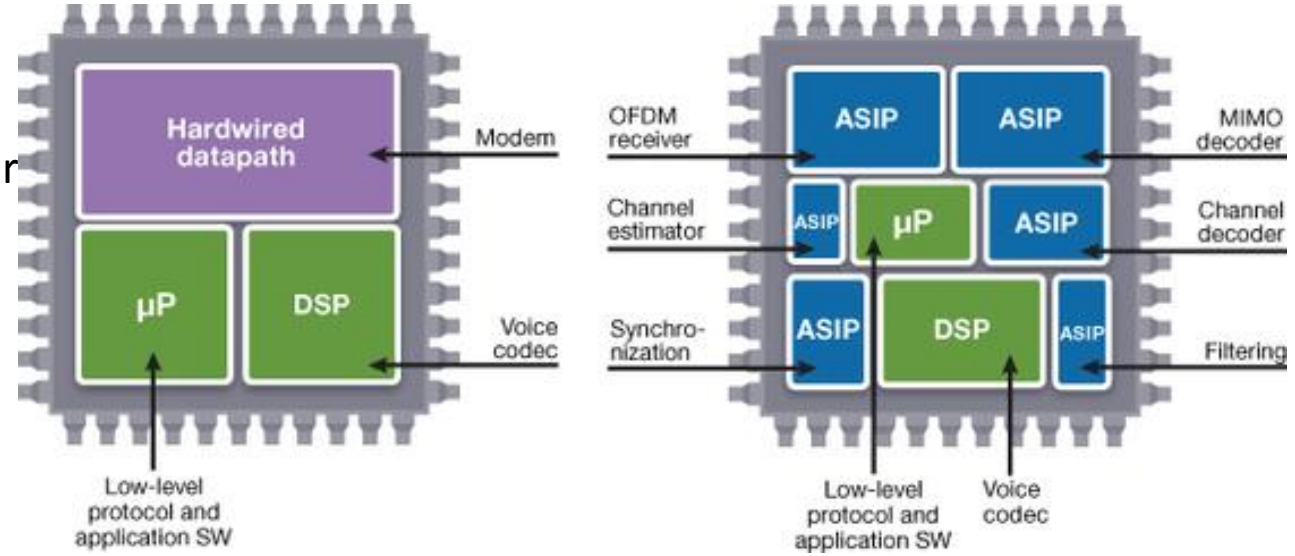


https://www.youtube.com/watch?v=XZmGGAbHqa0

Évolution des systèmes embarqués avec unités de calcul hétérogènes

- Garder un CPU pour le contrôle et la cohésion entre les blocs spécialisés
- Exploiter le parallélisme de façon explicite avec des accélérateurs matériels et coprocesseurs
  - Un CPU est souvent plus complexe que nécessaire pour une tâche spécifique
  - La question : quelle sorte de processeur choisir pour chaque unité de traitement?
    - Processeurs DSP, GPU, processeurs sur mesure
    - Présence d'opérations vectorisées, pipelines contrôlables, blocs de registres spécialisés, etc.
- Accès aux données
  - Largeur de bande suffisante
  - Énergie pour les transferts de données

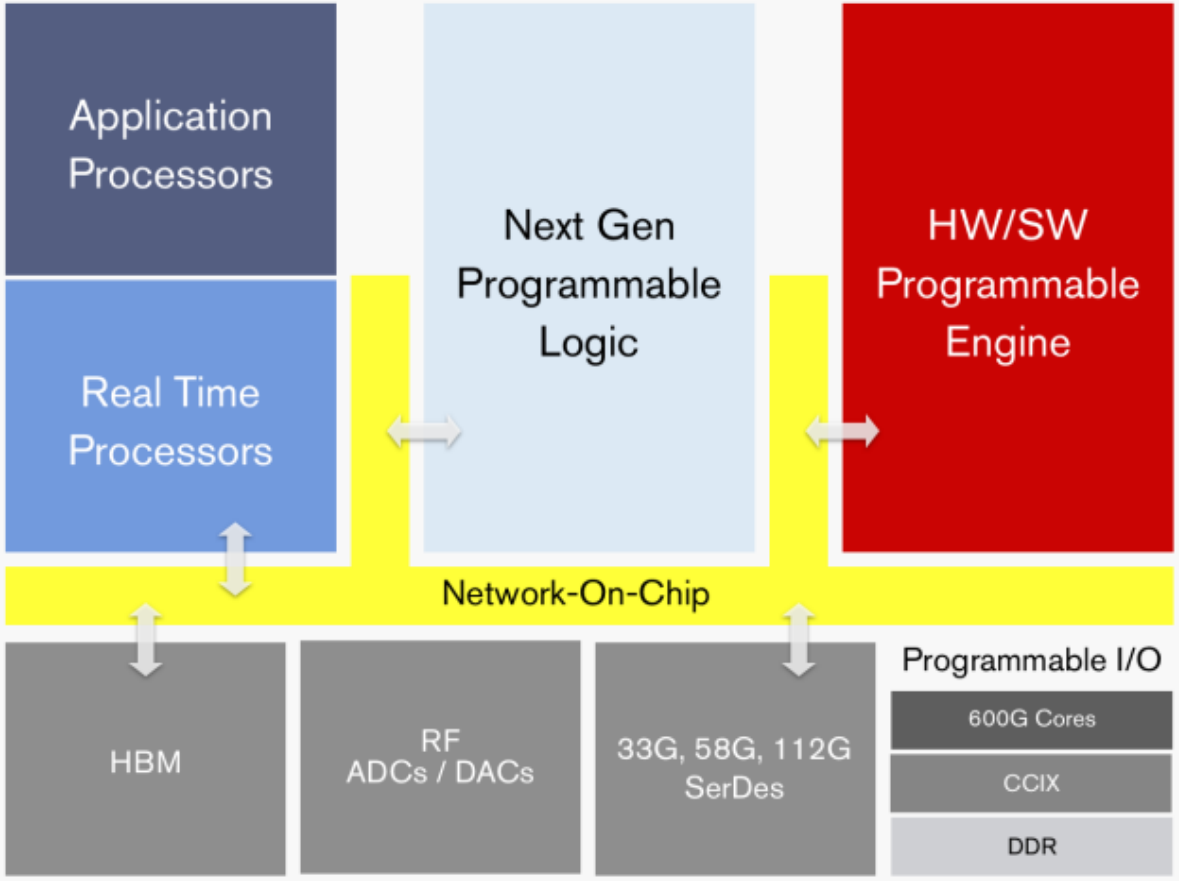
Une proposition : utiliser des processeurs configurables (ASIP) pour des unités de traitement à fonction unique.



M. Willems, Multicore Design Using ASIPs : Blending Performance and Efficiency with Programmability, Synopsys. https://www.synopsys.com/designware-ip/newsletters/technical-bulletin/multicore-design.html

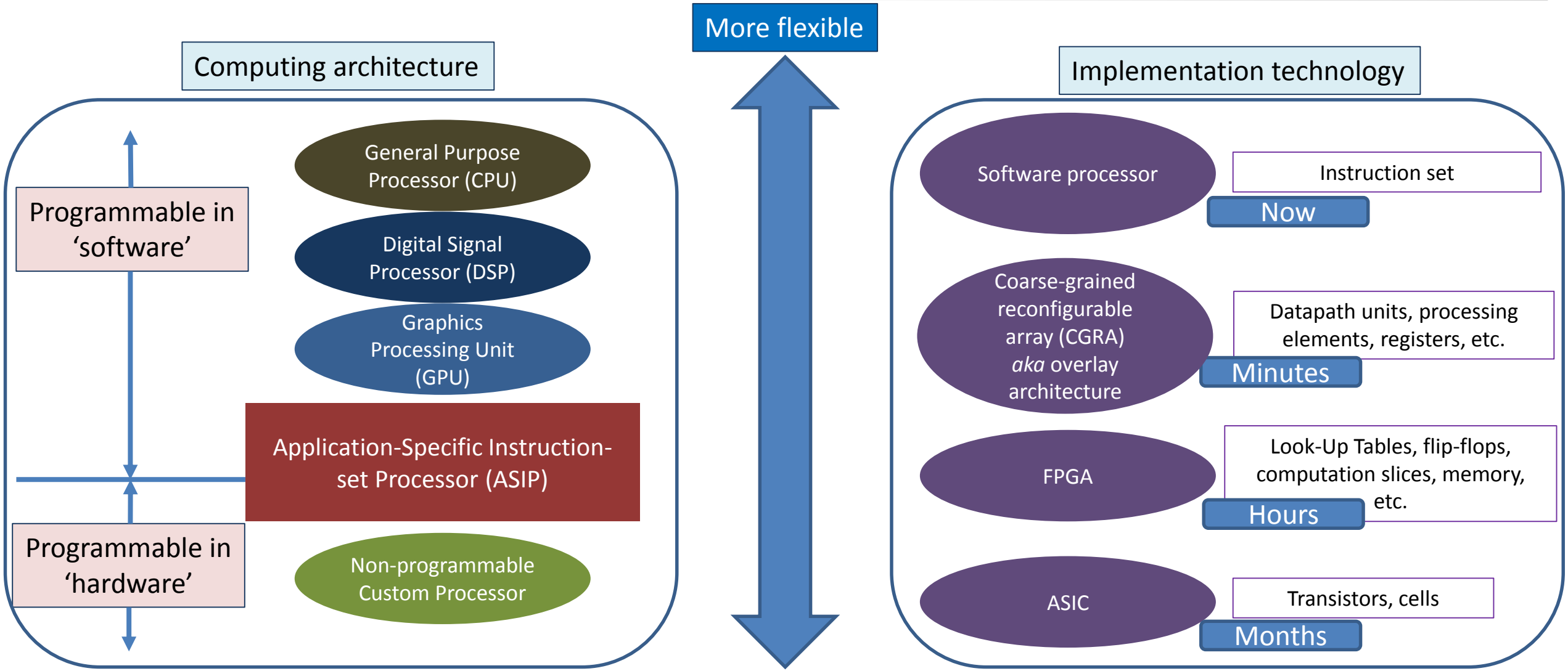
Xilinx Adaptive Compute Acceleration Platform (ACAP)

- "An ACAP is a highly integrated multi-core heterogeneous compute platform that can be changed at the hardware level to adapt to the needs of a wide range of applications and workloads."
  - Xilinx Press Release, 19 March 2018
- "… I talked about [a] hardware software programmable engine, which won't be an engine that is customizable down to a single bit, but it will have notions of some granular data paths and memory and things like that, and it has some overlay of an instruction set, but while most people won't program it, it still has hardware programmability. …"
  - Victor Peng, CEO, Xilinx, in interview with I. Cutress, 19 March 2018.

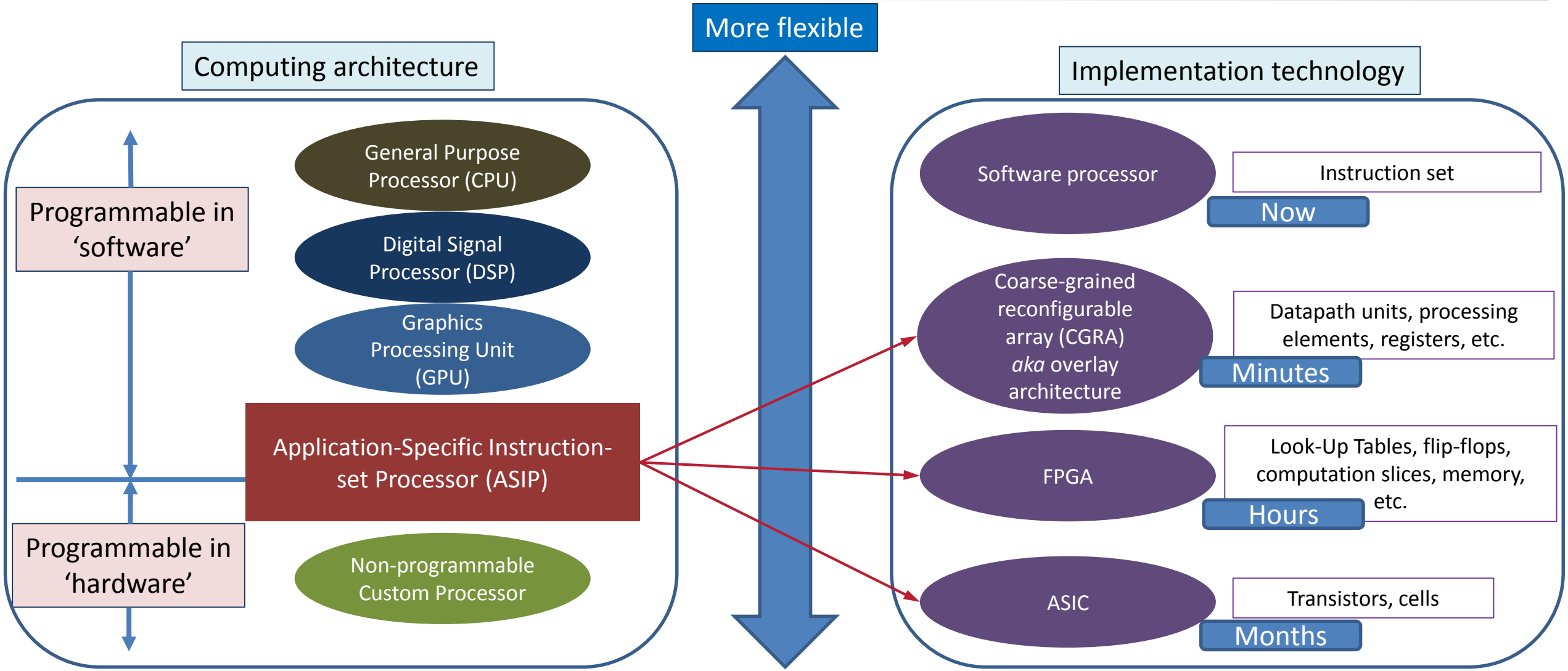


I. Cutress, AnandTech, 19 March 2018.
Online : https://www.anandtech.com/show/12509/xilinx-announces-project-everest-fpga-soc-hybrid

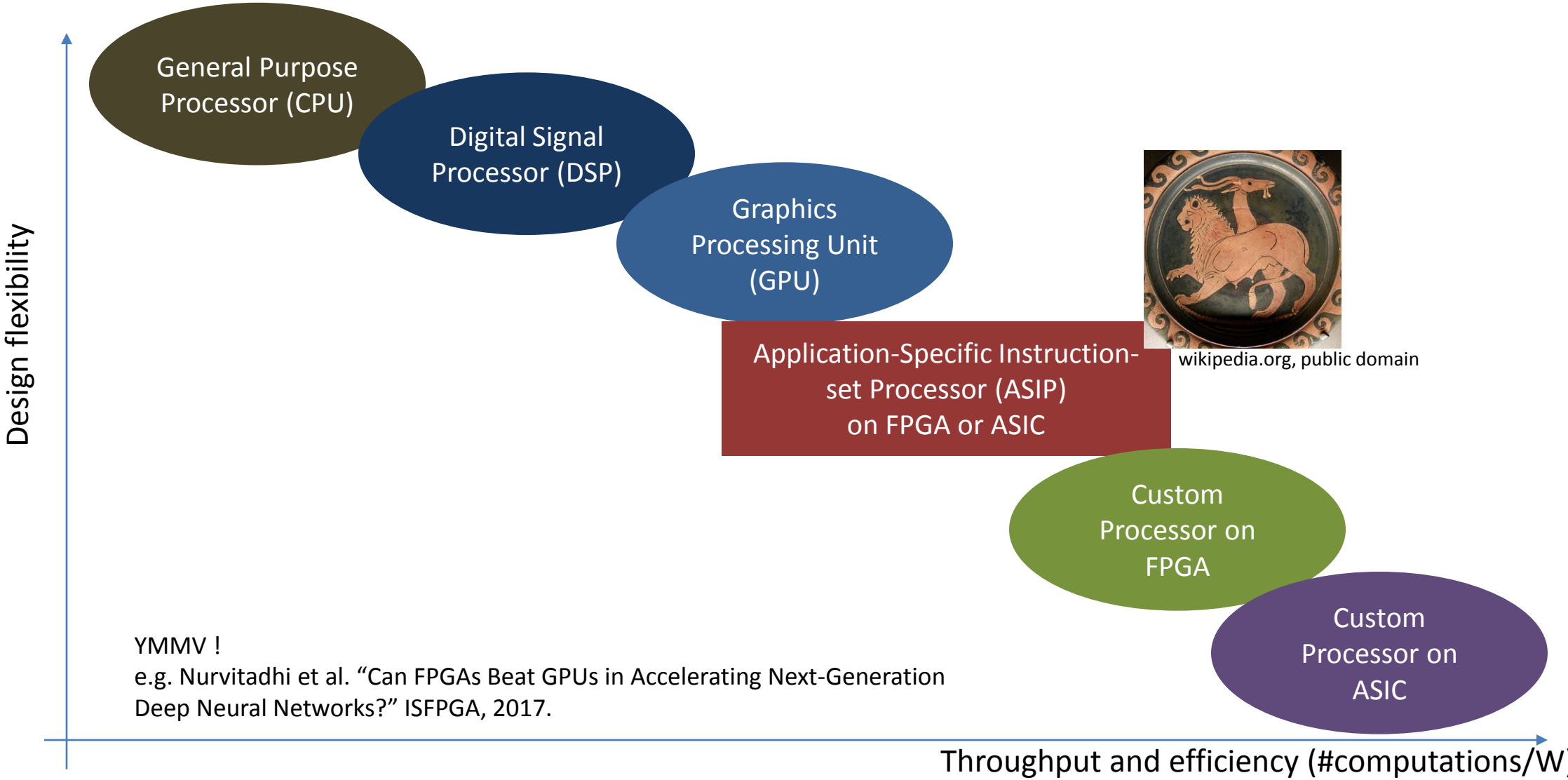
ASIPs in the design space



ASIPs in the design space



ASIPs in the design space



wikipedia.org, public domain

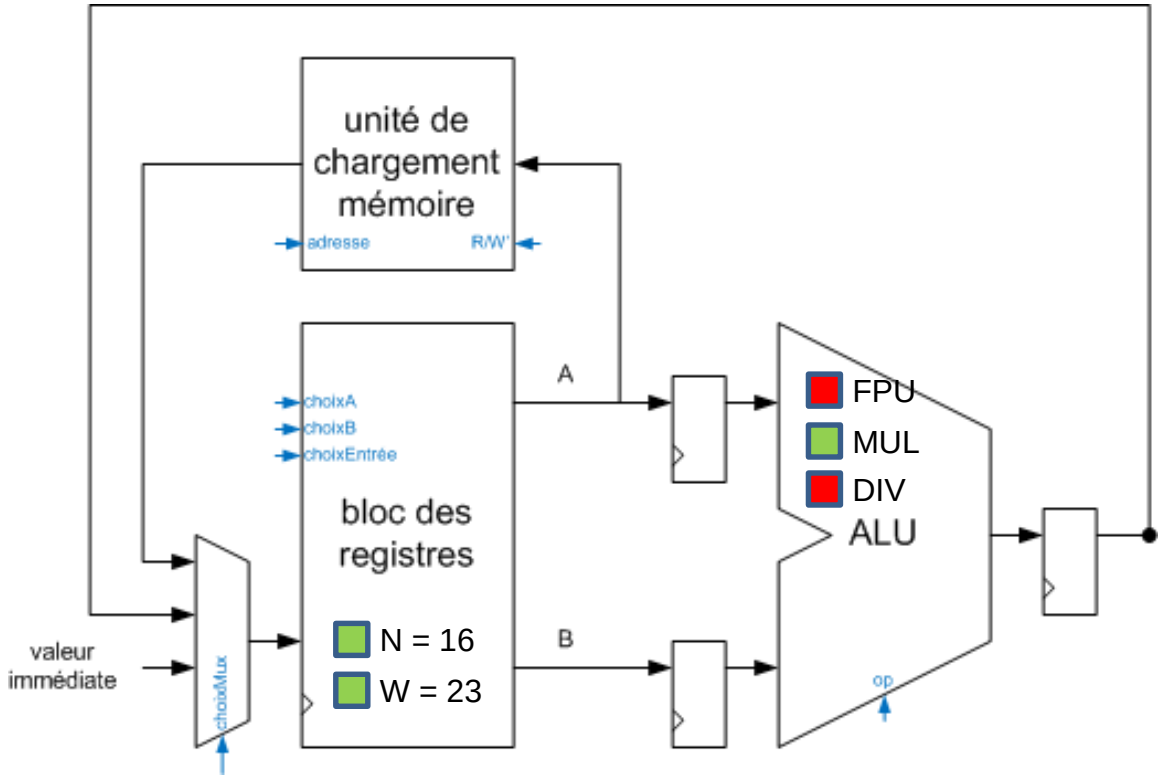
YMMV !
e.g. Nurvitadhi et al. “Can FPGAs Beat GPUs in Accelerating Next-Generation Deep Neural Networks?” ISFPGA, 2017.

Application-Specific Instruction-set Processors (ASIP)
The basics

- Processors that share a common architectural squeleton
- Configurable
  - Some features can be activated (or not) e.g. :
    - Multiplier(s)
    - Floating point unit(s)
    - caches, etc.
  - Some features can be tailored (size, number, etc.) e.g. :
    - Data path width
    - Number of pipelines
    - Number and width of interface buses
    - Register file size, etc.
- Extensible
  - Custom instructions
  - Additional register file(s)
  - Special memory interfaces
  - etc.
- Need for Development tools
  - Synthesis from a special language (HDL-like, or architecture description language)
  - Retargetable compiler
  - Cycle-accurate functional simulator
  - Etc.

ASIP feature #1 : configure existing processor parameters

- Paramétrage
  - Certaines caractéristiques peuvent être activées ou non, e.g. :
    - Multiplieurs
    - Diviseurs
    - Unités à virgule flottante
    - Décalage
    - Caches
  - Certaines caractéristiques peuvent être élargies ou multipliées, e.g. :
    - Largeur du chemin de données;
    - Nombre de pipelines;
    - Nombre et largeur des bus d'interface;
    - Taille et nombre de registres
    - Taille des caches



Effectivement, le paramétrage est rendu possible parce que les concepteurs du processeur de base ont prévu d'avance des options que les utilisateurs de ASIP pourraient désirer.

Ces options peuvent 'facilement' être ajoutées ou non dans le processeur de base.

ASIP feature #2 : add custom instructions

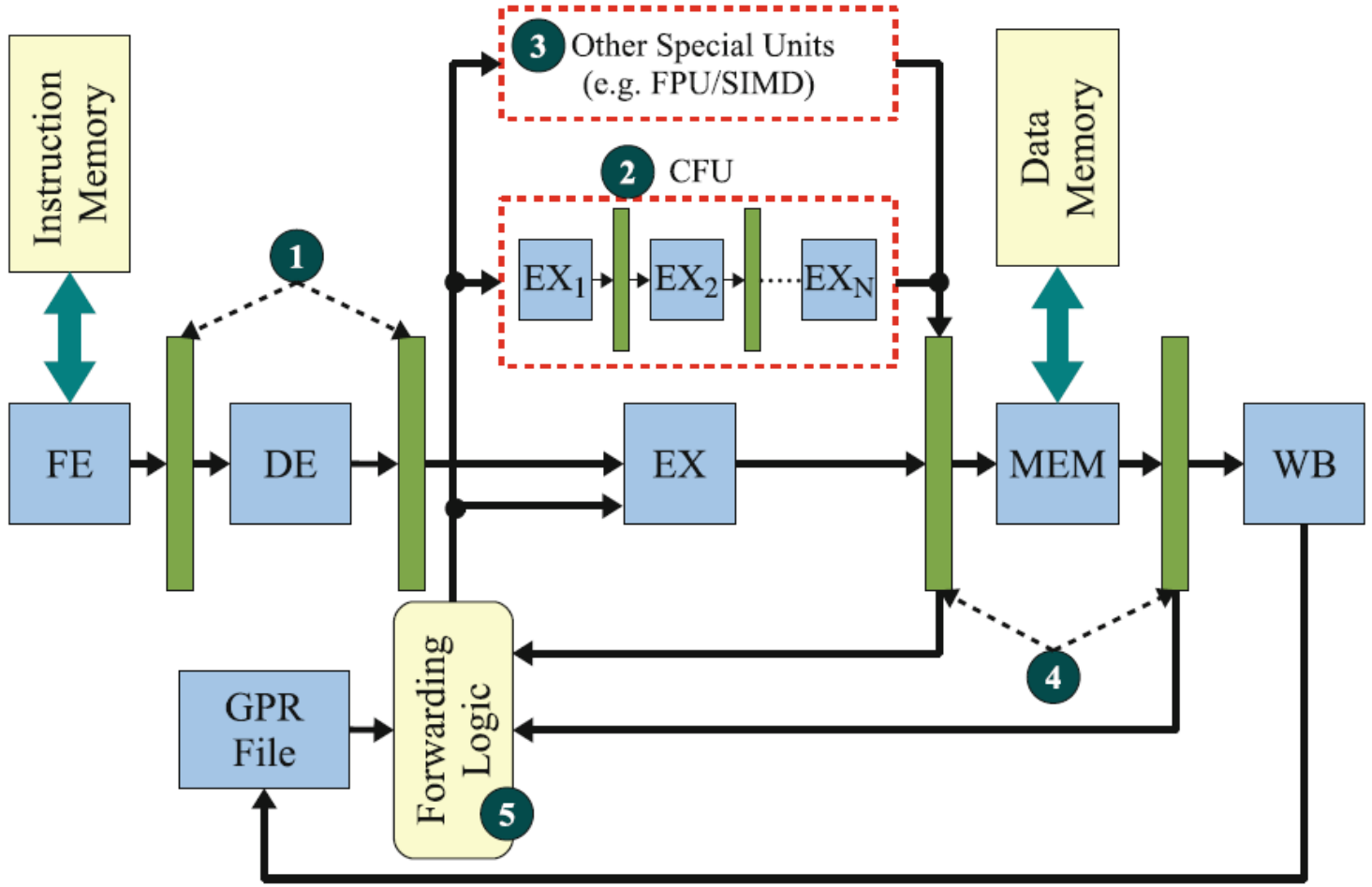
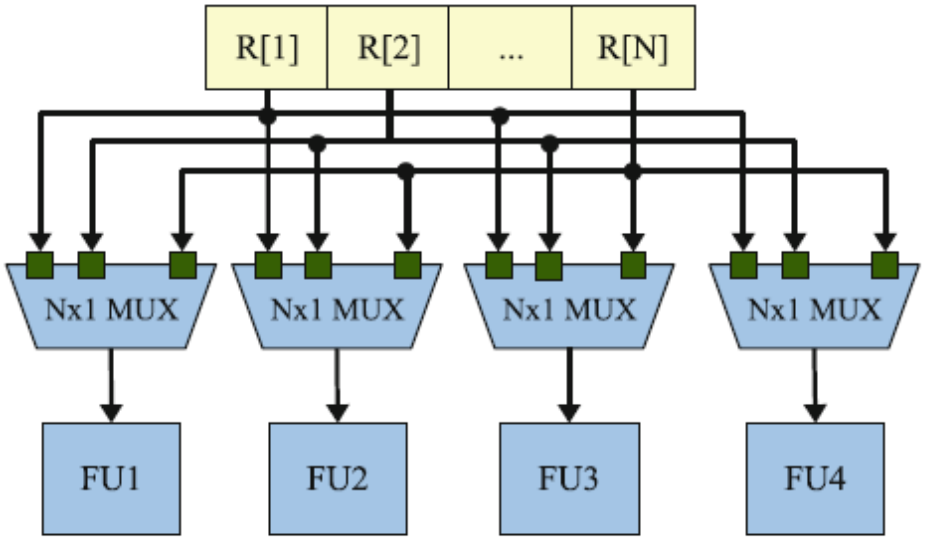


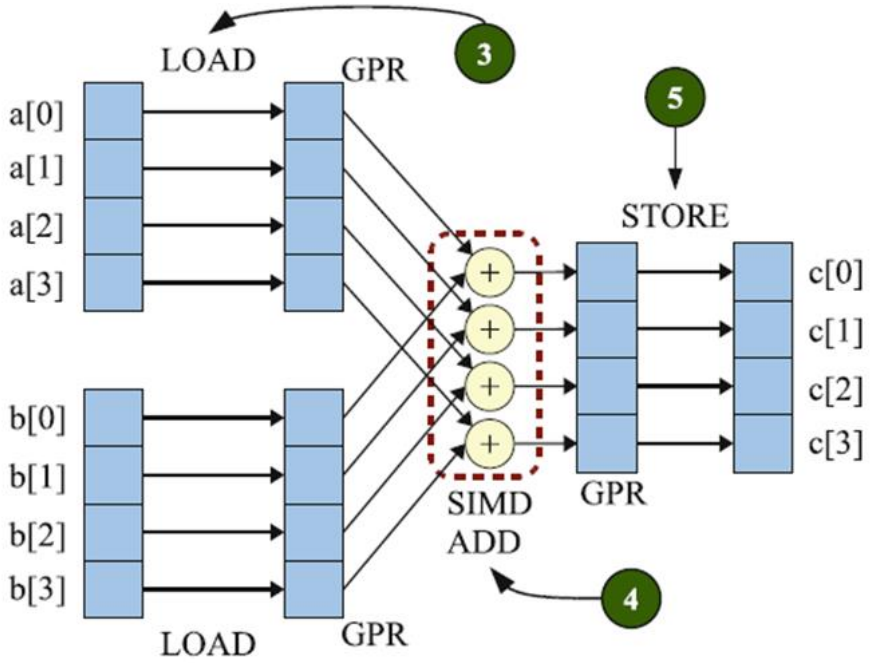
Fig. 2.1 Example pipeline architecture for ASIPs

K. Karuri and R. Leupers «Application Analysis Tools for ASIP Design, » Springer, 2011. ISBN 978-1-4419-8254-4

ASIP feature #3a : add parallelism



(a) Register Ports in a Non-clustered VLIW



(b) Implementation using SIMD

K. Karuri and R. Leupers «Application Analysis Tools for ASIP Design, » Springer, 2011. ISBN 978-1-4419-8254-4

ASIP feature #4 : add custom storage

- Registres internes aux unités fonctionnelles
- Registres à usage général, monolithiques (accessibles par tous les pipelines d'instructions) ou segmentés
- Mémoires bloc-notes *(scratchpad*) internes aux unités fonctionnelles
- Mémoire cache contrôlée en logiciel
- Mémoire cache transparente : contrôlée par le matériel
- Mémoire DRAM avec port d'accès unique
- Mémoire DRAM avec plusieurs ports d'accès

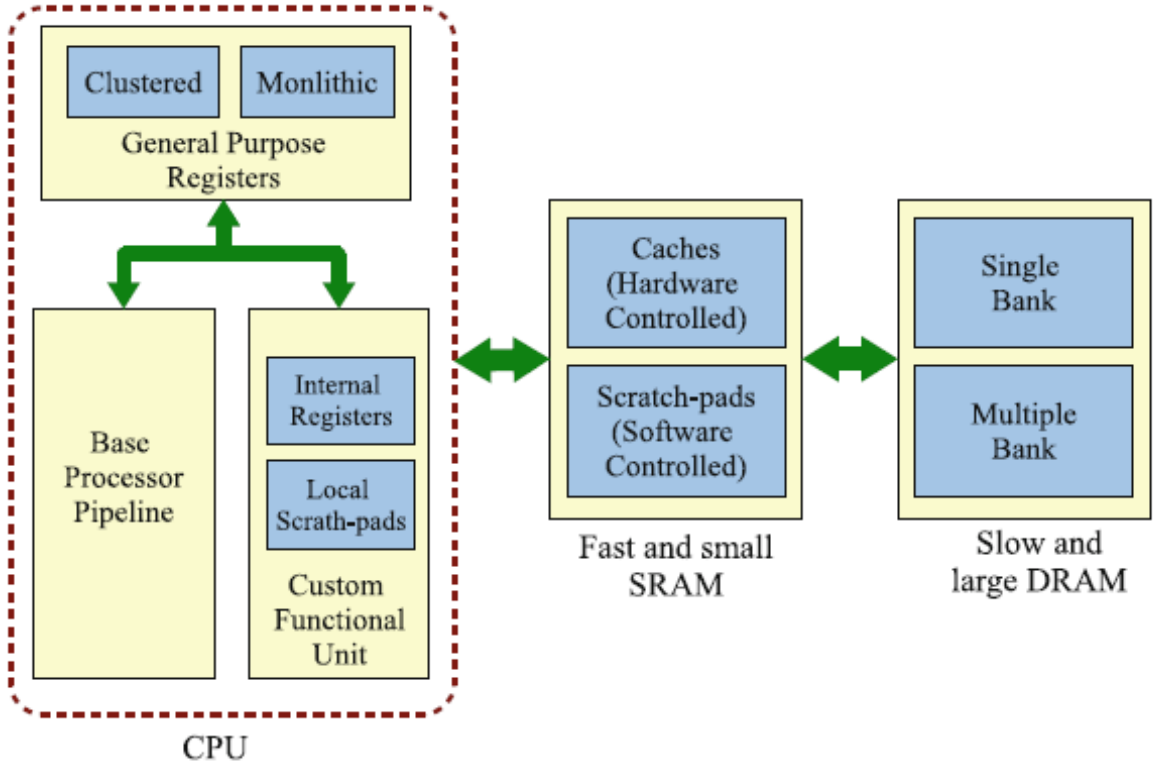
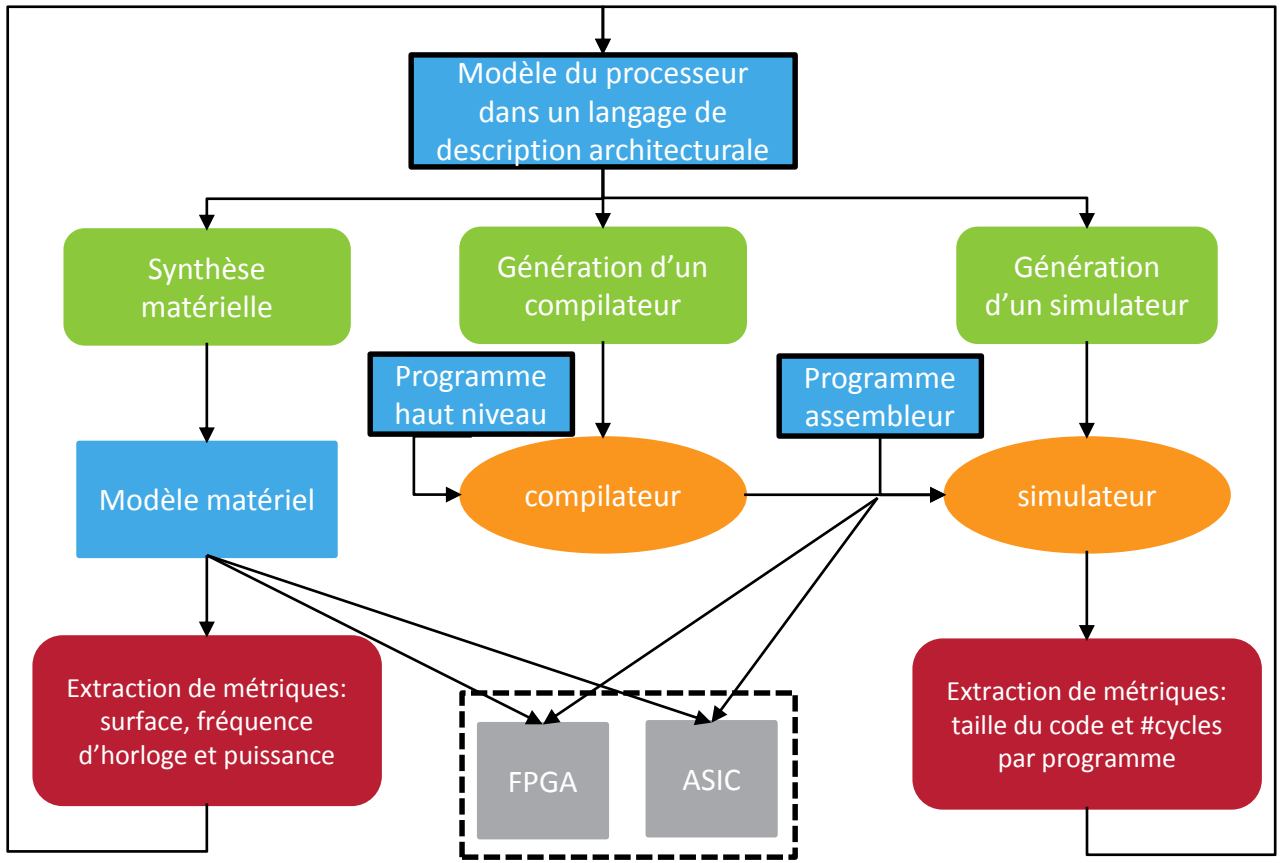


Fig. 2.8 Options for ASIP register file and memory hierarchy design

K. Karuri and R. Leupers «Application Analysis Tools for ASIP Design, » Springer, 2011. ISBN 978-1-4419-8254-4

ASIPs : two main approaches
#1 : Define the processor from scratch (often can modify a base model)



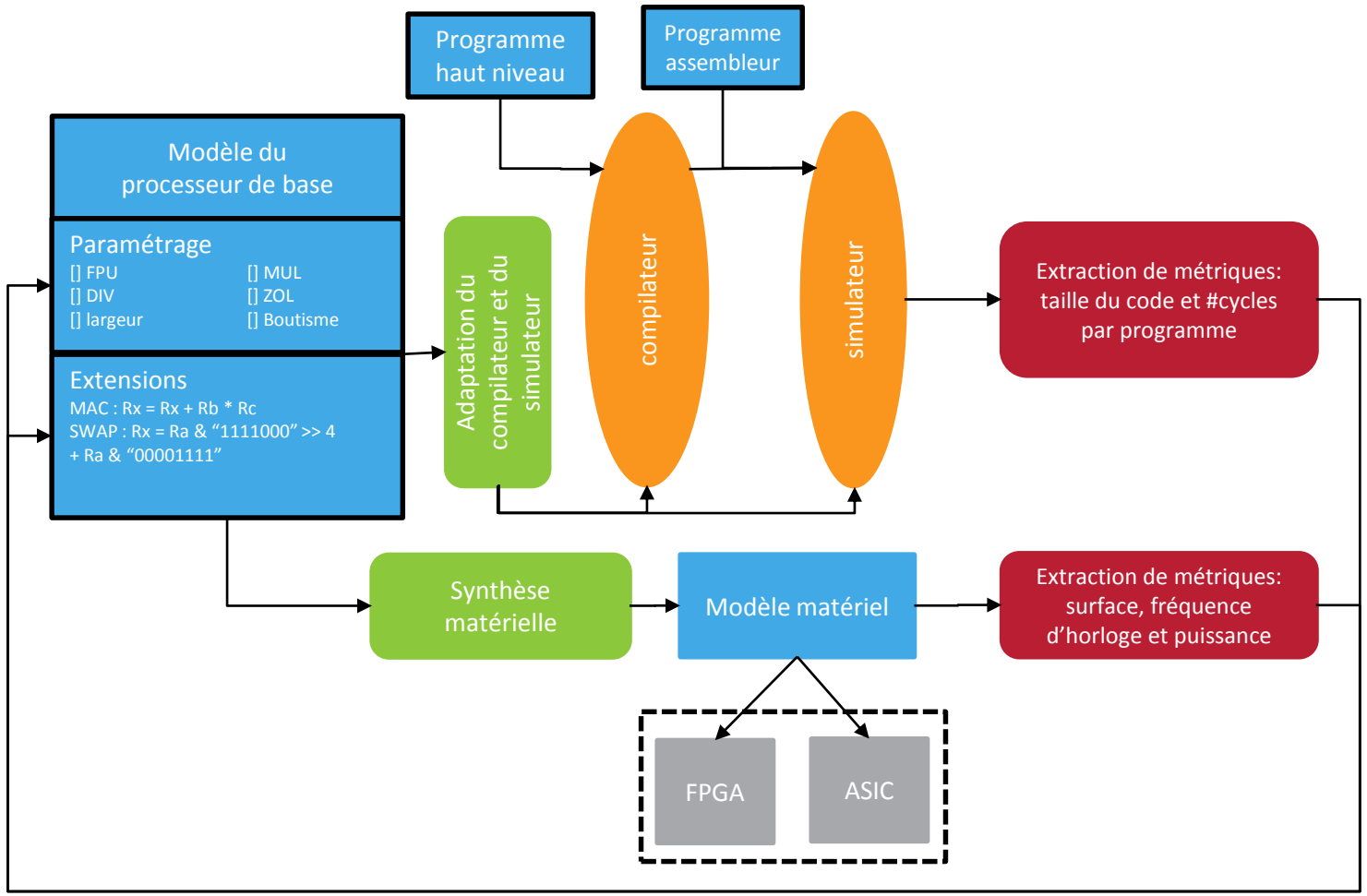
- Le processeur est modélisé à partir de zéro ou à partir d'une description existante; on peut lui ajouter ou retrancher des fonctionnalités.
- Un langage de description architecturale (*Architecture Description Language* – ADL) donne toute la flexibilité au concepteur d'ASIP.
- Trois grandes classes de langages de description architecturale :
  - Structurels : ils décrivent la structure du processeur; p. ex. MIMOLA, VHDL
  - Comportementaux : ils décrivent le comportement du processeur; p. ex. nML, ISDL
  - Mixtes p. ex. LISA, EXPRESSION

Correspond à l'approche Synopsys/ASIP Designer, ADL languages (LISA, nML, etc.)

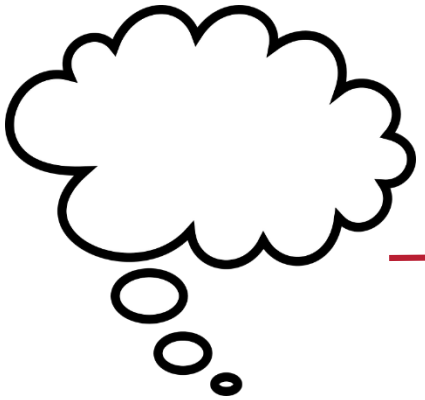
ASIPs : two main approaches
#2 : Configure and add to a base processor that cannot be altered

1. Processeurs d'une même famille et partageant un squelette d'architecture commun : *le processeur de base*
2. Paramétrage
3. Extensibilité
4. Outils de développement

Correspond à l'approche Cadence-Tensilica, Intel-Altera Nios, etc.



Exploration de l’espace de conception



Taille du code, nombre de cycles

Surface, fréquence d’horloge, puissance

Comment exprimer ces idées ? Avec quel langage? Quels termes? Quelle grammaire?

Modèle du processeur avec un ADL :

- Fonction
- Structure
- Paramètres
- …

Synthèse matérielle

Modèle matériel

Programme test

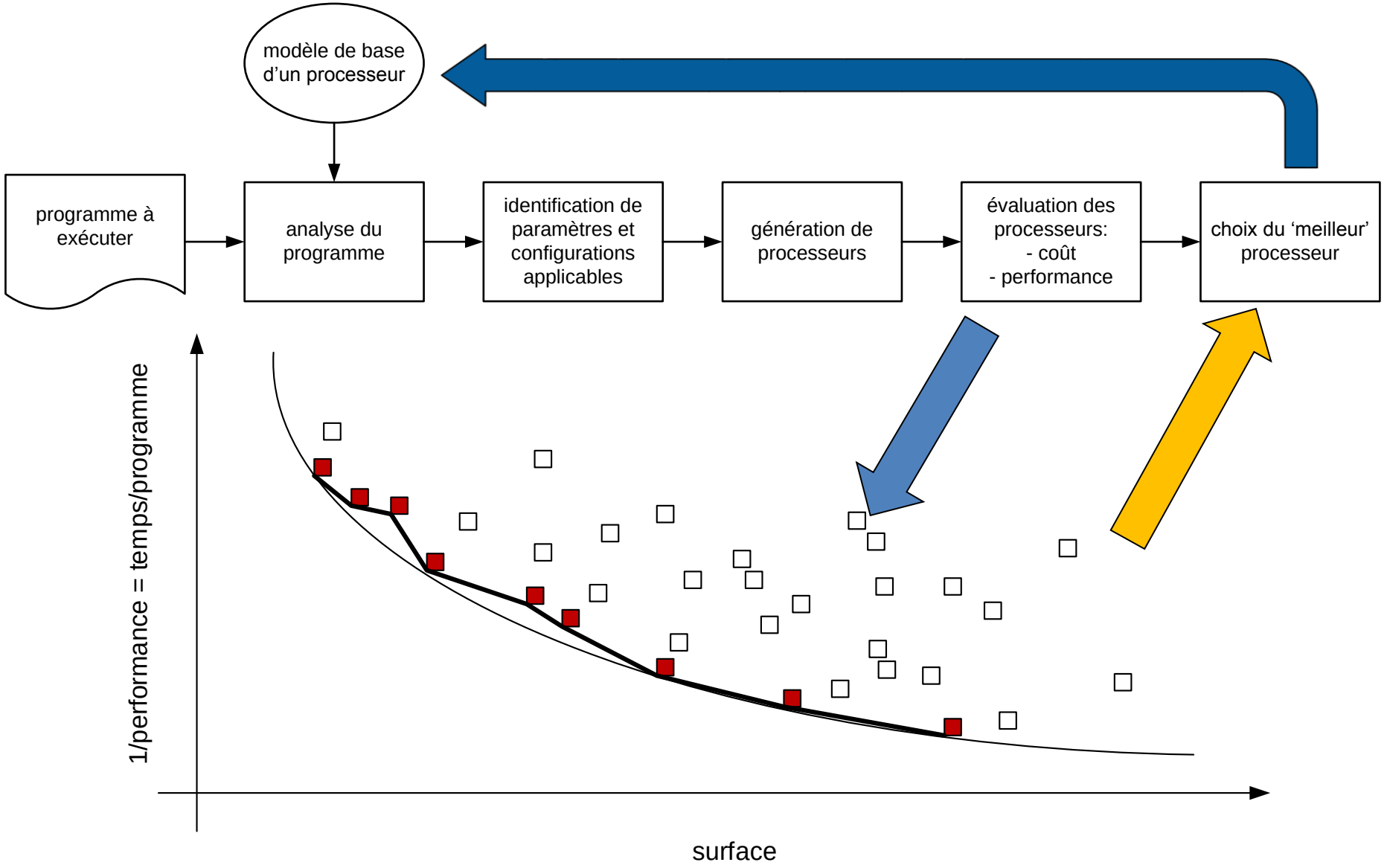
Génération d’un compilateur

compilateur

Génération d’un simulateur

simulateur

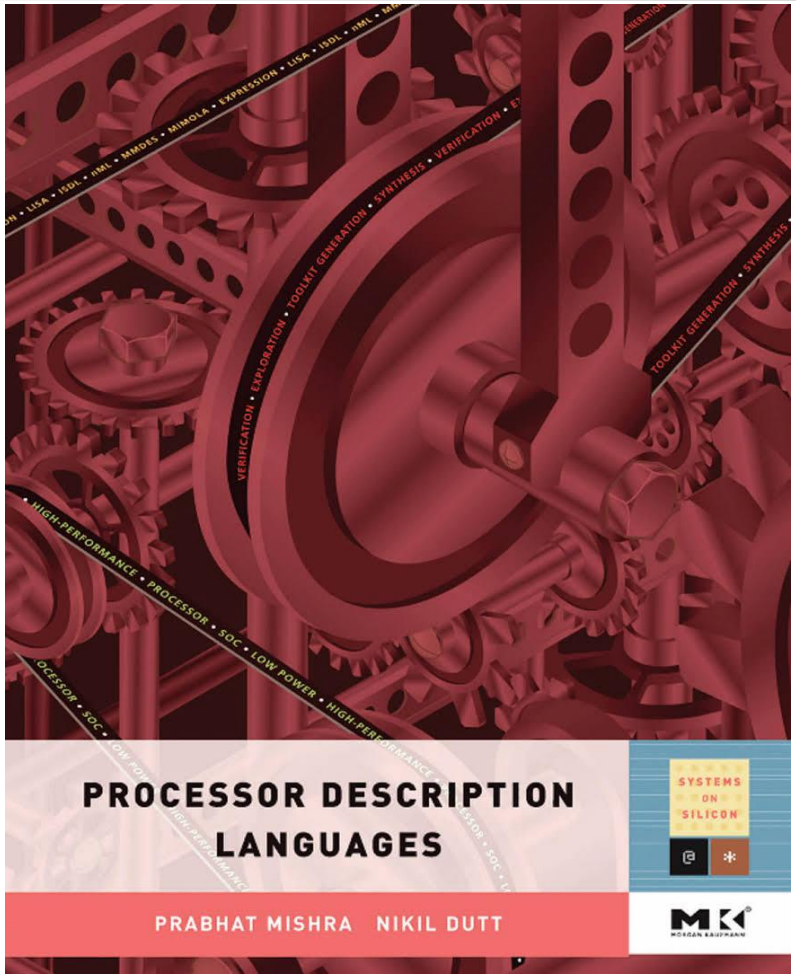
Quel est le flot suivi par un humain?



ADL pour la description de processeurs

Processor Description Languages vs Architecture Description Languages …

- Un premier ADL, MIMOLA, a été proposé en Allemagne dans les années 1970.
- Plusieurs ont suivi en Allemagne, aux Japon, au Brésil, aux USA et ailleurs :
  - nML, LISA, Expression, ASIP Meister, TIE (Tensilica/Cadence), ArchC, HMDES, ISDL, etc.
- La différence entre les ADL et les autres langages vient surtout:
  - du niveau d'abstraction désiré:
    - trop haut: on a le comportement global du système, mais pas celui de ses composantes;
    - trop bas: on est perdu dans les détails (e.g. VHDL).
  - de la possibilité d'illustrer la concurrence et la synchronisation.



Mishra et Dutt, Processor Description Languages, Morgan Kaufmann 2008.

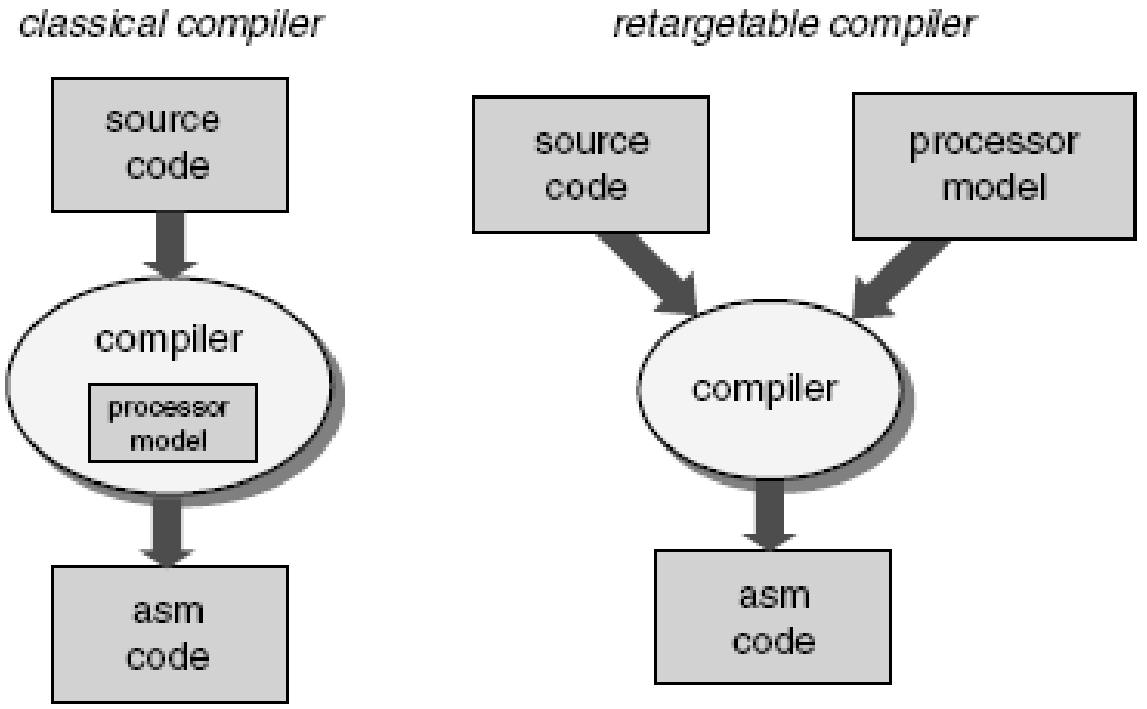
Taxinomie des ADL

- On peut classifier les ADL dans un espace à deux dimensions:
  - par le contenu: structurel, comportemental ou mixte;
  - par l'objectif: synthèse, validation, compilation, et simulation.
- Pour la dimension du contenu, on détermine la nature de l'information capturée par l'ADL.
- Pour la dimension de l'objectif, on détermine le produit principal obtenu de la description par ADL.
- Il y a une certaine correspondance entre le contenu et l'objectif.

<table>
<tr><th colspan="2" rowspan="2">classification des ADL</th><th colspan="3">contenu</th></tr>
<tr><th>Structurel 'architecture-centric'</th><th>Mixte</th><th>Comportemental 'instruction-set centric'</th></tr>
<tr><td rowspan="4">objectif</td><td>synthèse</td><td rowspan="2">MIMOLA, UDL/I, AIDL</td><td rowspan="4">LISA, EXPRESSION</td><td rowspan="2">?</td></tr>
<tr><td>validation</td></tr>
<tr><td>compilation</td><td rowspan="2">?</td><td rowspan="2">nML, ISDL, CSDL</td></tr>
<tr><td>simulation</td></tr>
</table>

Retargetable compilers

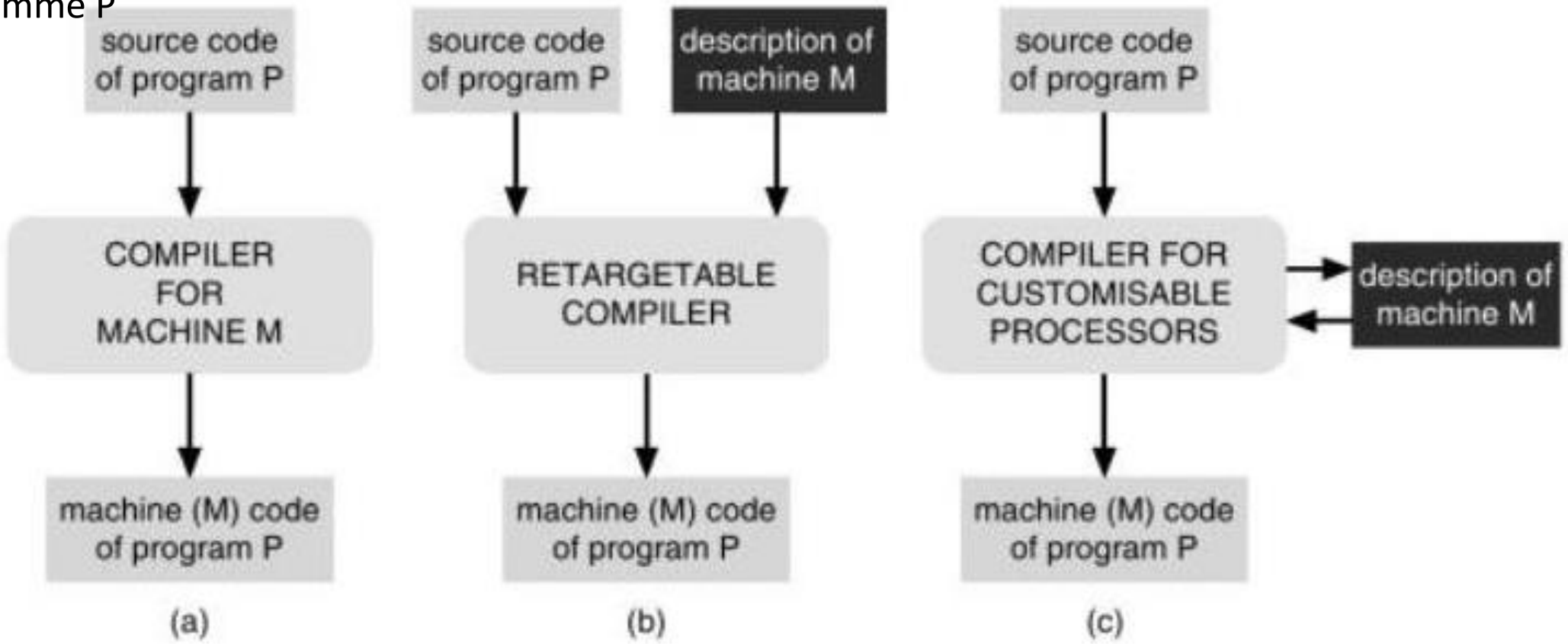
- Pour la partie finale, le compilateur doit avoir un modèle du processeur cible.
- Le modèle inclut toutes les ressources pertinentes du processeur:
  - le jeu d'instructions;
  - les blocs de registres;
  - les contraintes d'ordonnancement.
- Un compilateur à cible unique intègre ce modèle.
- Un compilateur à cibles multiples doit pouvoir lire le modèle du processeur désiré.



P. Ienne et R. Leupers, *Customizable Embedded Processors*, Morgan Kaufmann, 2007.

Automating the configuration and extension of an ASIP

- (a) modèle traditionnel avec processeur fixe
- (b) processeur paramétré et configuré par un humain; besoin d'un compilateur à cibles multiples
- (c) processeur paramétré et configuré automatiquement pour 'optimiser' la performance en fonction du programme P



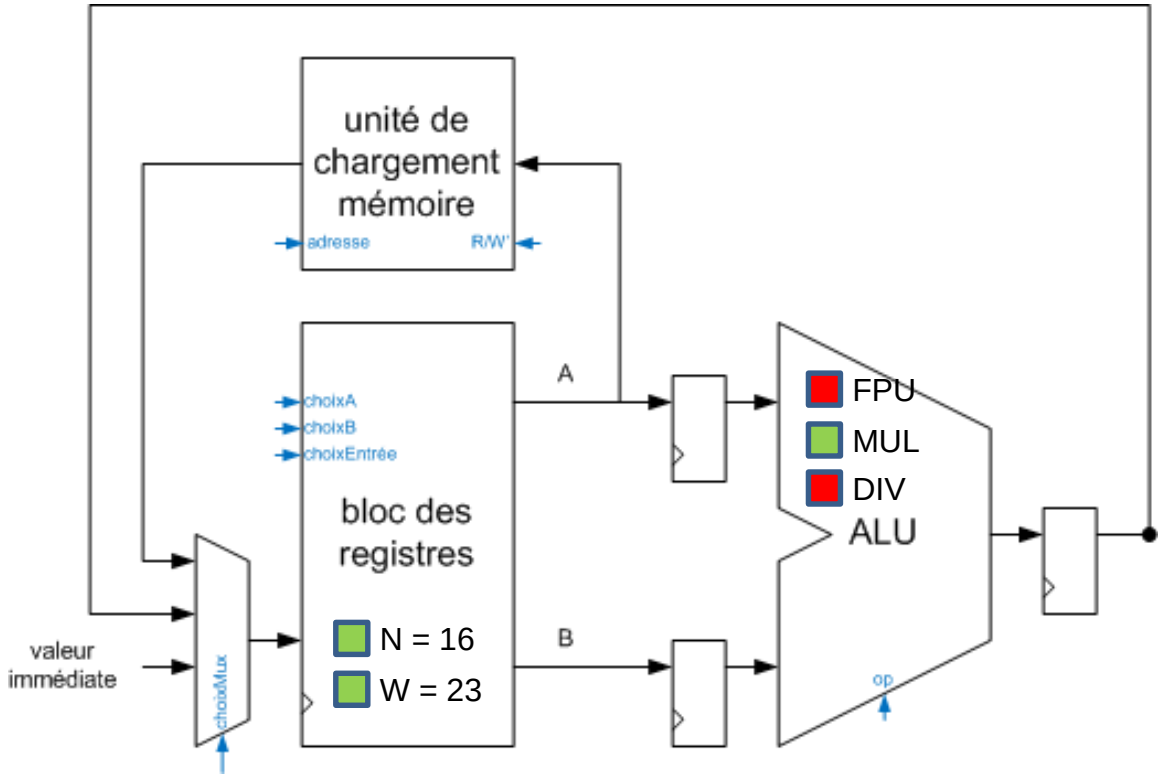
Pozzi et Ienne, chap. 7 de *Customizable Embedded Processors*, Morgan Kaufmann, 2007.

Automating the configuration and extension of an ASIP

- En pratique, plusieurs concepteurs passent de l'idée d'un algorithme à son implémentation en C.
- Pourtant, en passant à C, on fait déjà des suppositions architecturales fondamentales, comme par exemple:
  - une instruction à la fois;
  - ordre séquentiel;
  - ensemble 'traditionnel' d'opérations disponibles;
  - architecture Von Neumann ou Harvard;
  - etc..
- Pour deux concepteurs considérant le même problème, avec des solutions implémentées dans le même langage (C), il est probable d'obtenir deux versions assez différentes de code.
- Par exemple, pour l'outil Tensilica XPRESS, un style particulier d'écriture de code C est recommandé:
  - ne pas utiliser de langage machine;
  - ne pas utiliser de fonctions prédéfinies (compiler intrinsics);
  - utiliser des tableaux au lieu de pointeurs;
  - calculer les indices dans les tableaux de façon régulière.
- Pour automatiser le processus de paramétrage et de configuration du processeur, il faut déterminer un mécanisme et un niveau de description de l'algorithme.

Automating the configuration and extension of an ASIP

- Une option de paramétrage consiste à activer ou désactiver certaines caractéristiques:
  - multiplicateurs;
  - unités à point flottant;
  - accélérateur de boucles;
  - calcul min/max;
  - etc.
- Pour automatiser le paramétrage de ce genre d'option, une inspection simple du code peut suffire, par exemple pour déterminer si les unités de calcul en cause sont utilisées ou non (multiplication, virgule flottante, etc.).
- Il faut tenir compte du traitement fait sur les données ainsi que sur les adresses et pointeurs.



Automating the configuration and extension of an ASIP

- Une autre option de paramétrage consiste à choisir la taille d'une caractéristique:
  - largeur du chemin de données;
  - nombre de pipelines;
  - nombre et largeur des bus d'interface;
  - nombre de registres;
  - présence, largeur et profondeur de cache; etc.
- Pour ces paramètres, une possibilité consiste à considérer différentes options de taille et de largeur puis profiler le code automatiquement pour observer le coût et la performance.

- C'est un problème d'optimisation à plusieurs dimensions, donc qui peut facilement devenir impossible à résoudre dans un temps raisonnable.
- Le fait de restreindre le nombre de valeurs possibles (e.g. 8, 16, 32, 64 bits) facilite l'exploration de l'espace des choix.

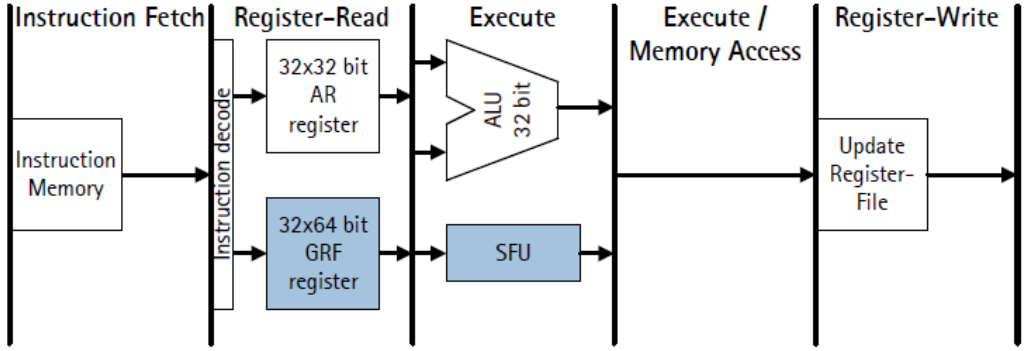
Rappel : article de Sheldon et al., 2006

Automating the configuration and extension of an ASIP

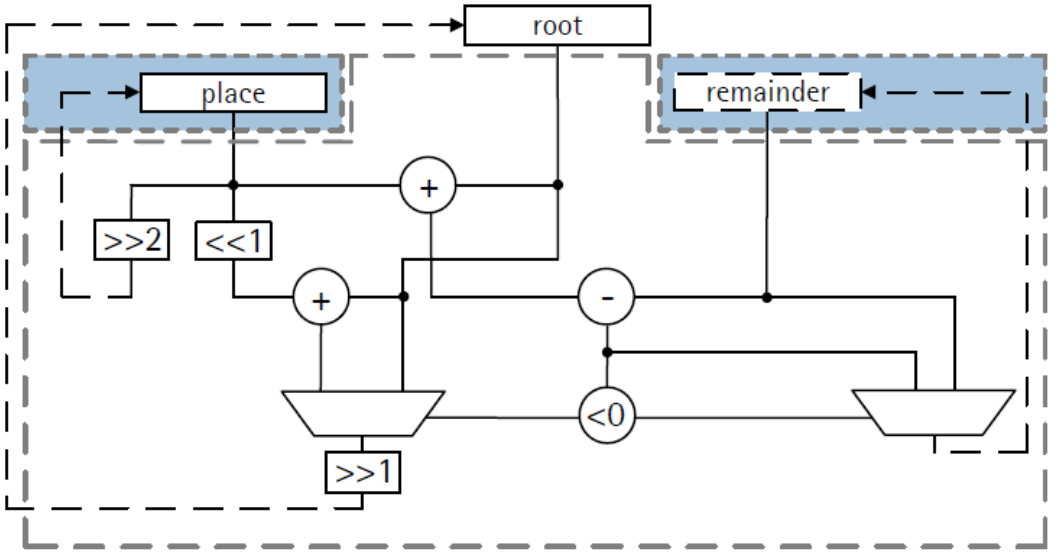
- Une autre option consiste à réduire la précision des calculs:
  - en modifiant les unités fonctionnelles; ou,
  - en arrondissant les résultats intermédiaires.
- Il faut à nouveau considérer plusieurs options de taille et de largeur et profiler le code automatiquement pour observer:
  - la qualité des résultats obtenus (voir : Catudal et. al, ISCAS 2005);
  - les coûts en matériel;
  - le temps d'exécution.

A design example
ASIP for image processing : SIFT

- SIFT algorithm
- Base processor : Tensilica LX
- 11 custom instructions
- 167x reduction in cycles, 6x area



Category	Instruction	Description	#EX
Data Handling	LOAD64	Load 64 bit into register	1
	STORE64	Store 64 bit word to memory	1
	MOVE64	Move 64 bit to different register	1
	SHIFT64	Shift 16 bit subwords by 1 subword	1
Special Functional Units (Base CV)	FIR	Symmetric and separable FIR-filter vertical and horizontal unit	1
Special Functional Units (SIFT)	ISQRT_INIT	Initialization step	1
	ISQRT	Calculation of integer square root	1
	ATAN	Calculation of arctangent	1
	SINE	Caluclation of sine, cosine	1
	HIST_VAL	Calculation of histogram entries	1
	HIST_POS	Calculation of histogram positions	1



Mentzer et al., Instruction-Set Extension for an ASIP-based SIFT Feature Extraction, SAMOS 2014.

A design example
ASIP for image processing : SVM

- Support Vector Machine algorithm
- Base processor : base RISC available in Synopsys' Processor Designer tool
  - 1. Vector/SIMD instructions and registers
  - 2. Wide SRAM interface and Wide Streaming interface
  - 3. 2-Dimensional Post Increment Vector Load
  - instruction
  - 4. Instructions with the Vector Register as the result
  - 5. Nested Hardware loops
  - 6. 2-Slot VLIW and Software pipelining
- Results :
  - 63x reduction in # of cycles
  - 6.5 mW (+SRAM) for 630 kSVM/sec
    - vs GPU @236 W
    - Vs Cortex-A8 16 kSVM/sec

Architectural Improvements	Cycles Consumed	% Improvement from previous
Original RISC	21710	-
Vector/SIMD instructions and registers (A)	15215	29.9%
Wide SRAM Interface and Wide Streaming Interface (B)	3227	78.8%
2-dimensional Post Increment load (C)	2070	35.8%
Instructions with Vector Register as the result (D)	1147	44.6%
Nested Hardware Loops (E)	556	51.5%
2-Slot VLIW + Software Pipelining (F)	344	38.1%

Gupta et al., Accelerating SVM on ultra low power ASIP for high throughput Streaming Applications,5 IC VLSI 2014.

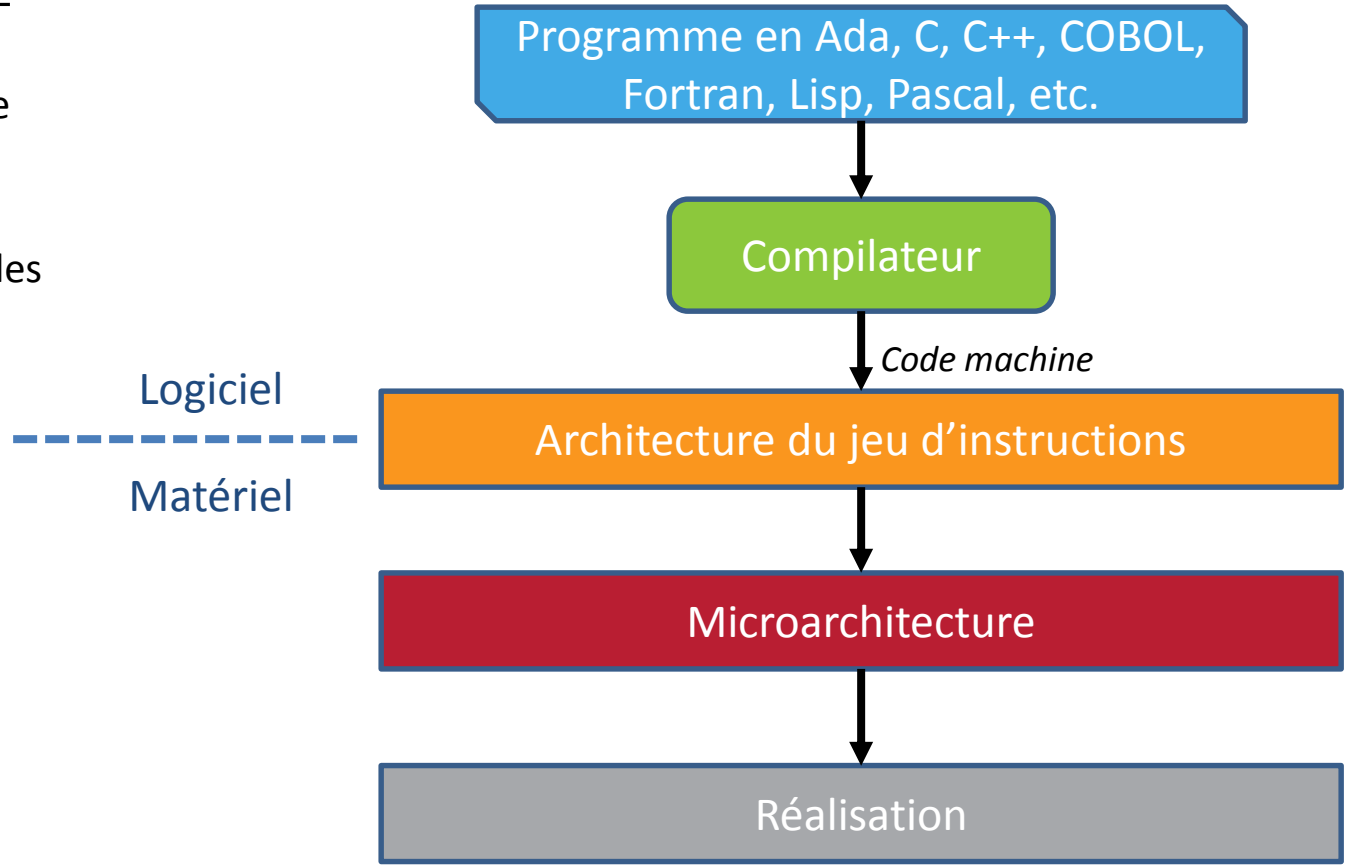
Outline

- The war is on : ACAP, ASIC, ASIP, CGRA, CPU, FPGA, GPU, SIMD, VLIW
- A few details about Application-Specific Instruction-set Processors (ASIPs)
  - Architecture
  - Instruction set
  - Memory hierarchy
  - Design tools
- Tools, tools and tools
- A couple recent research results

Thank you for your attention ☺

Jeu d'instructions, microarchitecture et réalisation

- Architecture du jeu d'instructions ou jeu d'instructions (*Instruction Set Architecture* – ISA)
  - Définit l'apparence fonctionnelle du processeur telle que vue par le compilateur ou le programmeur de code machine
  - Modèle d'exécution, opérations, types de données, modes d'adressage, registres visibles, etc.
- Microarchitecture ou implémentation
  - Point de vue du concepteur du processeur
  - Circuits logiques qui implémentent le jeu d'instructions
  - Registres physiques, unités fonctionnelles, pipelines
- Réalisation
  - Point de vue du concepteur de circuit ou de la puce
  - Réalisation physique des circuits logiques
  - Portes, cellules, transistors, fils, etc.

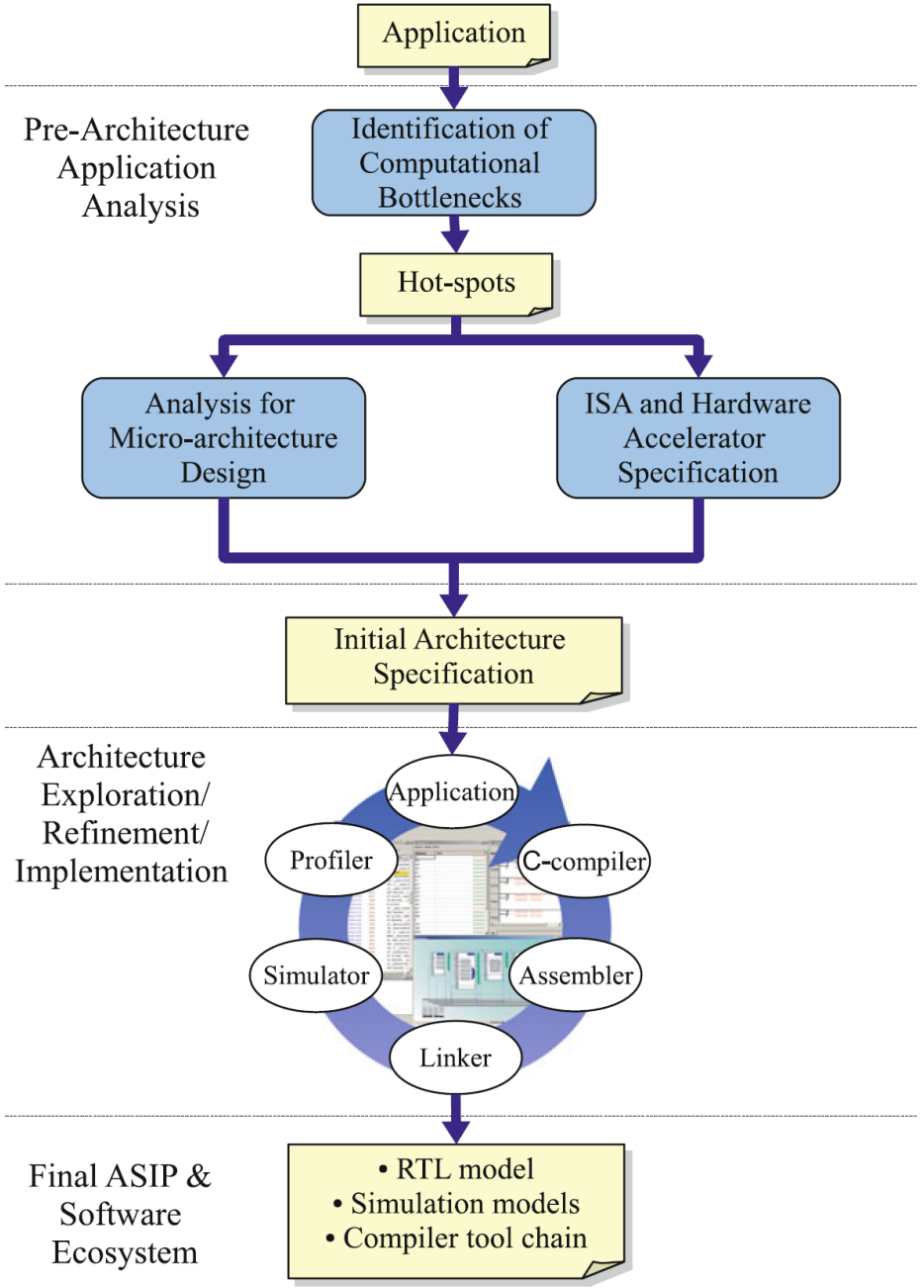


Une ISA peut avoir plusieurs microarchitectures,
et chaque microarchitecture peut avoir plusieurs réalisations!

4. Outils de développement

- Outils de développement
  - Profilage de l'application et identification des goulots d'étranglement;
  - Configuration du processeur :
    - Choix des paramètres;
    - Description des extensions;
  - Simulateur fonctionnel;
  - Compilateur à cibles multiples;
  - Synthèse de la microarchitecture;
    - Il faut spécifier pour quelle technologie;
    - Mesure des coûts;

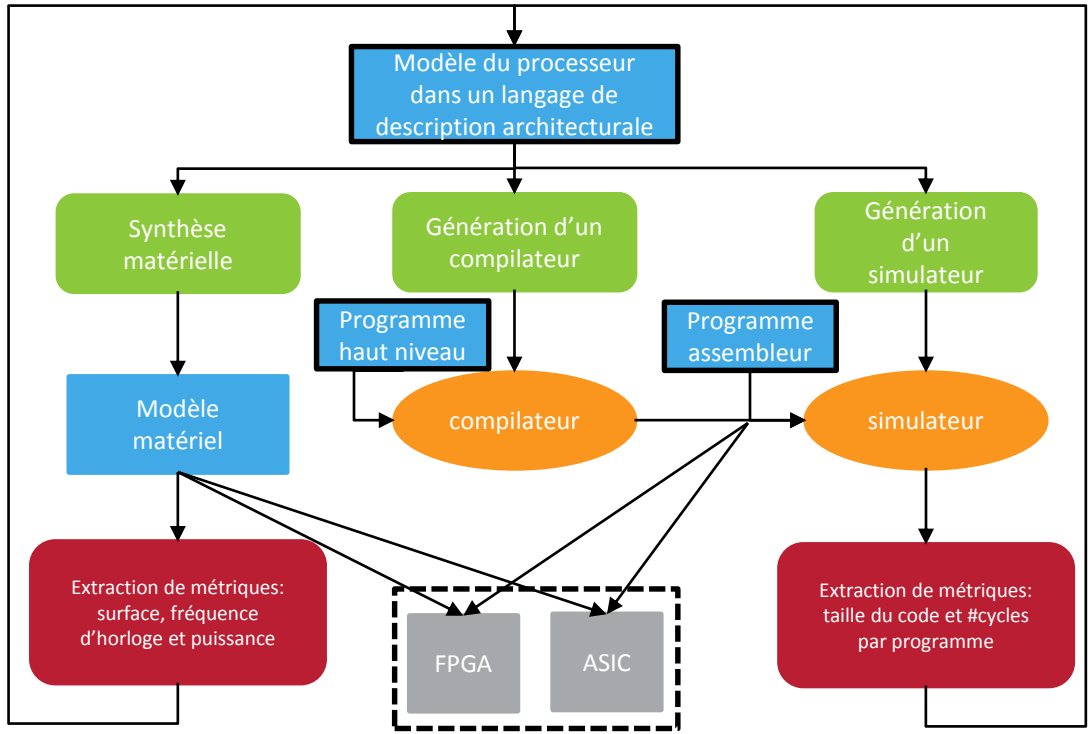
Les outils doivent être dynamiques : ils doivent tenir compte des modifications dues au paramétrage et aux extensions, ou à la description avec un ADL!



Karuri et Leupers, Application analysis tools for ASIP Design, Springer 2011.

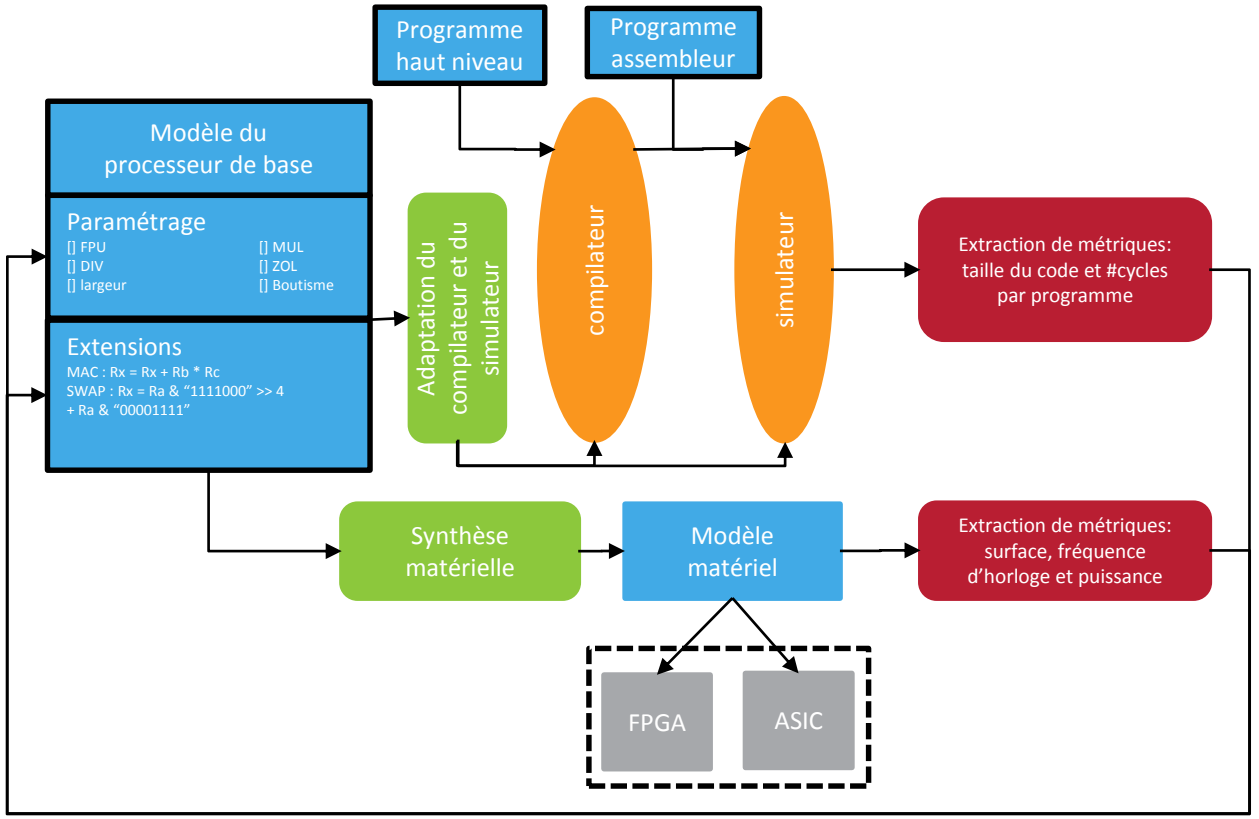
ASIPs : two main approaches

- Define the processor from scratch (often start from a base model that can be completely modified)



Correspond à l'approche Synopsys/ASIP Designer, ADL languages (LISA, nML, etc.)

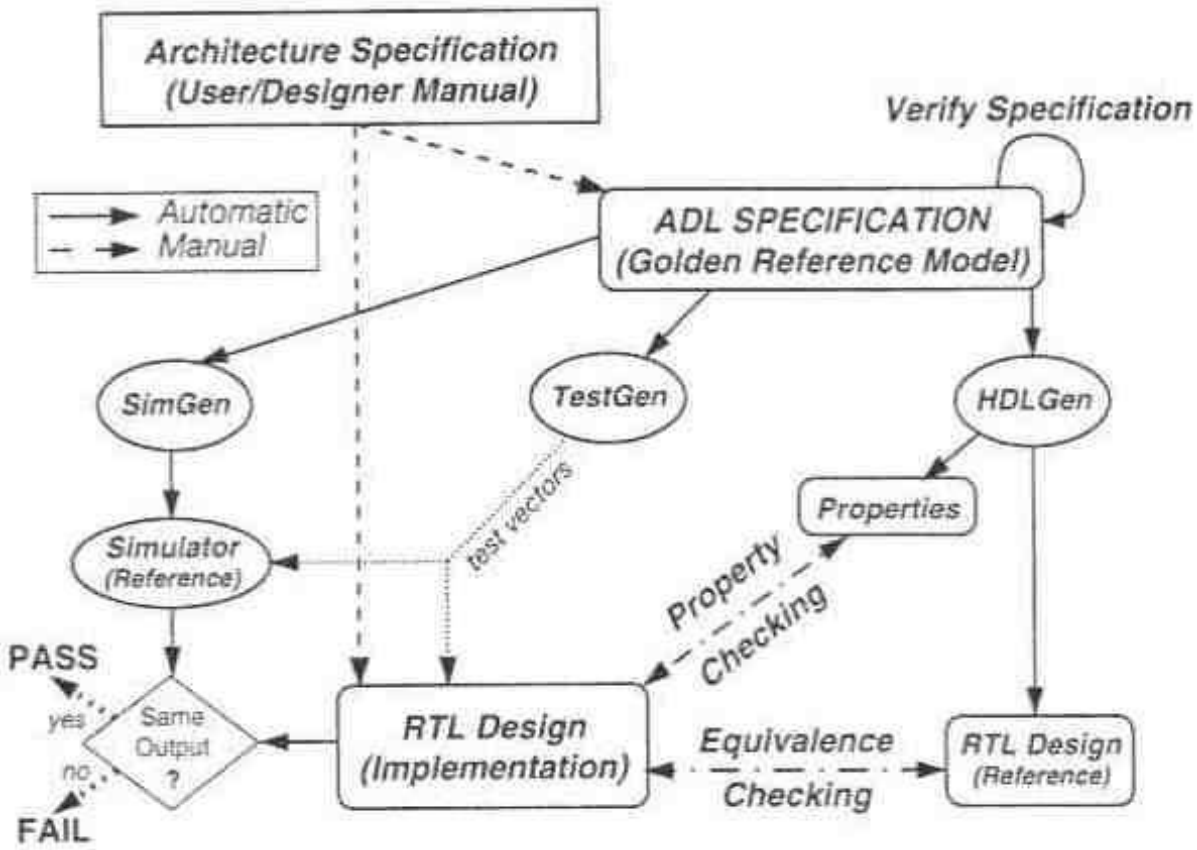
- Configure and add to a base processor that cannot be altered



Correspond à l'approche Cadence-Tensilica, Intel-Altera Nios, etc.

Flot de vérification avec un ADL

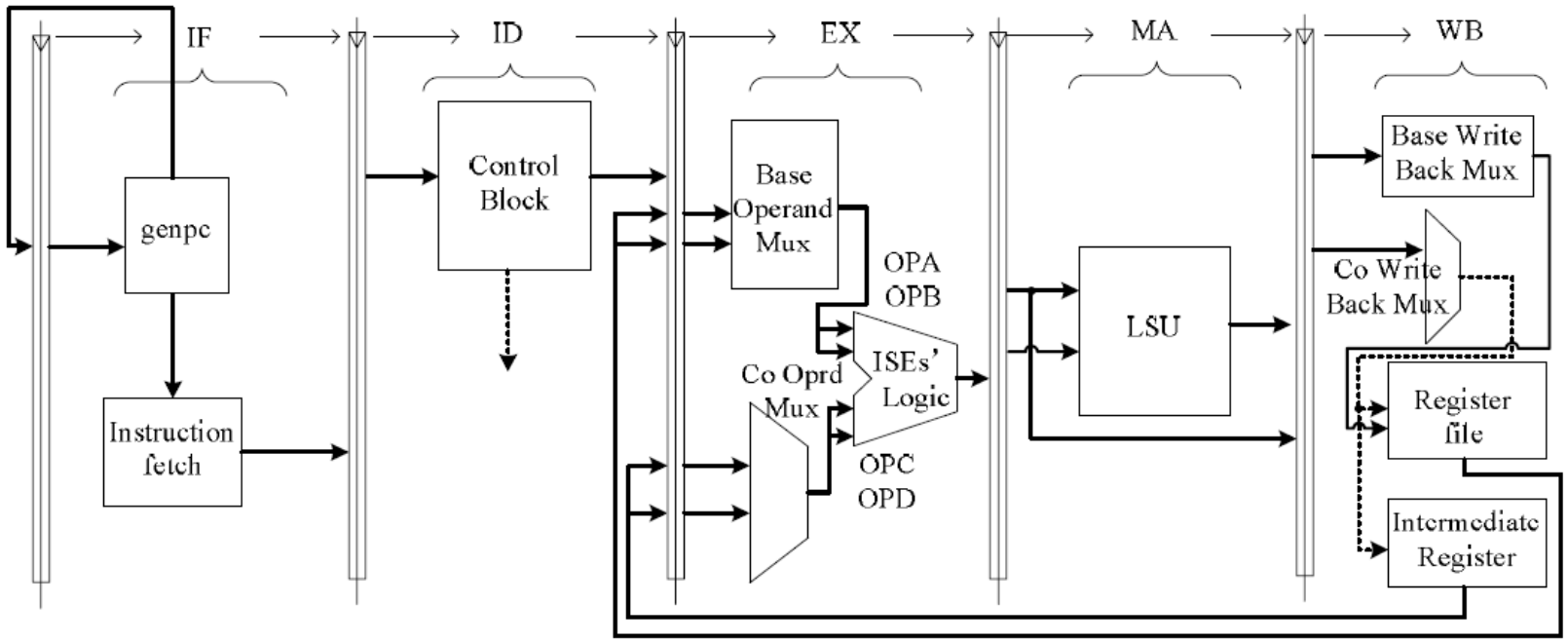
- À partir d'une spécification, on produit la description du processeur avec un ADL
- Vérifier que la description correspond à la spécification.
- À partir de la description, on produit le simulateur, la liste des interconnexions et des cas de test.
- La liste des interconnexions est implémentée.
- Le système implémenté est comparé à la liste des interconnexions, tout en vérifiant des propriétés spécifiques.
- Les vecteurs de test sont appliqués au simulateur ainsi qu'au système implémenté, et on vérifie que la sortie est la même dans les deux cas.
- On observe qu'on a effectivement deux chemins à partir de la description du processeur en ADL.



P. Ienne et R. Leupers, *Customizable Embedded Processors*, Morgan Kaufmann, 2007.

A design example
ASIP for cryptography

- Base processor : OR1K, open-source RISC
- 10 custom instructions
- 79% reduction in cycles



Jinguo et al., Fine-grained Analysis and Design of ASIP Instruction Set for Application of Encryption, WCNS 2012.