

---

# Accélération matérielle

---

*Olivier Romain*

*Professeur des Universités*

[olivier.romain@cyu.fr](mailto:olivier.romain@cyu.fr)



Équipes Traitement  
de l'Information  
et Systèmes

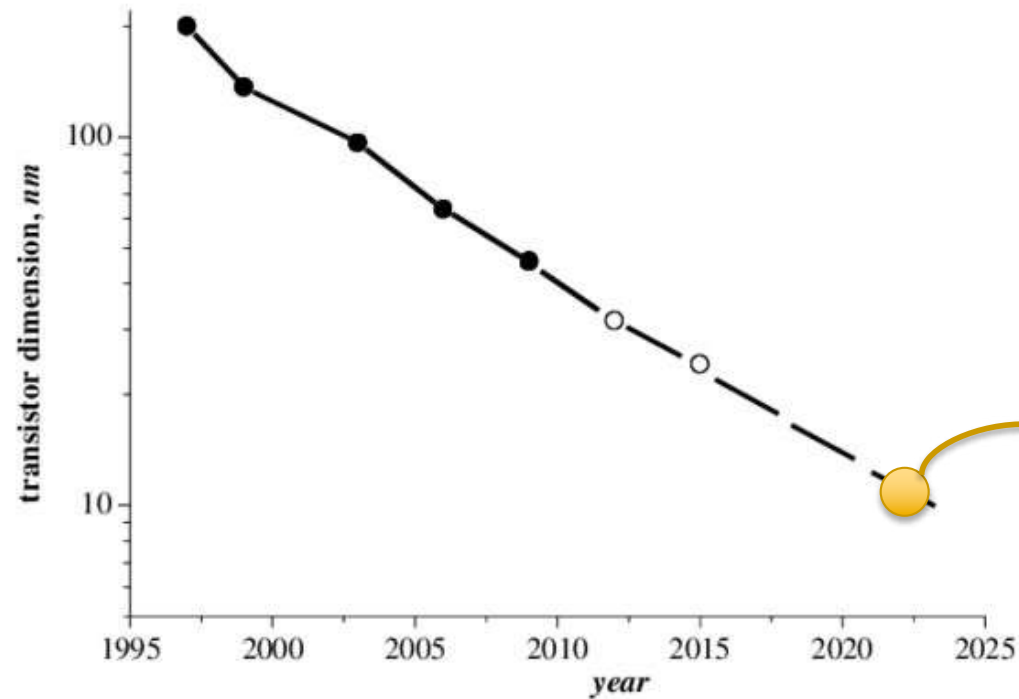


---

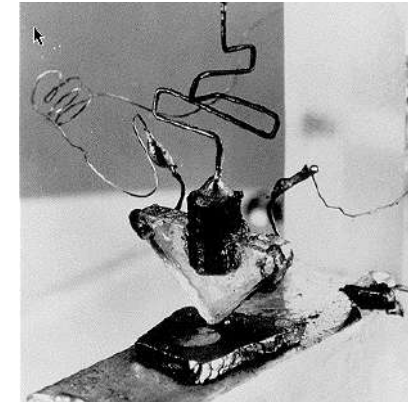
# Objectif du cours

- Présenter les différentes architectures d'accélération matérielle
  - FPGA et co-design : Olivier ROMAIN
  - GPU : Alexandre Bordat
  - Multi-cœur : Stéphane Zuckerman
  - TPU : Petr Dobias
-

# A la base du SoC : le transistor

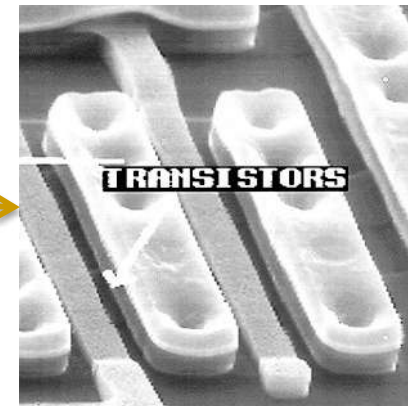


1947 - cm



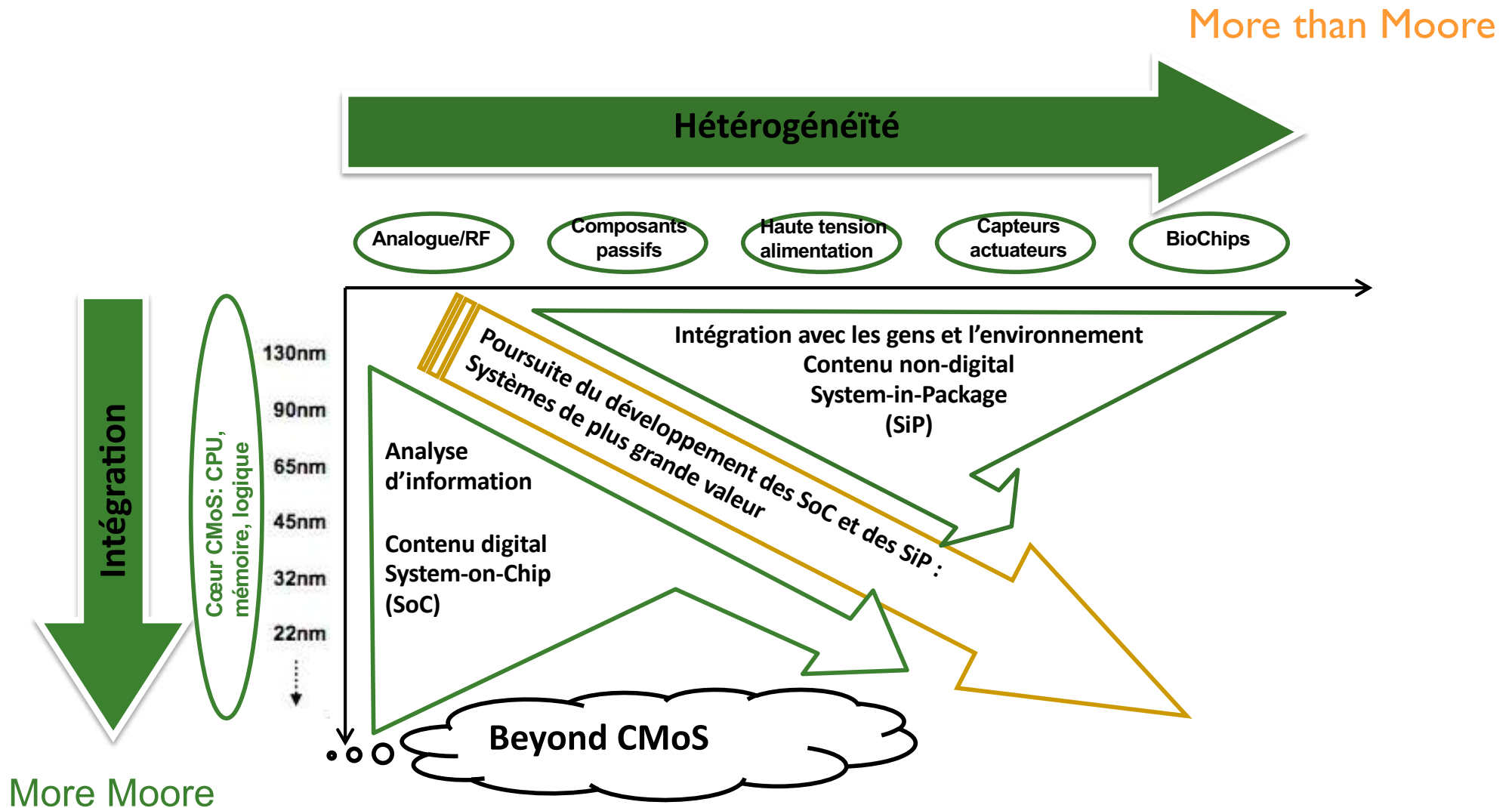
Quelques cm

2023 - 8 nm



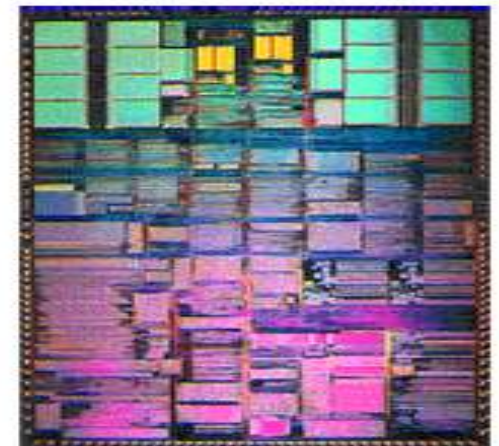
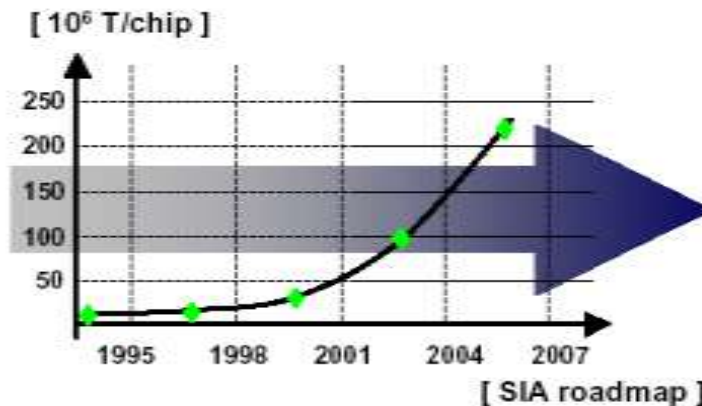
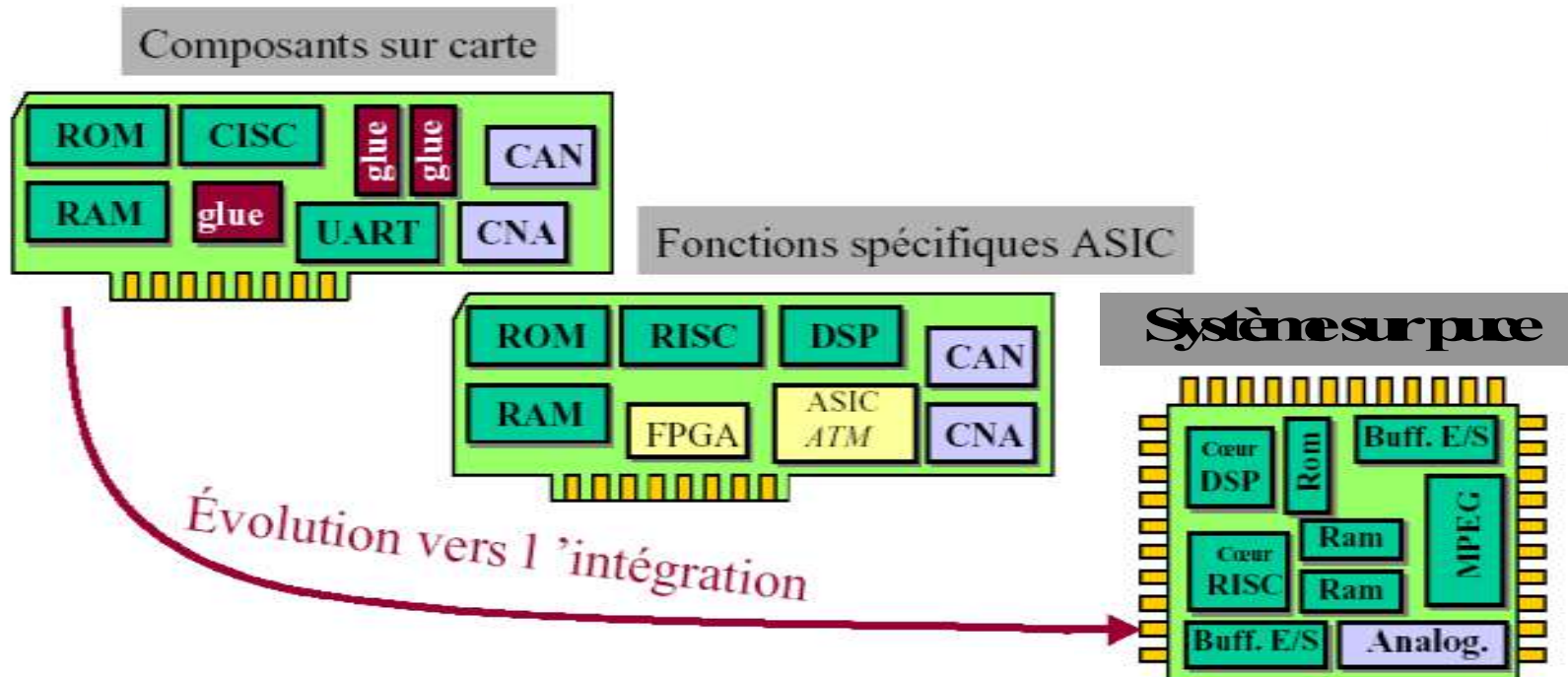
Quelques nm

# Evolution technologique : IC to SoC

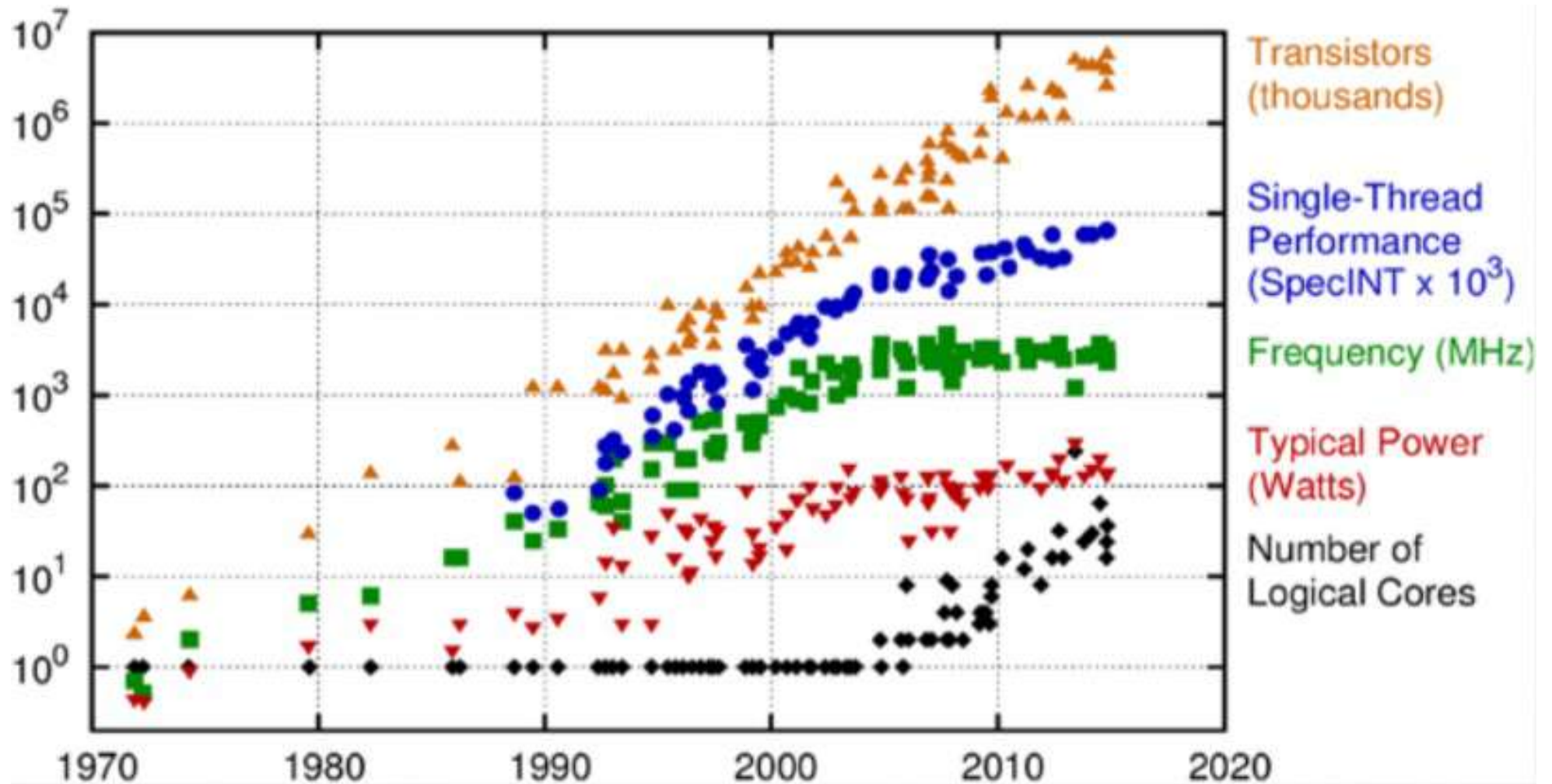


\* *ITRS – International Technologies Roadmap of Semiconductor*

# Introduction : Evolution des systèmes

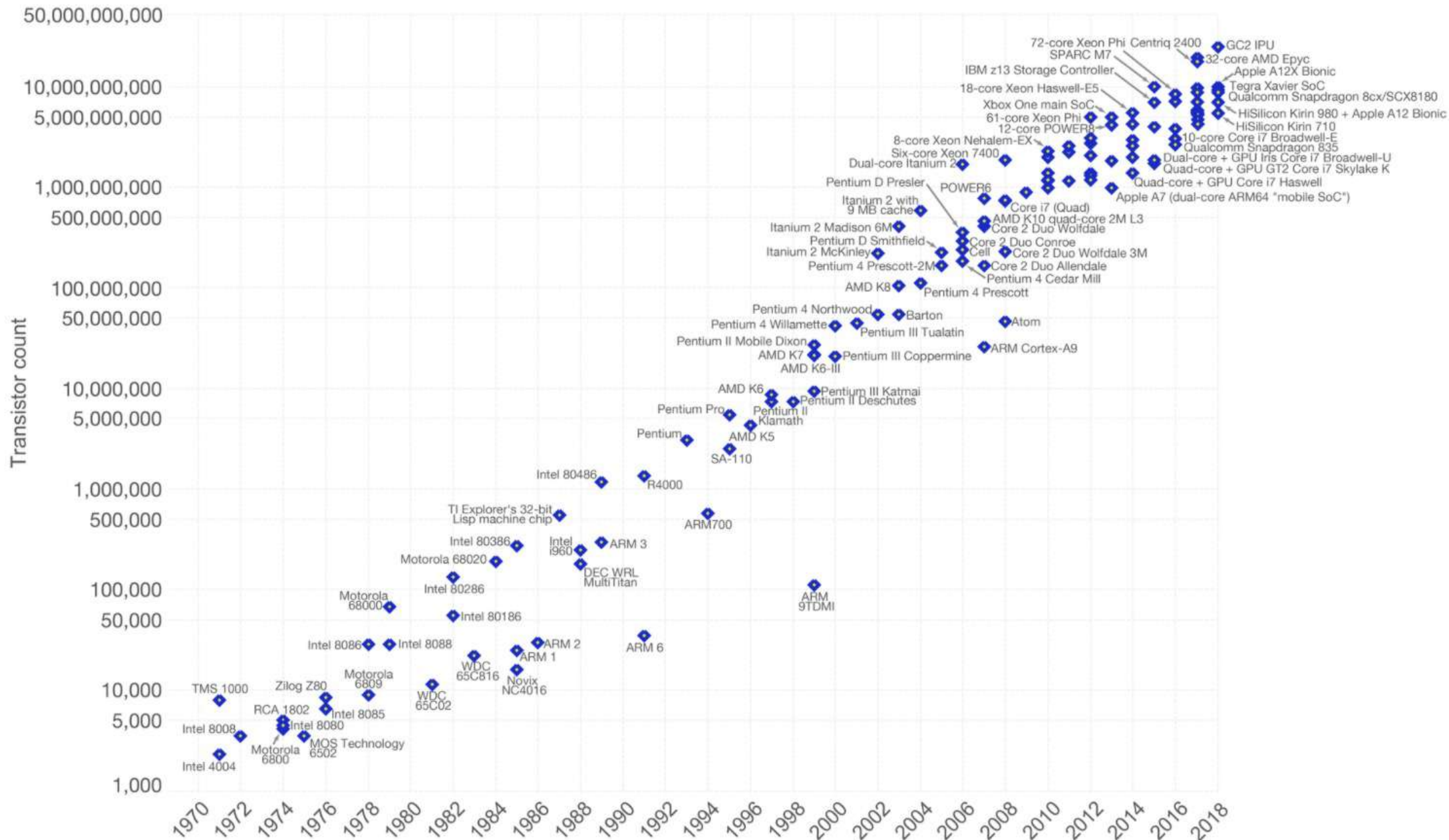


# Evolution des performances



Source : Horowitz2016





# Introduction : Pr vision d volution technologique



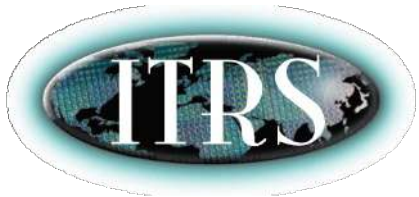
## International Technology Roadmap for Semiconductors

Table ES2 Overall Roadmap Technology Characteristics

2017 IRDS Executive Summary - ORTC							
YEAR OF PRODUCTION	2017	2019	2021	2024	2027	2030	2033
Logic device technology naming	P54M36	P48M28	P42M24	P36M21	P32M14	P32M14 T2	P32M14 T4
Logic industry "Node Range" Labeling (nm)	"10"	"7"	"5"	"3"	"2.1"	"1.5"	"1.0"
Logic device structure options	finFET FDSOI	finFET LGAA	LGAA finFET	LGAA VGAA	LGAA VGAA	VGAA, LGAA, 3DVLSI	VGAA, LGAA, 3DVLSI
LOGIC CELL AND FUNCTIONAL FABRIC TARGETS							
Average cell width scaling factor	1.00	0.90	0.90	0.90	0.90	0.90	0.90
LOGIC DEVICE GROUND RULES							
MPU/SoC Metalx � Pitch (nm) [1,2]	18	14	12	10.5	7.0	7.0	7.0
Physical gate length for HP Logic (nm) [3]	20	18	16	14	12	12	12
Lateral GAA (nanosheet) Minimum Width (nm)			7.0	7.0	6.0		
Minimum Device Width (fin, nanosheet) or Diameter (nm)	8	7.0	7.0	7.0	6.0	6.0	6.0
LOGIC DEVICE Electrical							
Vdd (V)	0.75	0.70	0.65	0.65	0.65	0.60	0.55
DRAM TECHNOLOGY							
DRAM � Pitch (nm) [1]	18	17.5	17	14	11	8.4	7.7
DRAM cell size factor: aF <sup>2</sup> [11]	6	6	4	4	4	4	4
DRAM bits/chip target	8G	8G	16G	16G	32G	32G	32G
NAND Flash							
Flash � Pitch (nm) (un-contacted Poly)(F) (2D) [1]	15.0	15.0	15.0	15.0	15.0	15.0	15.0
Flash Product Highest Density (independent of 2D or 3D)	512G	1T	1T	1.5T	3T	4T	4T+
Flash 3D Maximum Number of Memory Layers [6]	64	96	128	192	384	512	>512

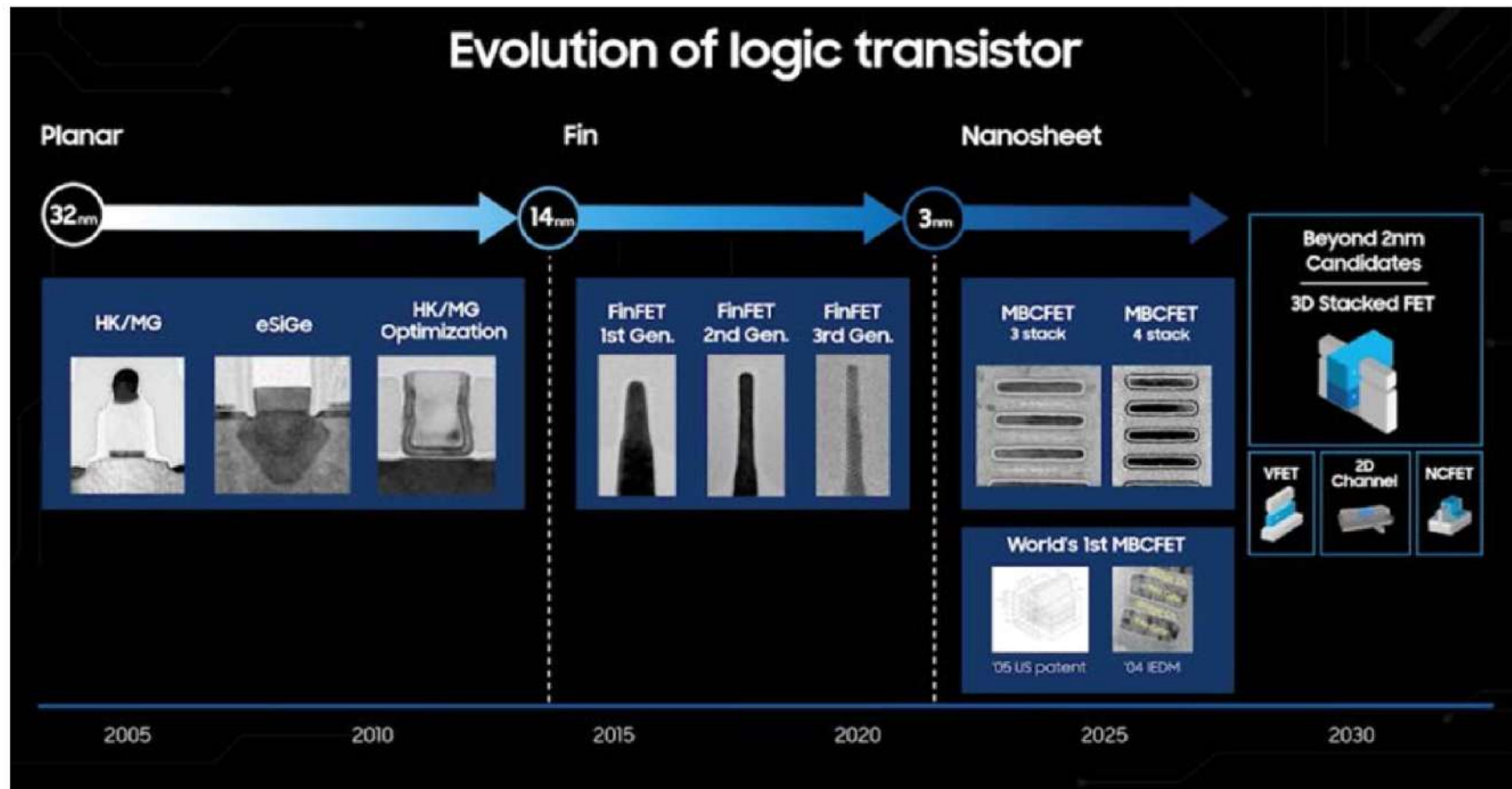


# Introduction : Pr vision d volution technologique



International Technology Roadmap for Semiconductors

[https://irds.ieee.org/images/files/pdf/2022/2022IRDS\\_ES.pdf](https://irds.ieee.org/images/files/pdf/2022/2022IRDS_ES.pdf)



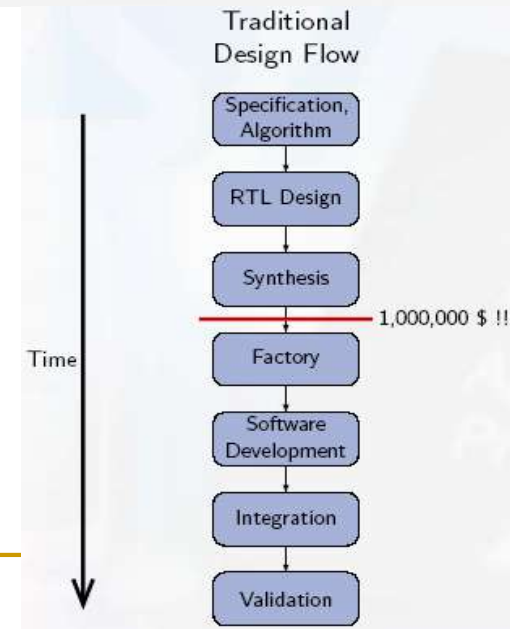
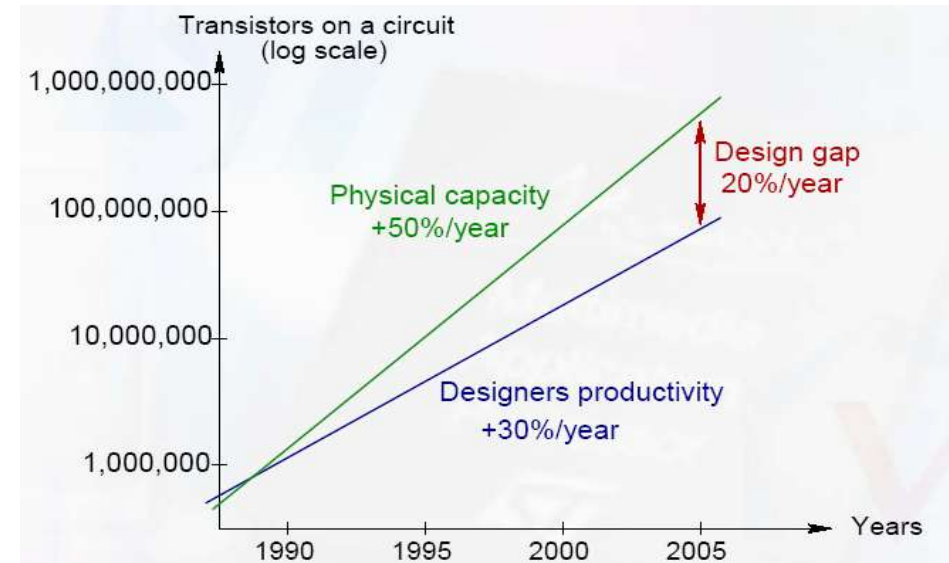
# Introduction : System On Chip

## ■ Un défi technologique

- Circuits sont de + en + complexe
  - Année 80 : Un circuit  $\Leftrightarrow$  10k transistors  $\Rightarrow$  100 homme/mois
  - Année 00 : Un circuit  $\Leftrightarrow$  500k transistors  $\Rightarrow$  30 000 h/mois

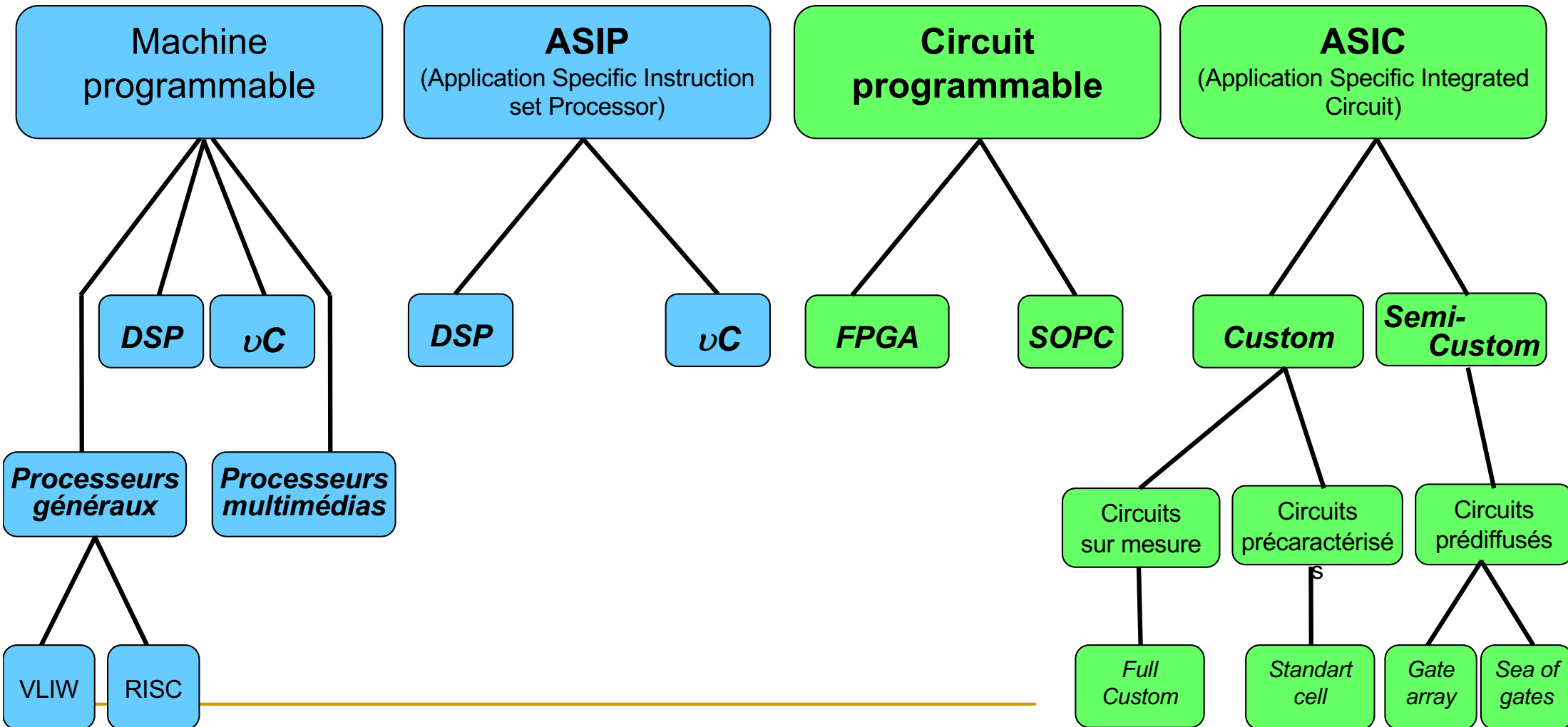
## ■ Un défi économique

- Réduction du « Time to Market »
  - Concurrence accrue
- Réduction du Prix
  - De 1M\$ à 300M\$



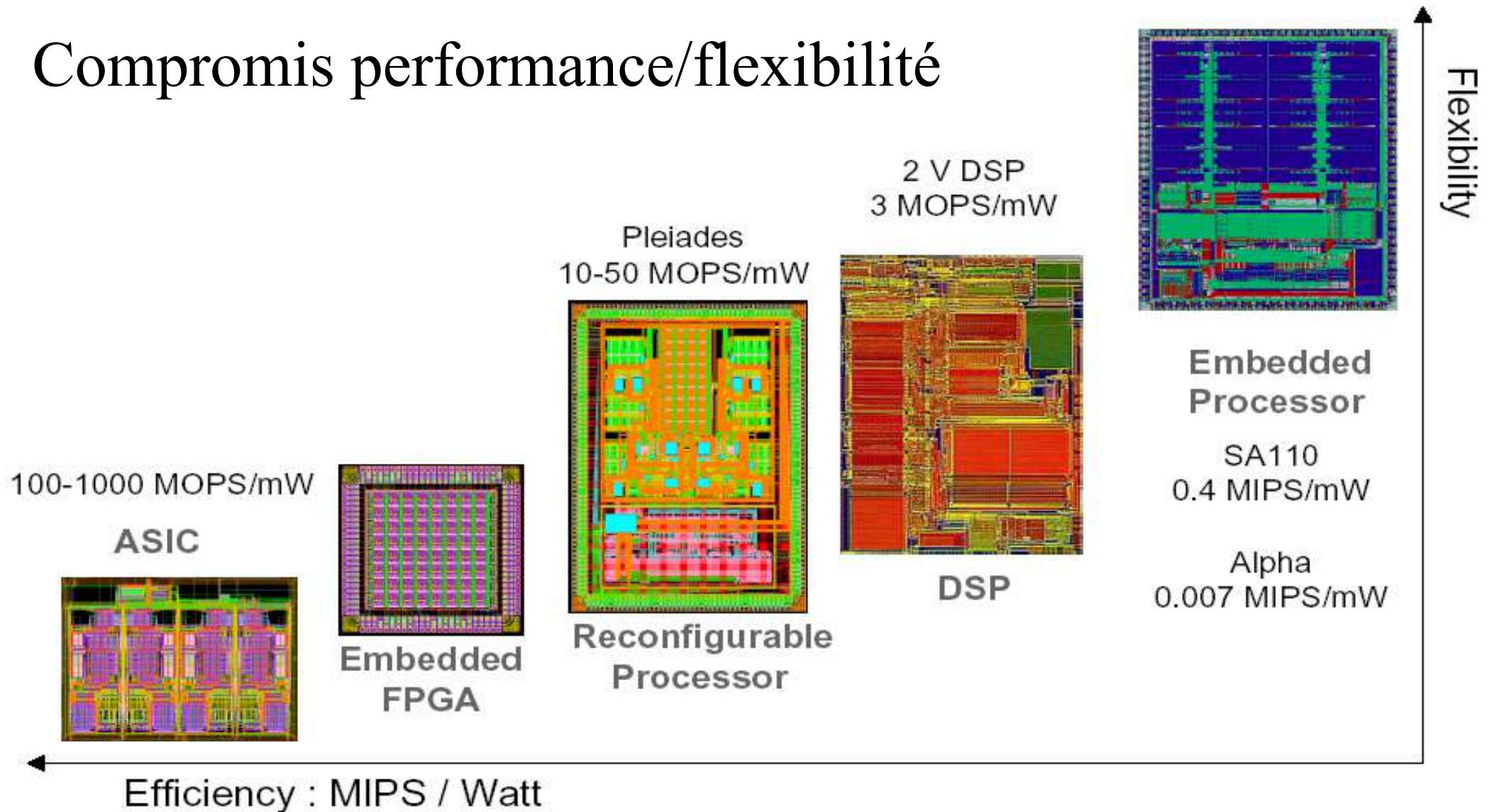
➡ *Repenser le mode de conception*

# Taxonomie des solutions matérielles



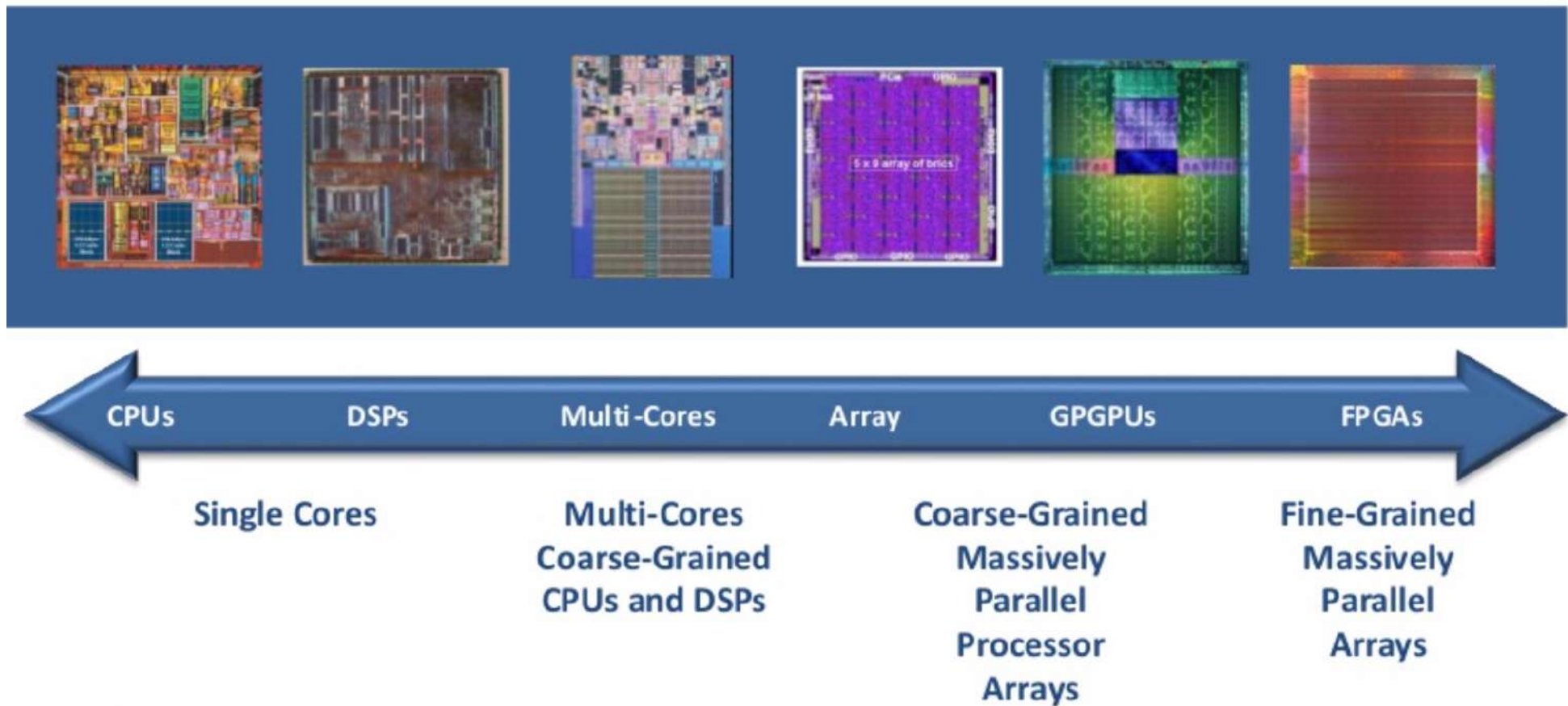
# Taxonomie des solutions matérielles

## Compromis performance/flexibilité



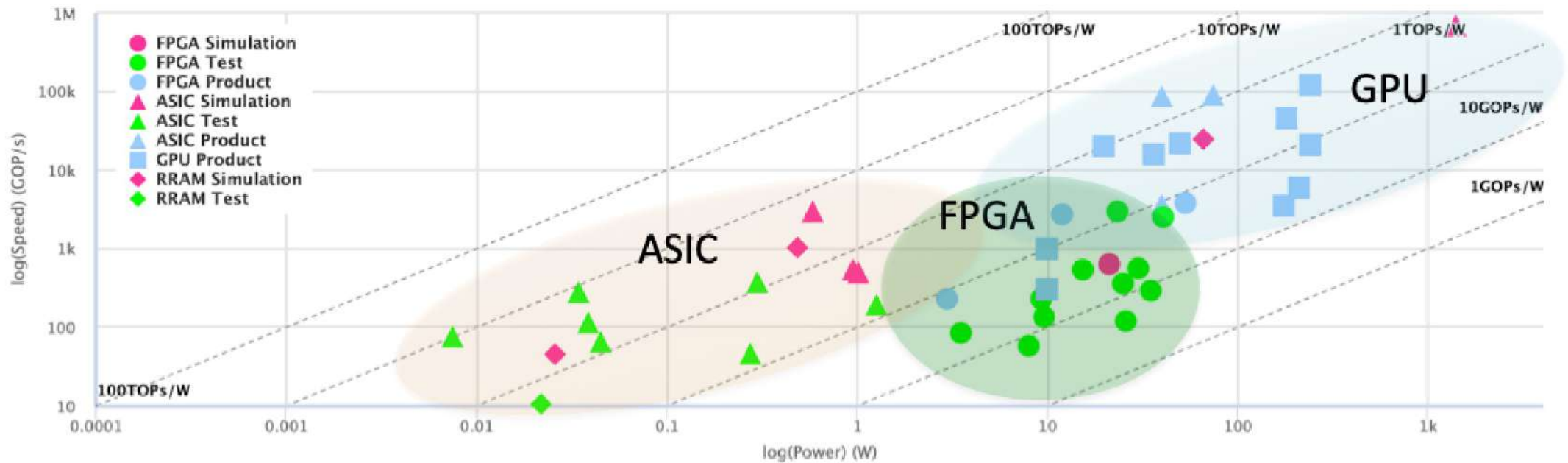


# Taxonomie des solutions matérielles : parallélisme multi-échelle



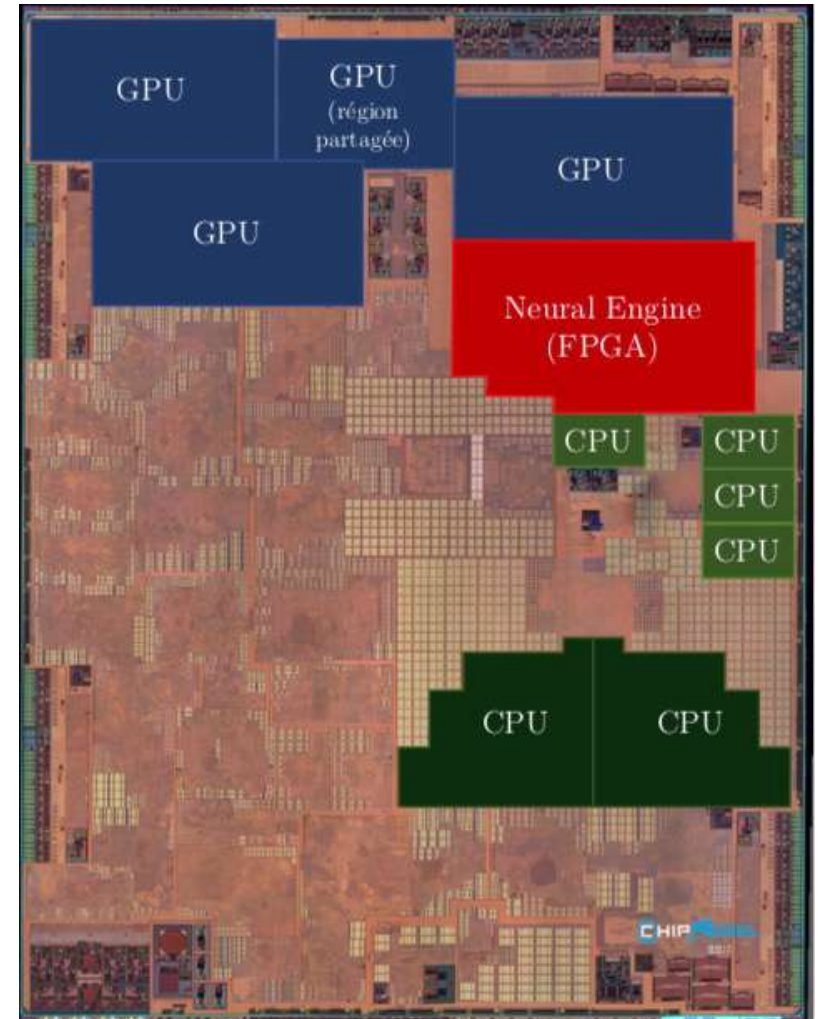


# Taxonomie des solutions matérielles



<https://nicsefc.ee.tsinghua.edu.cn/projects/neural-network-accelerator/>

- Parallélisme multi-échelle
- Circuits hétérogènes



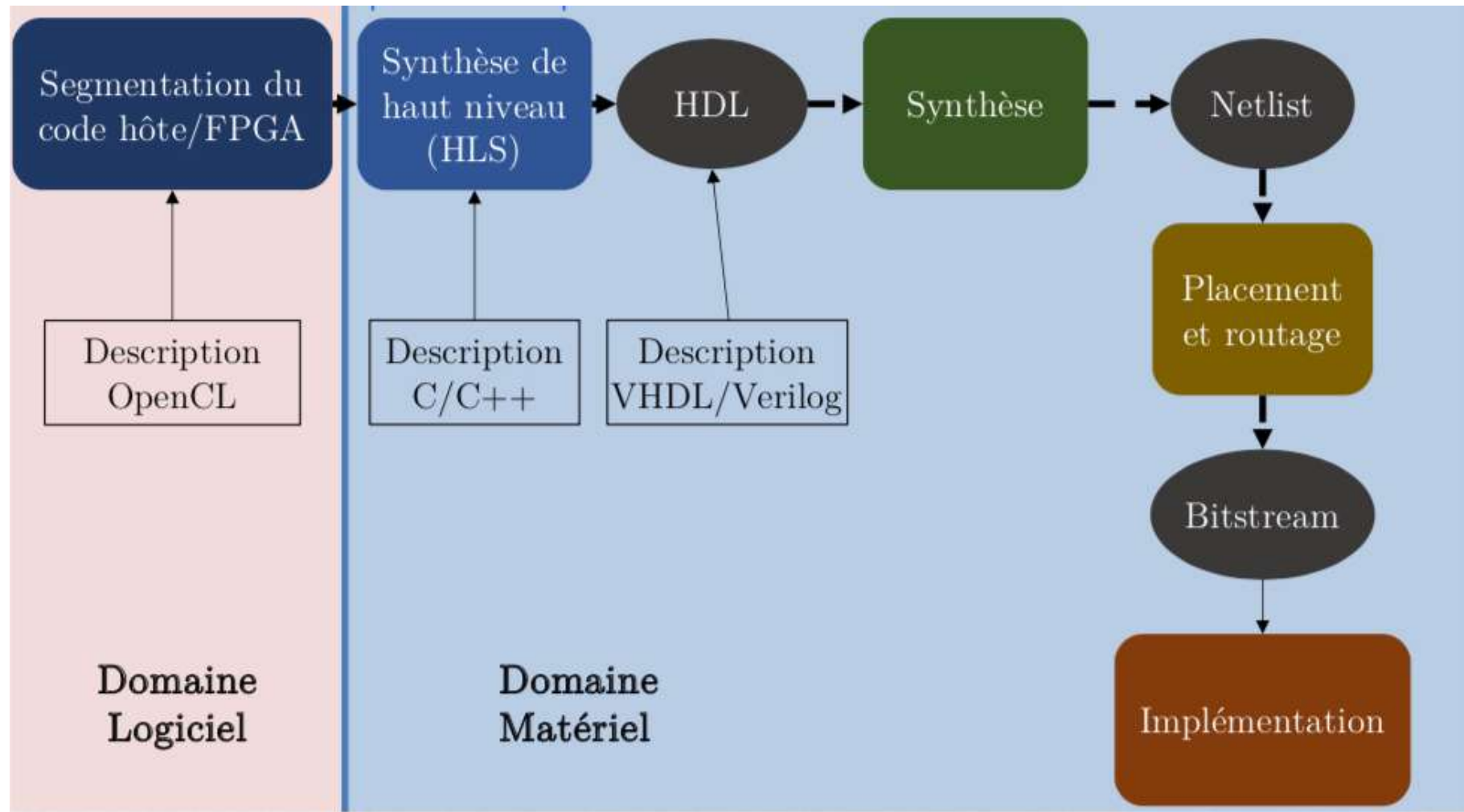
## Puce A11 d'Apple

# Comment exploiter le parallélisme des circuits FPGA ?

*Olivier Romain*  
Professeur des Universités  
[olivier.romain@u-cergy.fr](mailto:olivier.romain@u-cergy.fr)  
<http://olivieromain.free.fr>

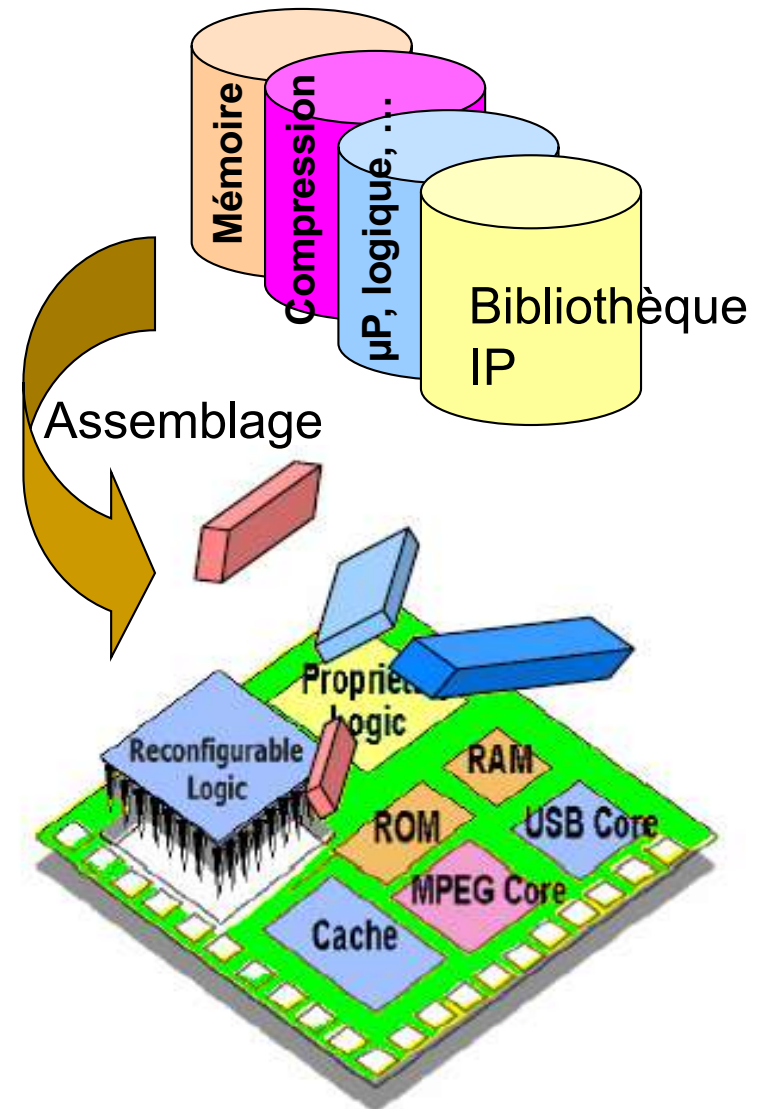


# Flot de conception



# Méthodologie à base IP

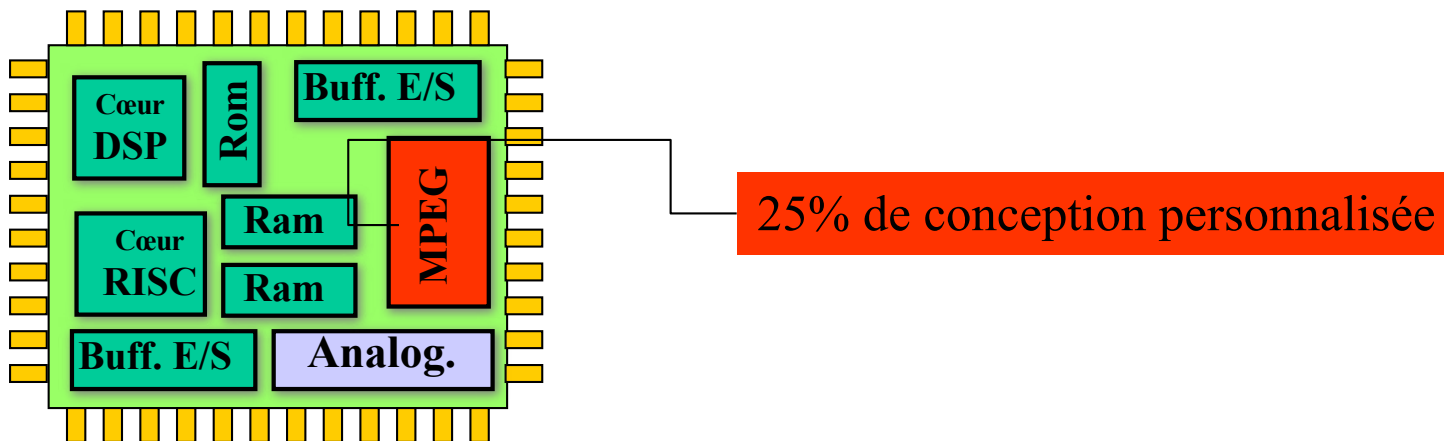
- Idées de la méthode
  - Assembler des Blocs de fonctions décrits HW ou SW
    - Bibliothèques constructeurs : Altera, Xilinx, Atmel
    - Bibliothèques libres : opencore, freecore, SocLib, etc...
  - Créer s'est propre bloc en vue de leur réutilisation
    - Bibliothèque personnelle





# Notion d'IP – Intellectual Property

- Réutiliser les blocs **déjà conçus** dans la société ;
- Utiliser les générateurs de **macro-cellules** (Ram, multiplieurs,...)
- **Acheter** des blocs conçus hors de l'entreprise.



# Introduction : Intellectual Property, IP

## ■ Définition

- Une IP est un composant virtuel décrivant un circuit réel.  
L'architecture développée est propre à chacun  $\Leftrightarrow$  IP.

## ■ Caractéristiques

- 2 types :
  - IP Soft :
    - Composant décrit en VHDL ou Verilog
    - Nécessite une optimisation pour une cible donnée
    - Paramétrable et synthétisable
  - IP Hard :
    - Description au niveau porte optimisée pour une cible donnée
- Interface standardisée et spécifiée
  - Bus VCI, AMBA, Avalon, etc ...

---

 *Prototypage rapide*

---

# Introduction : Intellectual Property

## ■ Avantages

- Mélange hétérogène d'IP
    - IP HW, SW
      - Conception conjointe, HW et SW
        - Optimiser un algorithme / cible
  - Design Reuse :
    - IP réutilisée un grand nombre de fois.
      - En respectant de certaines règles d'interconnexions
  - Prototypage rapide
    - Diminution du temps de conception
    - Plate forme Based Design
-

---

# Introduction : SOPC une alternative

## ■ SOPC (techno FPGA)

- ❑ System On Chip sur un composant programmable
- ❑ Orienté prototypage rapide
- ❑ Architecture flexible
- ❑ Composant reconfigurable
- ❑ Consommation + grande

## ■ SoC (techno ASIC)

- ❑ Répond aux critères de performances et d'intégration
  - ❑ Pas de flexibilité
    - Architecture figée
    - Evolution système réduite
  - ❑ Réservé aux grands volumes de production
  - ❑ Fabrication et test sont des étapes coûteuses
-

---

# Introduction : SOPC exemples

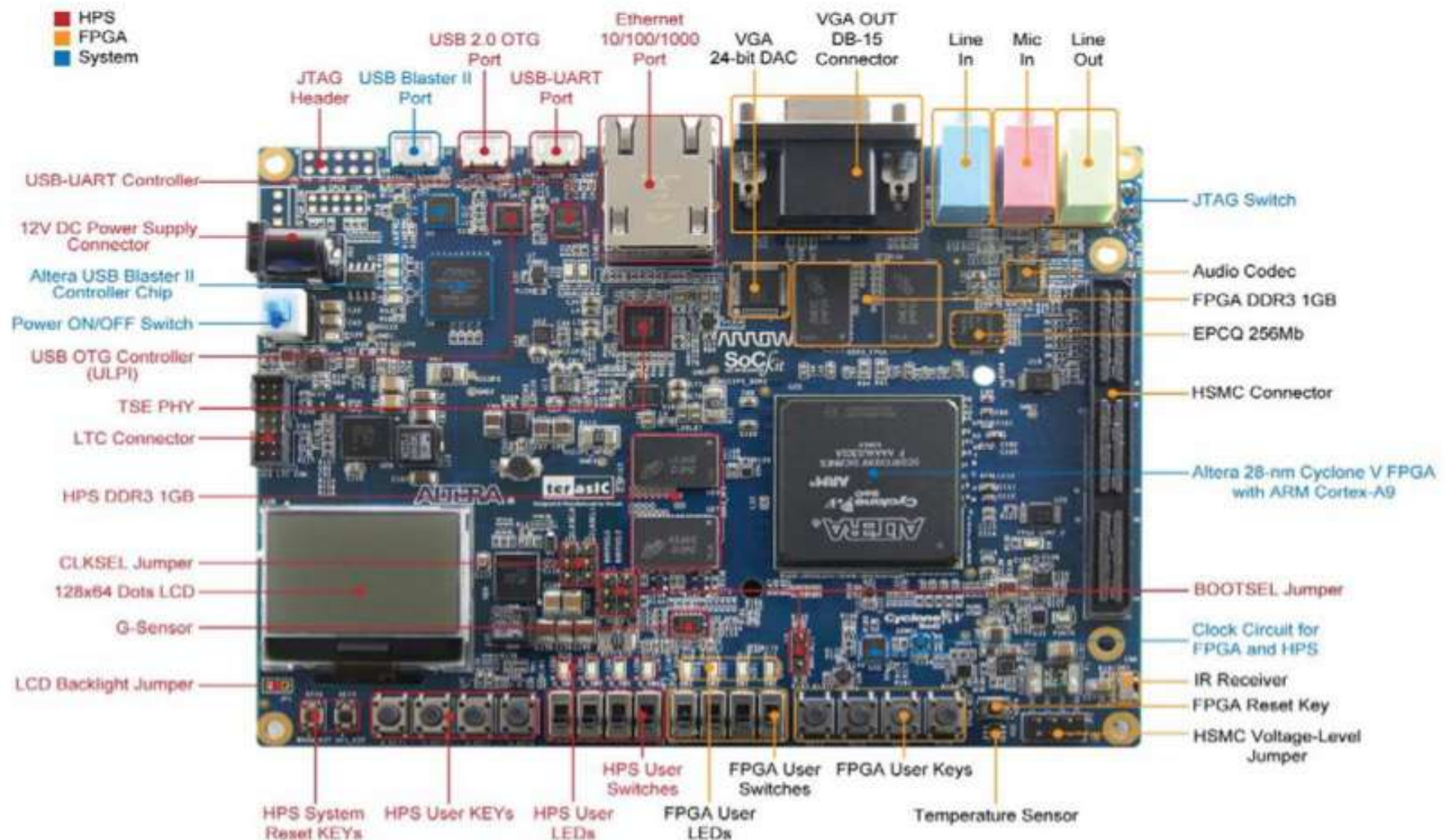
## ■ ALTERA

- ❑ Plusieurs kit de développement
  - ❑ Cœur de processeur embarqué dans un FPGA
    - ARM : IP Hardware – fin année 90
      - ❑ Processeur ARM922T, RISC 32 bits
      - ❑ Implanté en Hard dans un FPGA APEX 200k + passerelle zone programmable
      - ❑ 200 Mips
    - Nios II : IP logicielle – Softcore
      - ❑ Processeur décrit en VHDL : paramétrable
      - ❑ Dédié aux familles, APEX, Cyclone et Stratix
-



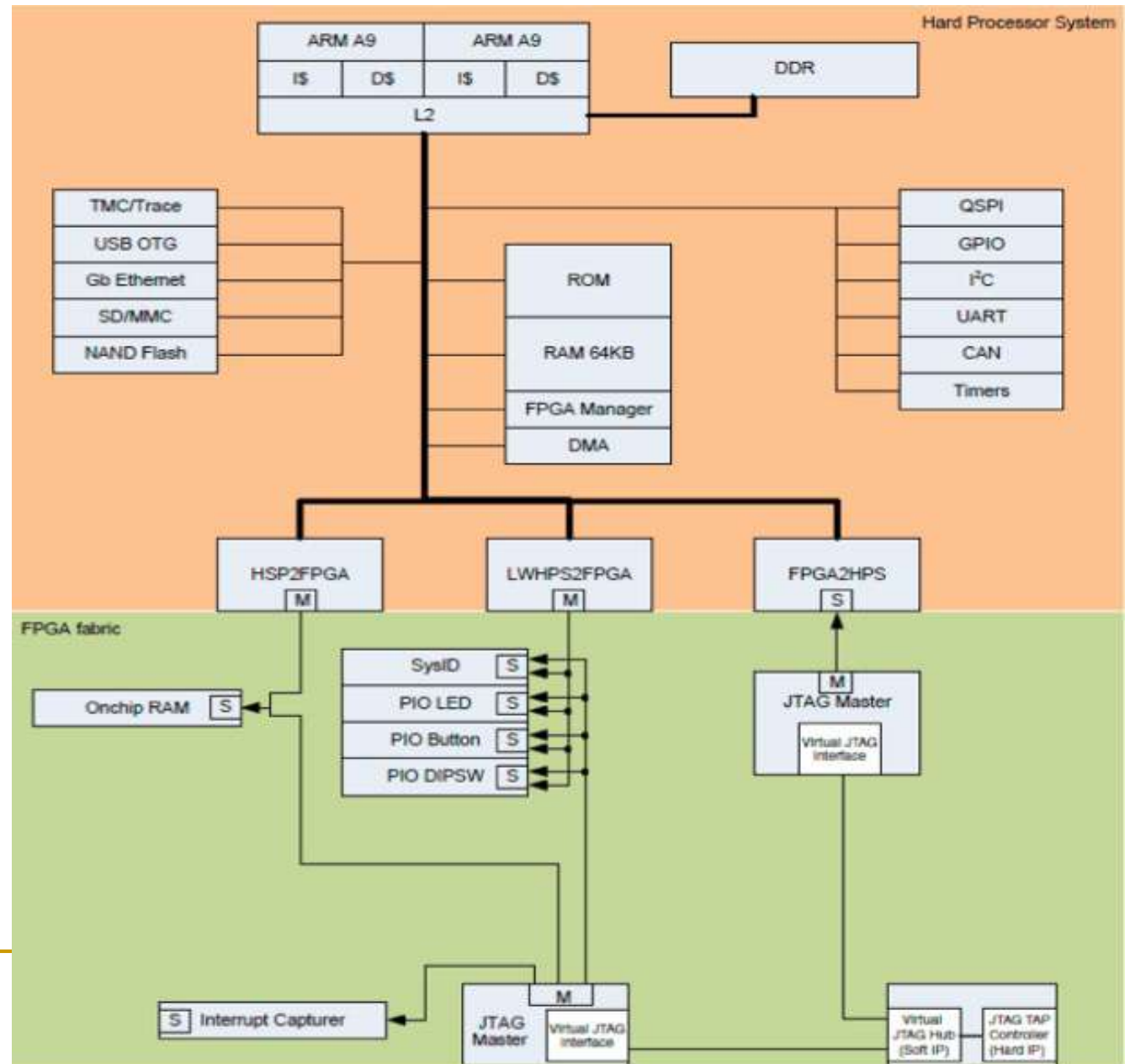
# Introduction : SOPC examples

- SoC-Kit Altera
  - Cortex A9

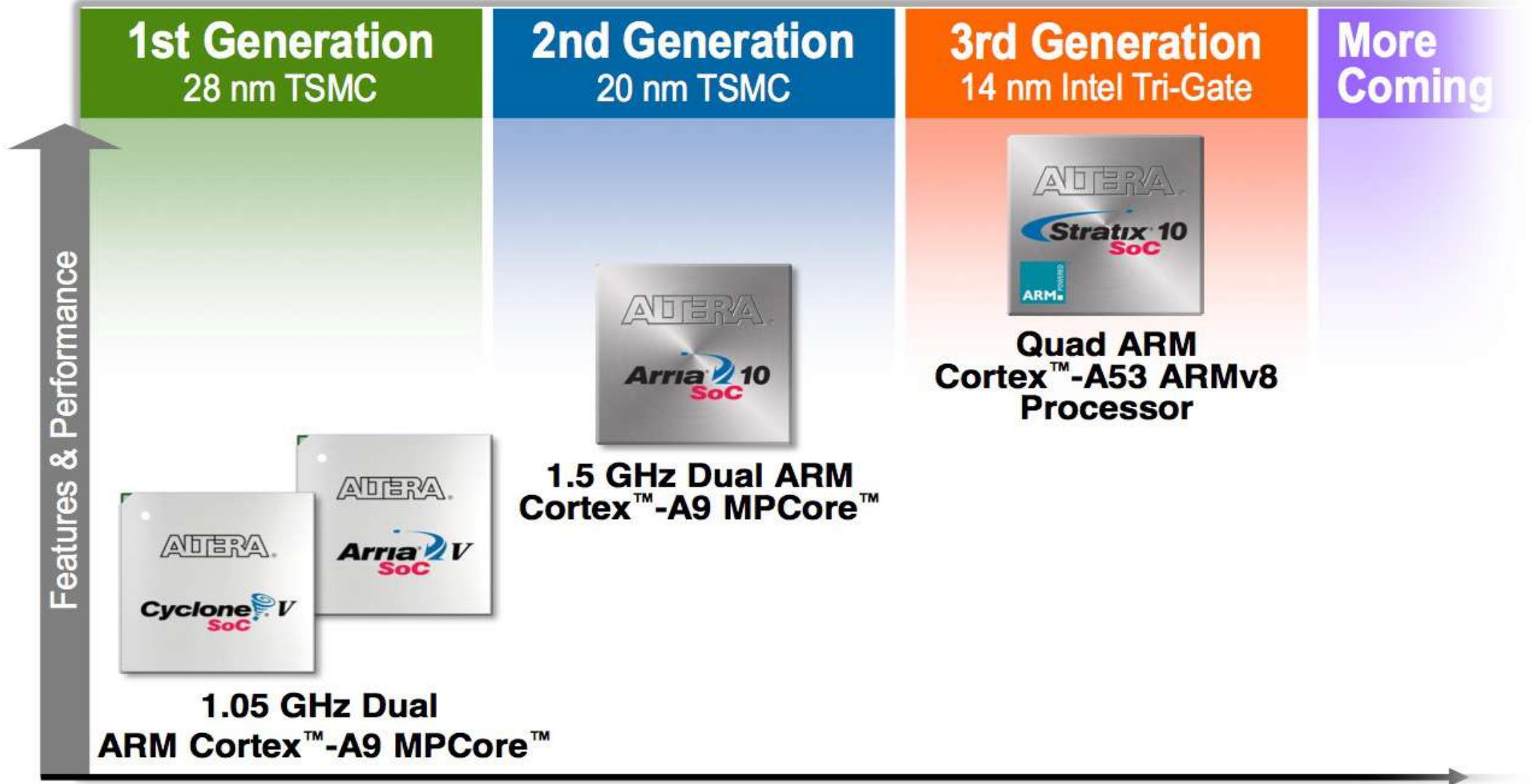


# Introduction : SOPC examples

- SoC-Kit Altera
  - Cortex A9



# Altera roadmap SoC



# Altera SoC Feature Comparison

Feature	Arria V SoC	Arria 10 SoC	Stratix 10 SoC
Process Technology	28 nm TSMC	20 nm TSMC	14 nm Intel Tri-Gate
Processor	Dual-core ARM Cortex-A9 MPCore	Dual-core ARM Cortex-A9 MPCore	Quad-core ARM Cortex-A53 MP Core
Maximum Processor Performance	1.05 GHz	1.5 GHz	1.5 GHz
Logic Core Performance	300 MHz	~500 MHz	1 GHz
Power Dissipation	1X	0.6X	0.3X
Logic Density Range	350 – 462K logic element (LE)	160 – 660K LE	500K LE - 5.5M LE
Embedded Memory	23 Mb	39 Mb	229 Mb
18 x 19 Multipliers	2,136	3,356	11,520
Maximum Transceivers	30	48	144
Maximum Transceiver Data Rate (Chip to Chip)	10 Gbps	17.4 Gbps	30 Gbps



## Derniers FPGA d'ALTERA – 2023

## Intel Agilex 7 FPGA and SoC M-Series Features

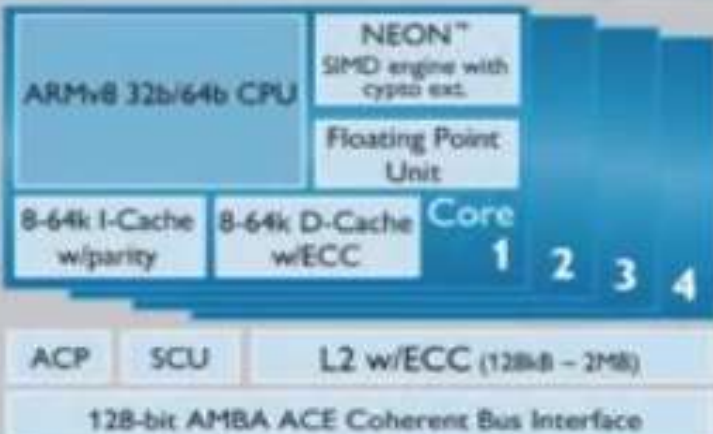
View device ordering codes on [page 55](#).

Product Line		AGM 032	AGM 039
Resources	Logic elements (LEs)	3,245,000	3,851,520
	Adaptive logic modules (ALMs)	1,100,000	1,305,600
	ALM registers	4,400,000	5,222,400
	M20K memory blocks	15,932	18,960
	M20K memory size (Mb)	311	370
	MLAB memory count	55,000	65,280
	MLAB memory size (Mb)	33	40
	High-bandwidth DRAM memory size (HBM2E) (Gigabytes)	16 / 32	16 / 32
	Fabric PLL	8	8
	I/O PLL	16	16
	Variable-precision digital signal processing (DSP) blocks	9,375	12,300
	18 x 19 multipliers	18,750	24,600
	Single-precision or half-precision tera floating point operations per second (TFLOPS)	14 / 28	18.4 / 37
	Maximum EMIF x72	4	4
Maximum Available Device Resources	Memory devices supported	LPDDR5, DDR5, DDR4, QDR IV	
	Maximum AIB interfaces	4	
	Secure Device Manager (SDM)	Provides SHA-384 bitstream integrity, ECDSA 256/384 bitstream authentication, AES-256 bitstream encryption, physically unclonable function (PUF) protected key storage, side-channel attack resistance, SPDMM attestation, cryptographic services, physical anti-tamper support	
	Hard processor system	Quad-core 64 bit Arm Cortex-A53 up to 1.41 GHz with 32KB I/D cache, NEON coprocessor, 1 MB L2 cache, direct memory access (DMA), system memory management unit, cache coherency unit, hard memory controllers, USB 2.0 x2, 1G EMAC x3, UART x2, SPI x4, I2C x5, general purpose timers x7, watchdog timer x4	



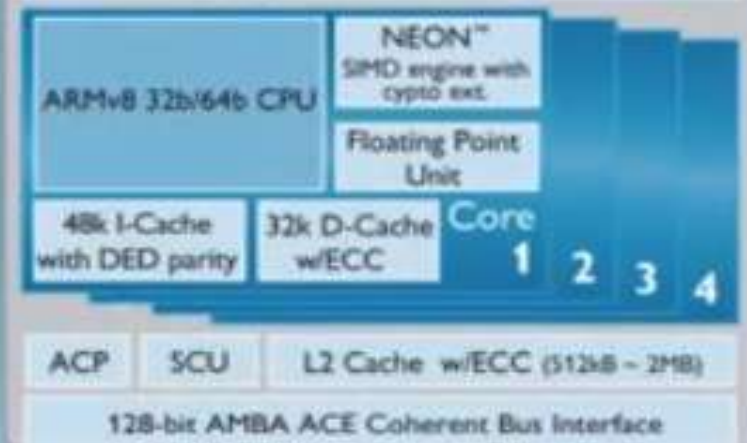
## Cortex™-A53

ARM CoreSight™ Multicore Debug and Trace



## Cortex™-A57

ARM CoreSight™ Multicore Debug and Trace



# CO-Design

---

En quoi ça consiste ?

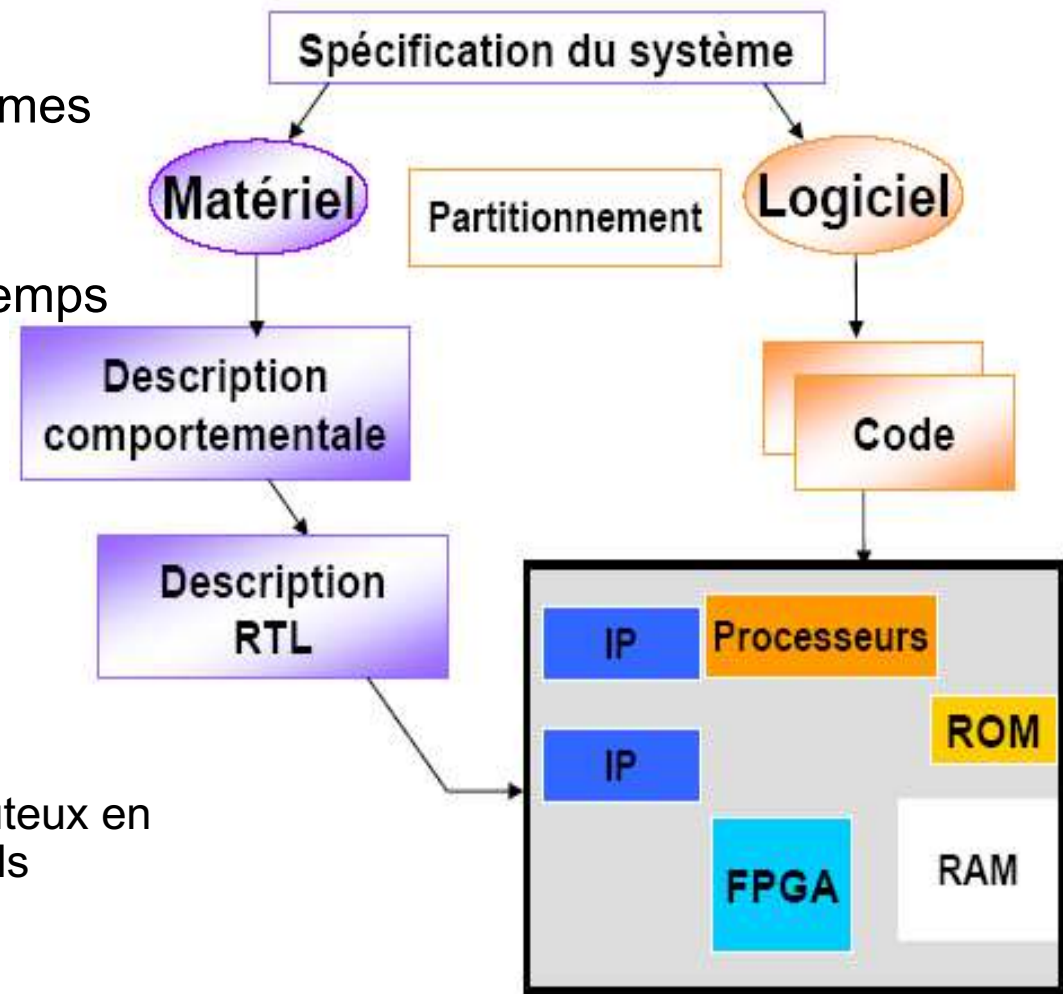
# Co-Design : qu'est ce que c'est ?

## ■ Objectif :

- ❑ Accélérer les traitements des algorithmes coûteux en temps de calcul
- ❑ Contexte d'application embarquées temps réels

## ■ Comment ?

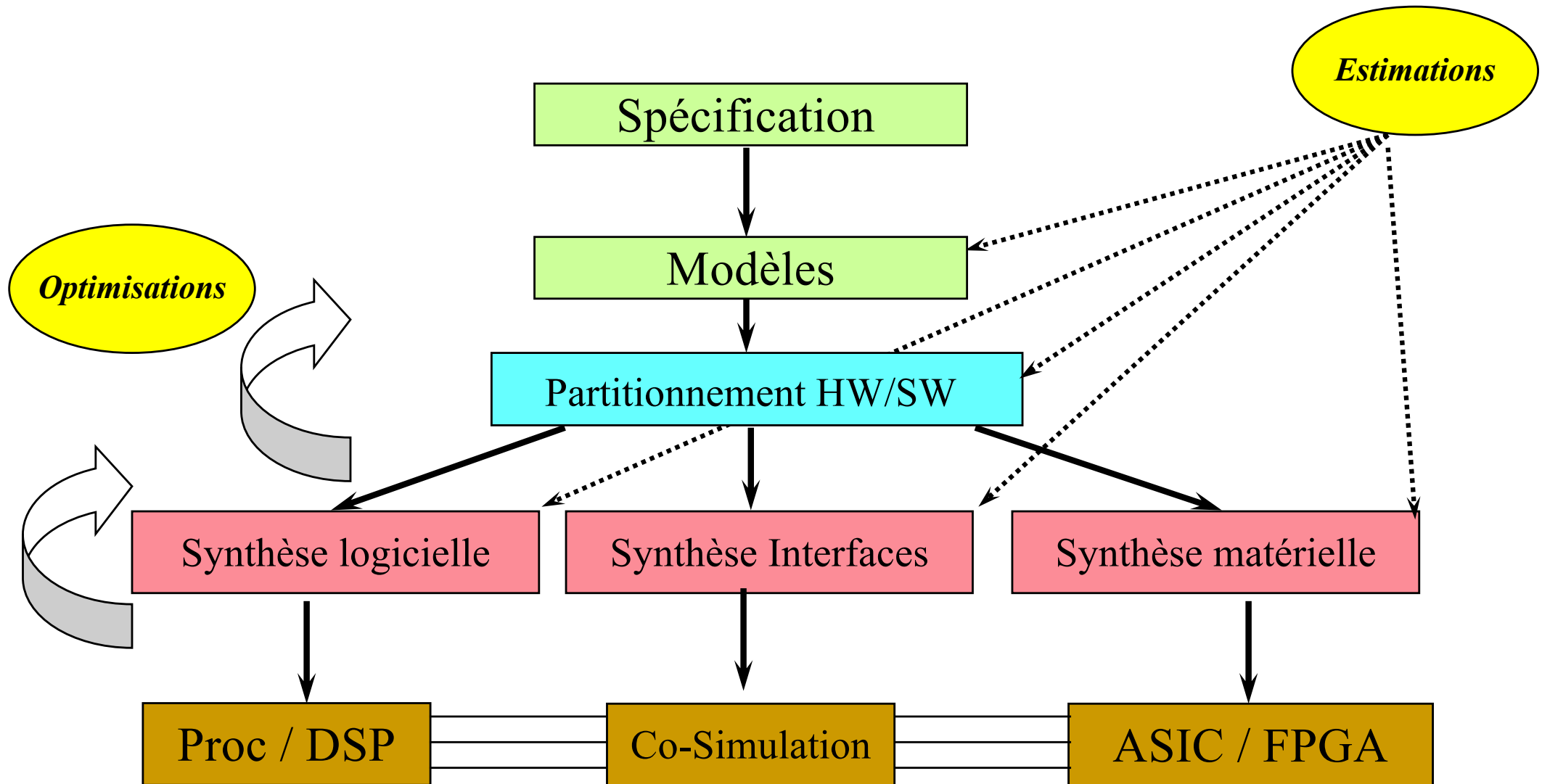
- ❑ partitionner une application
  - Matérielle et Logicielle
- Remplacement des blocs logiciels coûteux en temps de calcul par des blocs matériels réalisant la même fonction.



# Co-Design : qu'est ce que c'est ?

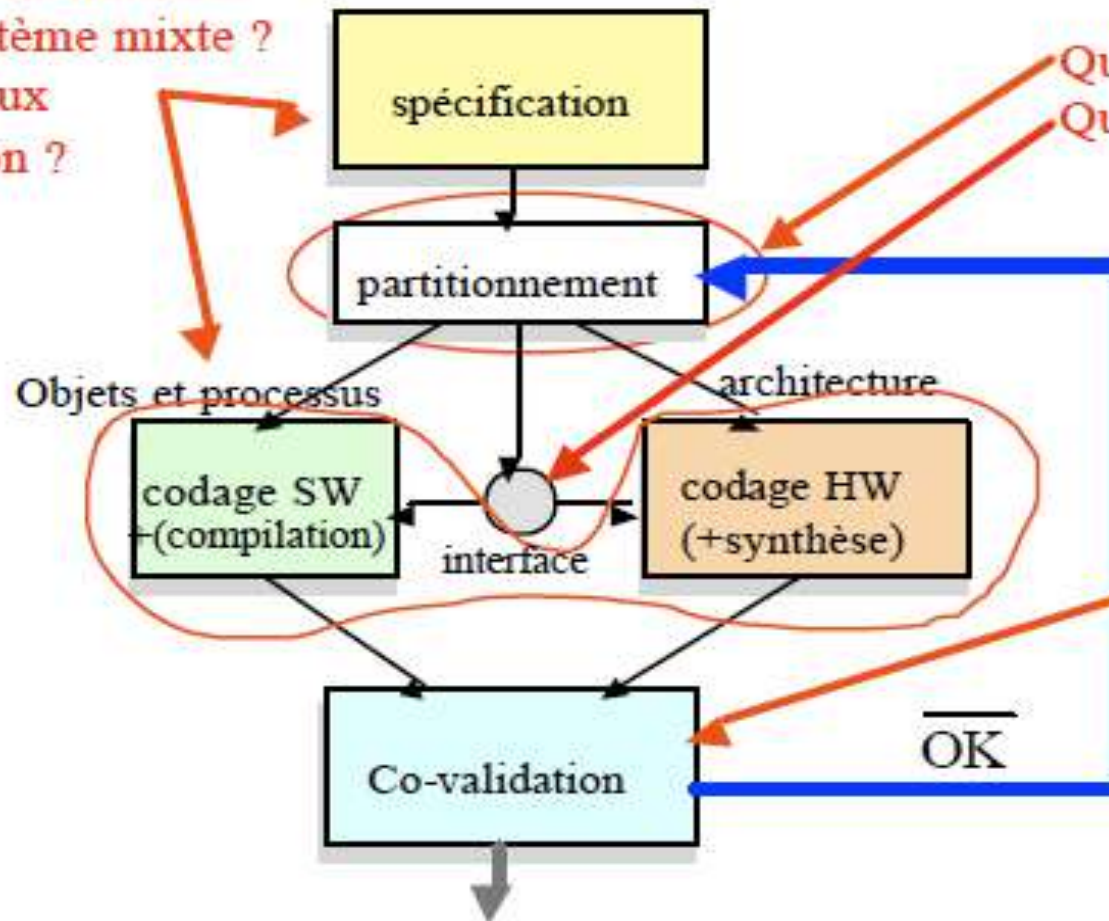
- Le Co-Design désigne une méthodologie de conception concurrente matérielle et logicielle d'un système électronique complexe
- *Méthodologie de conception en 4 étapes*
  - *Partitionnement : Décider quelles tâches sont effectuées en SW et quelles autres sont effectuées en HW et comment se fait l'interface entre HW et SW*
  - *Modélisation : Coder la spécification de façon à représenter du mieux possible la fonctionnalité du système et ses contraintes.*
  - *Co-Validation : valider simultanément les parties matérielles et logicielles. Valider la fonction, la cohérence temporelle des tâches, analyser le respect des contraintes et préparer la synthèse*
  - *Co-synthèse : Effectuer la compilation (SW) et la synthèse logique (HW).*

# Le flot de conception CoDesign



# Le flot de conception CoDesign : les questions qui se posent

Quels types de modèles  
dans un système mixte ?  
Quels niveaux  
d'abstraction ?



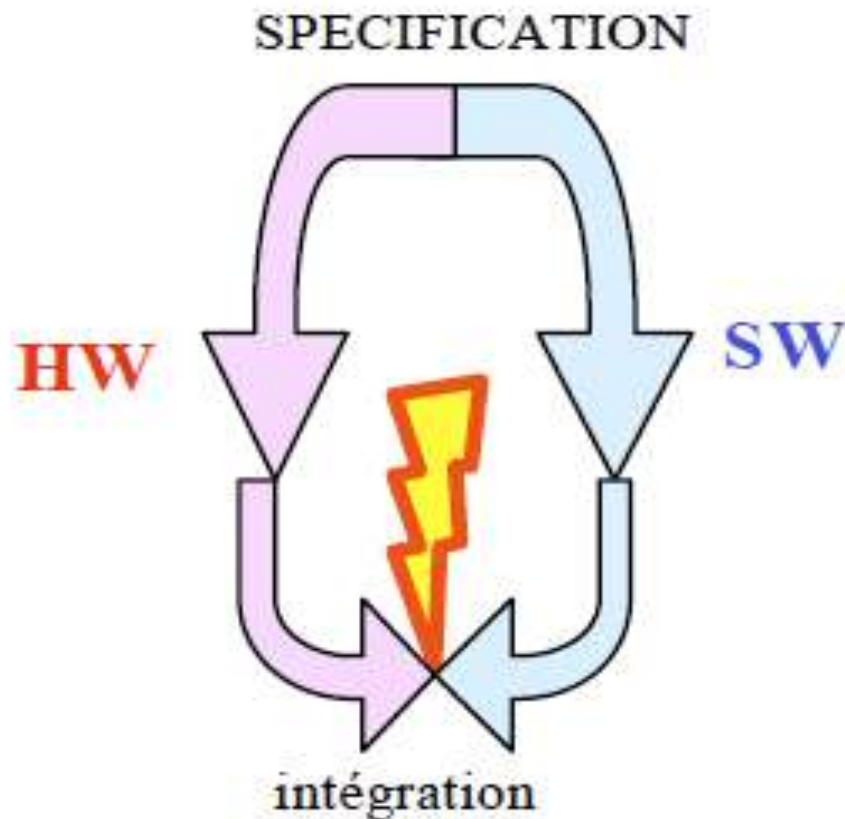
Quel Partitionnement optimal ?  
Quelle interface ?

Comment valider un  
système hétérogène ?

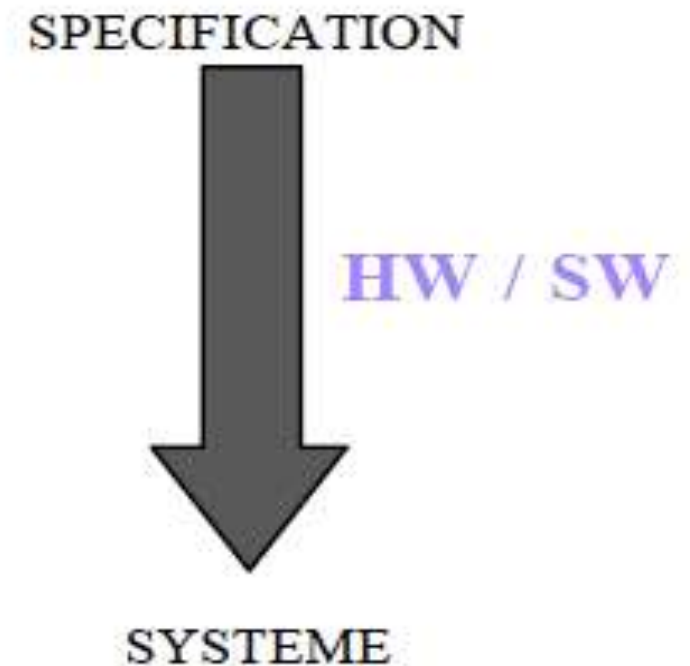


# Le flot de conception CoDesign : Les types de partitionnement

## ■ Manuel

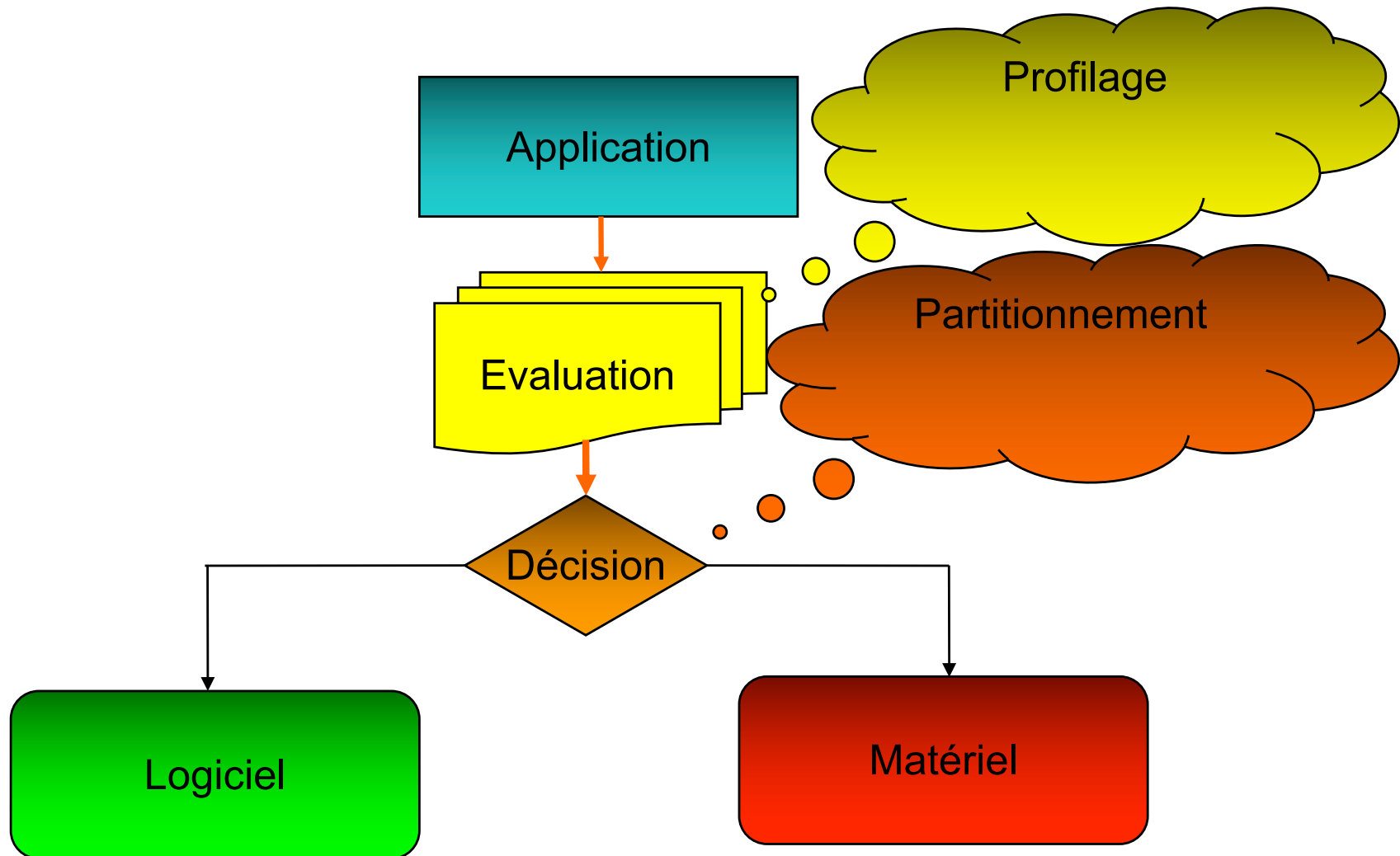


## ■ Automatique

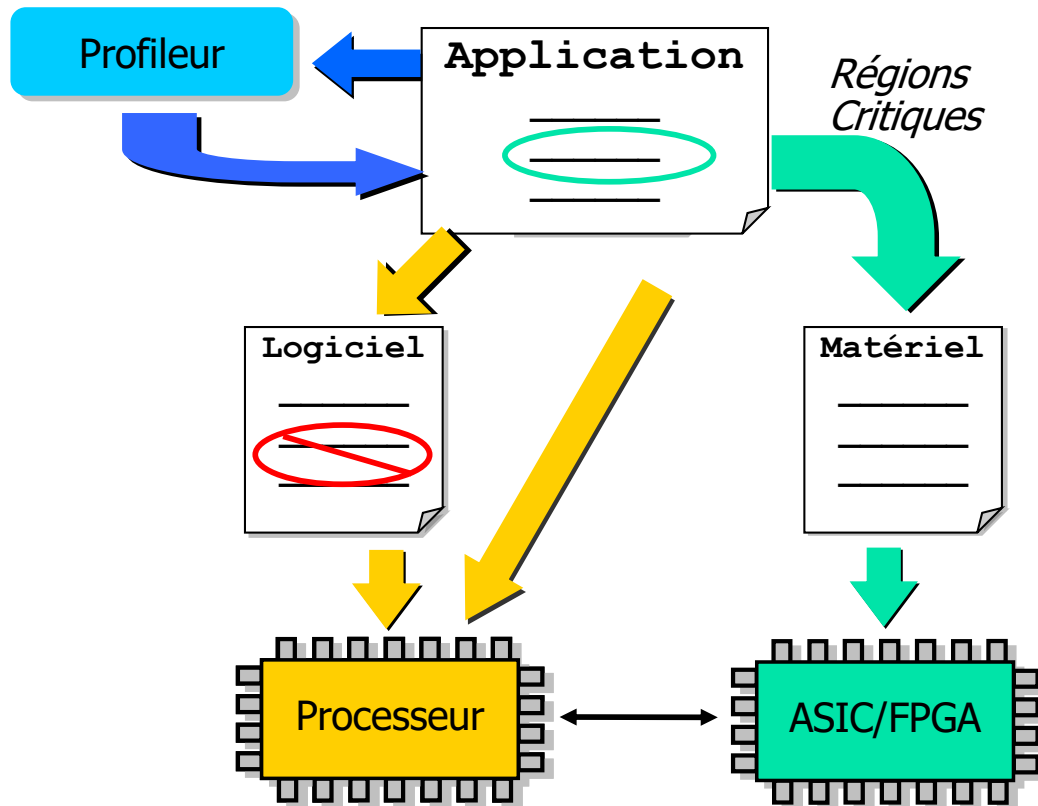


Il n'y a plus qu'une équipe HW/SW

# Une approche simple

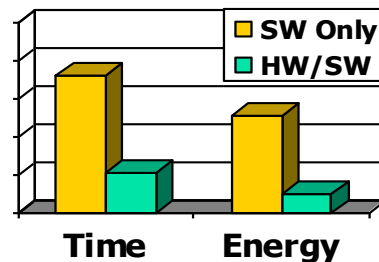


# Profilage et Partitionnement



## Bénéfices

- Accélération de 10 à 200 fois
- Accélération possible de 800 fois
- Beaucoup plus de potentiel que les optimisations dynamiques logicielles (internes au processeur, déroulage de boucle, pipeline logiciel,...)
- Réduction de la consommation d'énergie de 25 à 95%



---

# Profilage

- Le Profilage permet d'apprendre les endroits, en terme de code, où le programme passe son temps. Quelle fonction appelle quelle autre durant son exécution.
  - Le profilage s'effectue via des données collectée lors de l'exécution de l'application. Cette méthode peut donc être utilisée pour analyser des programmes trop complexe pour une analyse via la lecture des sources.
  - Ces informations de profil, montre les bouts de code où le programme est plus lent qu'attendu.
  - Ces bouts de code sont de bons candidats à :
    - **une réécriture optimisées**
    - **une transformation matérielle**
-

---

# Comment réaliser un profilage ?

- **Echantillonnage** : interruption périodique de l'OS ou lecture des compteurs matériels du processeur
  - **Instrumentation**: insertion de code espion pour mesurer
-

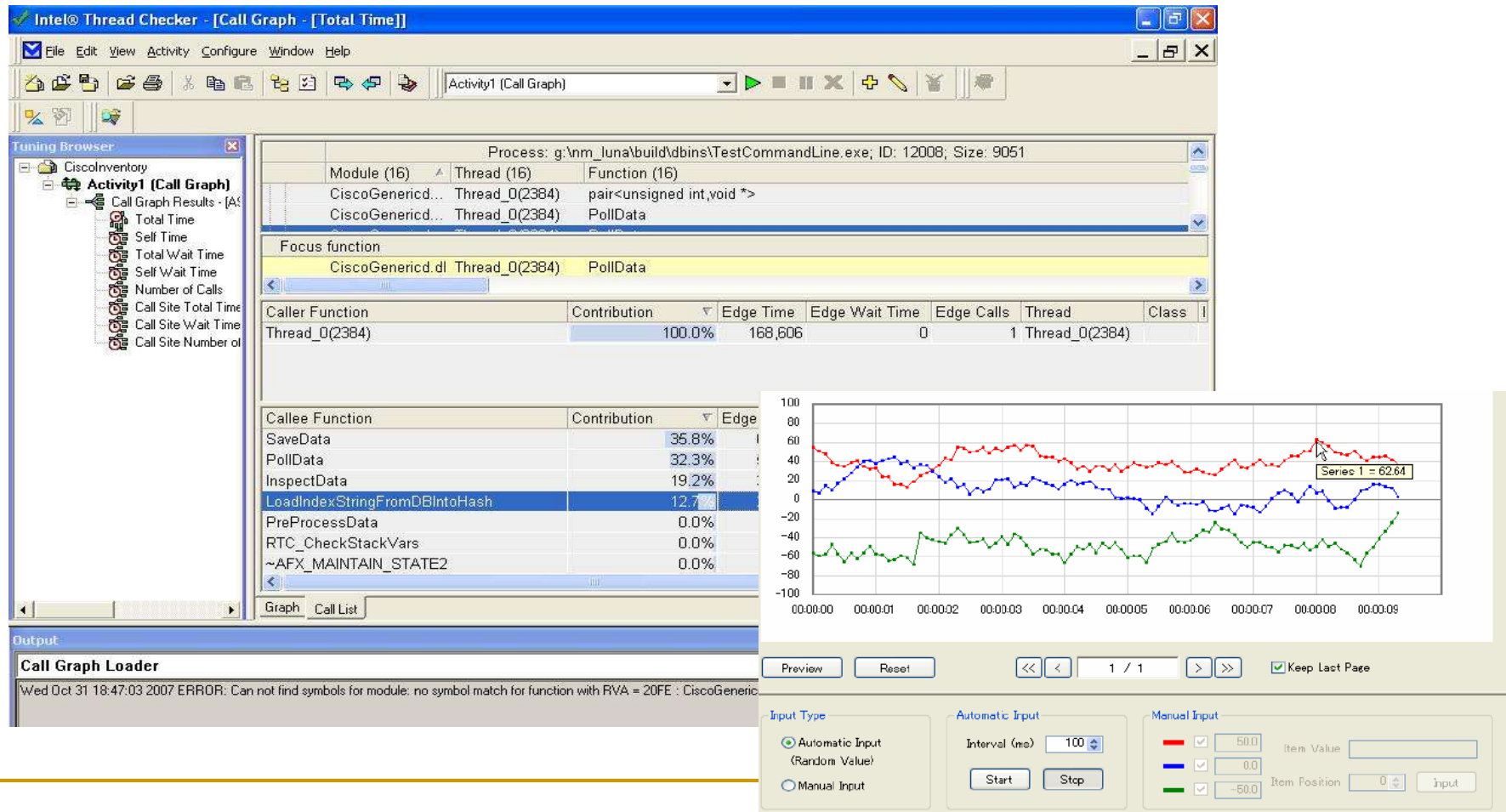
# Echantillonage / Instrumentation

	Echantillonage	Instrumentation
Overhead	Typically about 1%	High, may be 500% !
System-wide profiling	Yes, profiles all app, drivers, OS functions	Just application and instrumented DLLs
Detect unexpected events	Yes , can detect other programs using OS resources	No
Setup	None	Automatic ins. of data collection stubs required
Data collected	Counters, processor an OS state	Call graph , call times, critical path
Data granularity	Assembly level instr., with src line	Functions, sometimes statements
Detects algorithmic issues	No, Limited to processes , threads	Yes – can see algorithm, call path is expensive



# Exemple d'outil de profilage

- Visual C++ Performance Profiler (Microsoft)



# Evaluation et Comparaison

			Mem & Perf Validator			V-Tune			VS Perf. Analyzer			ValGrind			Google Profiler		
Platforms																	
Linux	Win32	X64	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Line level code			Yes			Yes			Yes			Yes			Yes		
Graphical o/p			Yes			Yes			Yes			No			No		
Diagnostic			Yes			Yes			No			No			No		
Strongness			<ul style="list-style-type: none"><li>Interesting graphical environment</li><li>No need to instrument code</li></ul>			<ul style="list-style-type: none"><li>quickly identify critical functions</li><li>Helps identify system level perf. Issues</li><li>No need to instrument code</li><li>Important diagnostic feature</li></ul>			<ul style="list-style-type: none"><li>Easy to use</li><li>Visual studio</li></ul>			<ul style="list-style-type: none"><li>Strong in memory profiling</li><li>all reads and writes of memory are checked</li><li>Easy to use</li><li>Freeware</li></ul>			<ul style="list-style-type: none"><li>Freeware</li><li>Increase cache efficiency</li></ul>		
Weakness			<ul style="list-style-type: none"><li>Weak performance Profiling</li></ul>			<ul style="list-style-type: none"><li>Not Stable for X64 PF</li><li>Monitors all active software on system</li></ul>			<ul style="list-style-type: none"><li>Is Not a stand alone profiler</li><li>Very expensive</li><li>No memory profiling</li></ul>			<ul style="list-style-type: none"><li>Weak performance profiling</li><li>No windows PF</li></ul>			<ul style="list-style-type: none"><li>Old</li><li>Buggy</li><li>Inaccurate</li></ul>		
Notes & Users Evaluation			Weak perf. profiling						Is Not a stand alone profiler			Massif in Heap Profiling			Old, buggy, Inaccurate		
Price (€)			407			695			9470			Freeware			Freeware		

# Exemple de profilage via gprof

- Avec gcc, il faut tout d'abord compiler et lier le programme avec les options de profilage autorisées :
  - ▣ ***gcc -o myprog.exe myprog.c utils.c -g -pg***
- Il faut ensuite exécuter le programme pour collecter les données du profil d'exécution
  - ▣ **Le programme écrit les données collectées dans un fichier *'gmon.out'* juste avant de finir.**
- Il est possible après d'utiliser gprof pour analyser les données collectées :
  - ▣ ***gprof options myprog.exe gmon.out > outfile***
  - ▣ ***gprof* crée un fichier de profil et un graphe d'exécution**

# Exemple de profilage via gprof

Each sample counts as 0.01 seconds.

% time	cumulative seconds	self seconds	calls	self ms/call	total ms/call	name
33.34	0.02	0.02	7208	0.00	0.00	open
16.67	0.03	0.01	244	0.04	0.12	offtime
16.67	0.04	0.01	8	1.25	1.25	memccpy
16.67	0.05	0.01	7	1.43	1.43	write
16.67	0.06	0.01				mcount
0.00	0.06	0.00	236	0.00	0.00	tzset

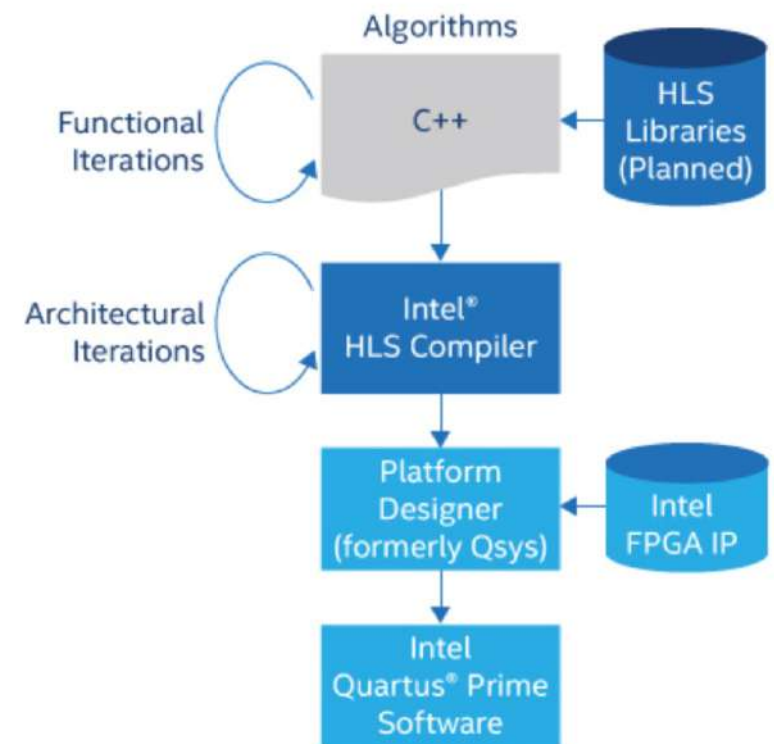
- *% time* : pourcentage du temps total d'exécution que le program a passer dans cette fonction.
- *cumulative seconds*: Temps cumulatif en seconde que le processeur a passé a exécuter cette fonction ainsi que toutes les fonctions appelées dans cette fonction.
- *self seconds*: Temps en secondes utilisé pour cette seule fonction.
- *calls*: Nombre de fois total où cette fonction a été appelée.
- *self ms/call*: Temps moyen en milliseconde pris par chaque appel de la fonction.
- *total ms/call*: Temps moyen en milliseconde pris par chaque appel de la fonction et de ses descendants.
- *name*: Nom de la fonction.

# Faiblesse de cette première approche

- I. Certaines fonctions ne sont pas triviales à réaliser en matériel.
- II. Les décisions prises trop tôt dans le flot risque de ne pas être optimales
- III. Aucune considération pour la communication et l'interfaçage.
- IV. Si l'application change alors il faut ré-exécuter un profilage et ensuite un partitionnement.

# Le flot de conception CoDesign : outils de haut niveau : HLS

- HLS Vivado : Xilinx
- Intel HLS compiler : Altera
- Catapult C
- ...
- Outils académiques
  - ❑ Gaut 3.0 : <http://www.gaut.fr>
    - <https://www.youtube.com/watch?v=AWkxQHgUTSM>





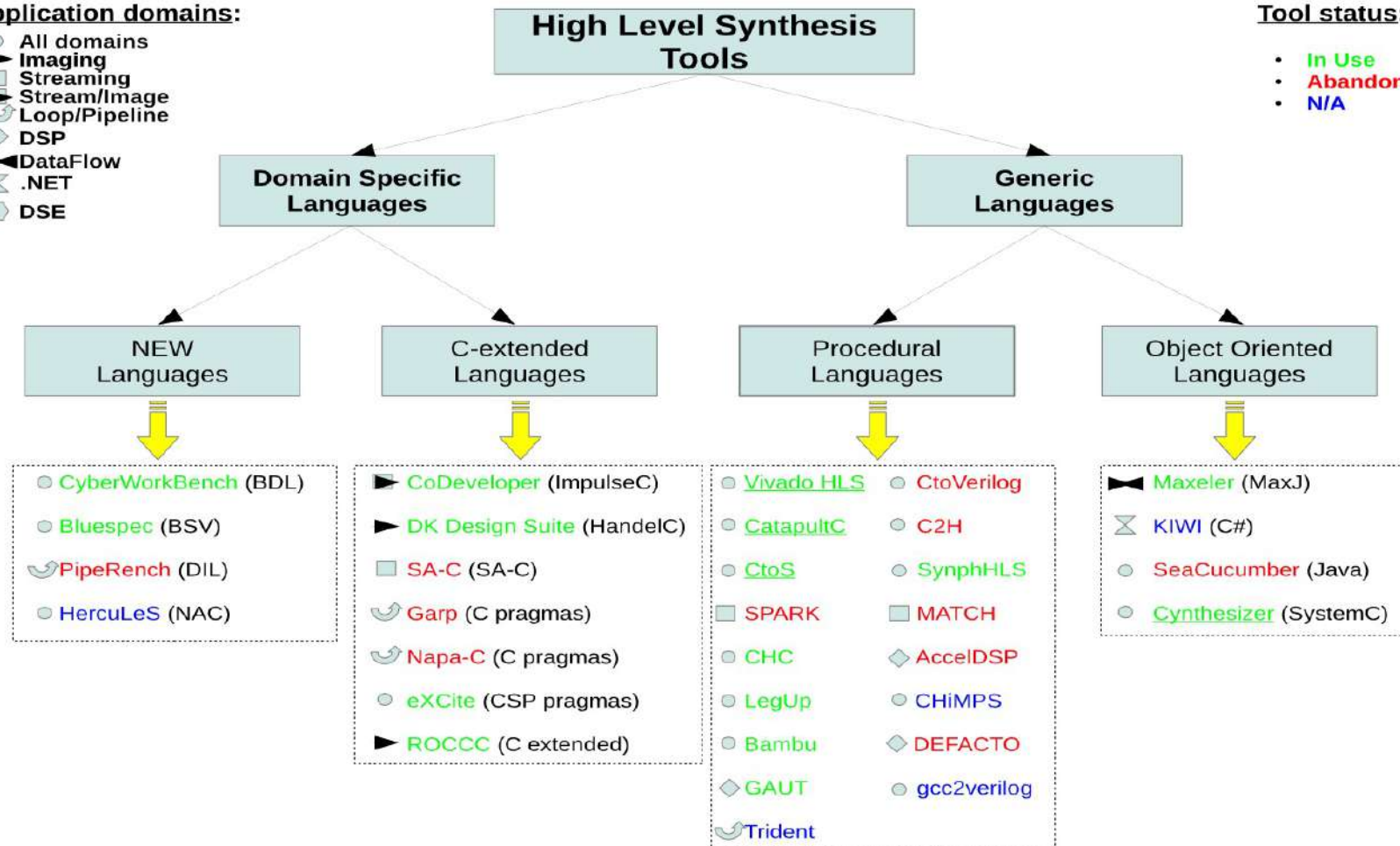
# Outils de haut niveau : HLS

## Application domains:

- All domains
- ▶ Imaging
- ▣ Streaming
- ▶ Stream/Image
- ▶ Loop/Pipeline
- ◇ DSP
- ▶ DataFlow
- ⌂ .NET
- ⬢ DSE

## Tool status:

- In Use
- Abandoned
- N/A



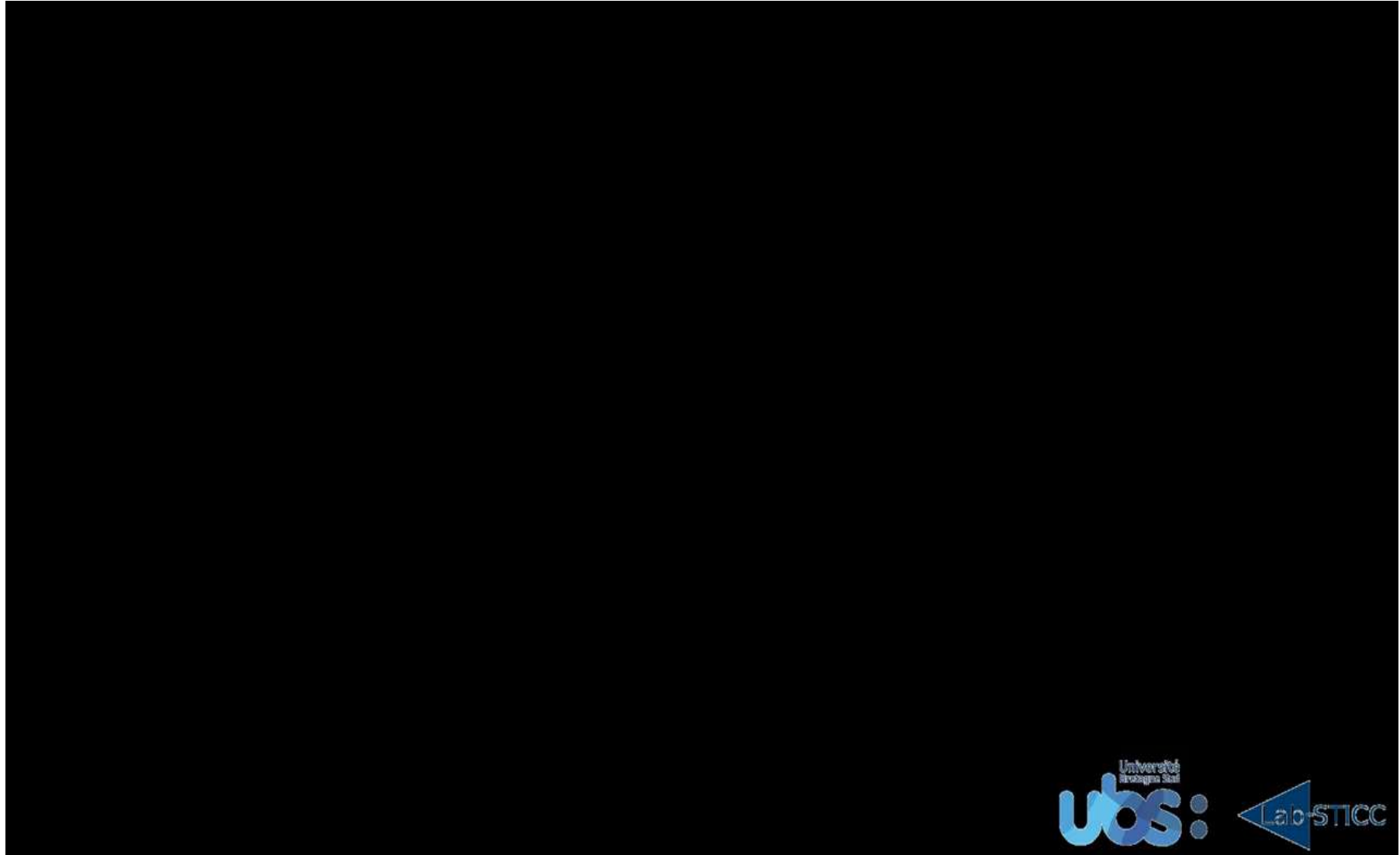
# Outils de haut niveau : HLS

Status	Compiler	Owner	License	Input	Output	Year	Domain	TestBench	FP	FixP
In Use	eXCite	Y Explorations	Commercial	C	VHDL/Verilog	2001	All	Yes	No	Yes
	CoDeve- loper	Impulse Accelerated	Commercial	Impulse-C	VHDL Verilog	2003	Image Streaming	Yes	Yes	No
	Catapult-C	Calypso Design Systems	Commercial	C/C++ SystemC	VHDL/Verilog SystemC	2004	All	Yes	No	Yes
	Cynthesizer	FORTE	Commercial	SystemC	Verilog	2004	All	Yes	Yes	Yes
	Bluespec	BlueSpec Inc.	Commercial	BSV	SystemVerilog	2007	All	No	No	No
	CHC	Altium	Commercial	C subset	VHDL/Verilog	2008	All	No	Yes	Yes
	CtoS	Cadence	Commercial	SystemC TLM/C++	Verilog SystemC	2008	All	Only cycle accurate	No	Yes
	DK Design Suite	Mentor Graphics	Commercial	Handel-C	VHDL Verilog	2009	Streaming	No	No	Yes
	GAUT	U. Bretagne	Academic	C/C++	VHDL	2010	DSP	Yes	No	Yes
	MaxCompiler	Maxeler	Commercial	MaxJ	RTL	2010	DataFlow	No	Yes	No
	ROCCC	Jacquard Comp.	Commercial	C subset	VHDL	2010	Streaming	No	Yes	No
	Symphony C	Synopsys	Commercial	C/C++	VHDL/Verilog SystemC	2010	All	Yes	No	Yes
	Cyber- WorkBench	NEC	Commercial	BDL	VHDL Verilog	2011	All	Cycle/ Formal	Yes	Yes
	LegUp	U. Toronto	Academic	C	Verilog	2011	All	Yes	Yes	No
	Bambu	PoliMi	Academic	C	Verilog	2012	All	Yes	Yes	No
	DWARV	TU. Delft	Academic	C subset	VHDL	2012	All	Yes	Yes	Yes
N/A	VivadoHLS	Xilinx	Commercial	C/C++ SystemC	VHDL/Verilog SystemC	2013	All	Yes	Yes	Yes
	Trident	Los Alamos NL	Academic	C subset	VHDL	2007	Scientific	No	Yes	No
	CHiMPS	U. Washington	Academic	C	VHDL	2008	All	No	No	No
	Kiwi	U. Cambridge	Academic	C#	Verilog	2008	.NET	No	No	No
	gcc2verilog [45]	U. Korea	Academic	C	Verilog	2011	All	No	No	No
Abandoned	HercuLeS	Ajax Compiler	Commercial	C/NAC	VHDL	2012	All	Yes	Yes	Yes
	Napa-C	Sarnoff Corp.	Academic	C subset	VHDL/Verilog	1998	Loop	No	No	No
	DEFACTO	U. South Calif.	Academic	C	RTL	1999	DSE	No	No	No
	Garp	U. Berkeley	Academic	C subset	bitstream	2000	Loop	No	No	No
	MATCH	U. Northwest	Academic	MATLAB	VHDL	2000	Image	No	No	No
	PipeRench	U.Carnegie M.	Academic	DIL	bitstream	2000	Stream	No	No	No
	SeaCucumber	U. Brigham Y.	Academic	Java	EDIF	2002	All	No	Yes	Yes
	SA-C	U. Colorado	Academic	SA-C	VHDL	2003	Image	No	No	No
	SPARK	U. Cal. Irvine	Academic	C	VHDL	2003	Control	No	No	No
	AccelDSP	Xilinx	Commercial	MATLAB	VHDL/Verilog	2006	DSP	Yes	Yes	Yes
	C2H	Altera	Commercial	C	VHDL/Verilog	2006	All	No	No	No
	CtoVerilog	U. Haifa	Academic	C	Verilog	2008	All	No	No	No

# Outils de haut niveau : HLS

	Commercial			BAMBU			DWARV			LEGUP		
Benchmark	LUTp	BRAMB18	DSP48s	LUTp	BRAMB18	DSP48s	LUTp	BRAMB18	DSP48s	ALMs	M20K	DSPs
adpcm_encode	4319	0	68	19931	52	64	5626	18	6	2490	0	43
aes_encrypt	5802	6	1	8485	4	0	15699	16	3	4263	8	0
aes_decrypt	6098	4	1	8747	4	1	12733	16	3	4297	14	0
gsm	5271	8	49	11864	10	75	6442	0	8	4311	1	51
sha	2161	16	0	4213	12	0	10012	0	0	6398	26	0
blowfish	2226	0	0	6837	0	0	7739	0	0	1679	0	0
dfadd	7409	0	0	7250	0	0	7334	0	0	2812	1	0
dfdiv	15107	0	24	11757	0	24	13934	1	40	4679	4	42
dfmul	3070	0	16	3430	0	16	14157	1	40	1464	1	28
dfsin	22719	0	43	21892	0	59	30616	43	43	9099	3	72
jpeg	16192	25	1	46757	154	26	ERR	ERR	ERR	16276	41	85
mips	1963	3	8	2501	0	8	3904	3	20	1319	0	15
motion	ERR	ERR	ERR	2776	2	0	45826	6	0	6788	0	0
satd	790	0	0	4425	0	0	1411	0	0	2004	0	0
sobel	792	0	6	3106	0	28	1160	0	12	1241	0	36
bellmanford	485	0	0	1046	0	0	633	0	0	493	0	0
matrix	175	0	3	551	0	3	471	0	3	225	0	2
<b>GEOMEAN</b>	2711.75	1.84	4.57	5253.60	2.15	5.30	5148.72	2.43	4.88	2197.66	2.01	5.67
<b>GEOMEAN (ALL)</b>				5754.49	2.76	5.28				2641.94	2.30	6.00

# Gaut3.0



---

# Conception Orientée Plate forme

## Platform based design

---

Solution ALTERA



# Architecture des kits

## ■ Centrée autour du FPGA

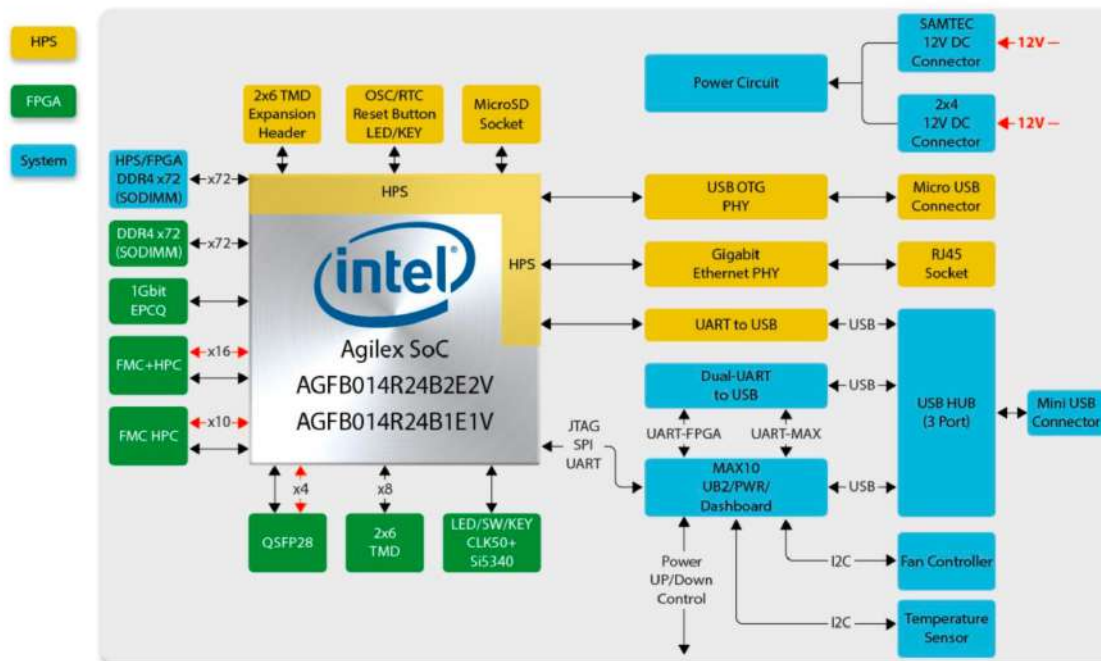
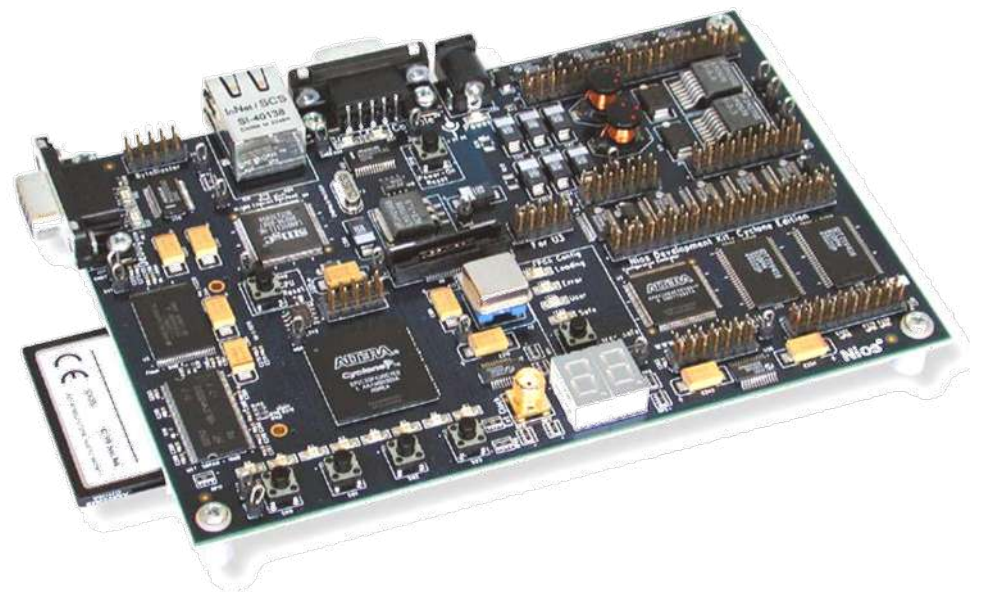


Figure 1-4 Block diagram of the Apollo Agilex board

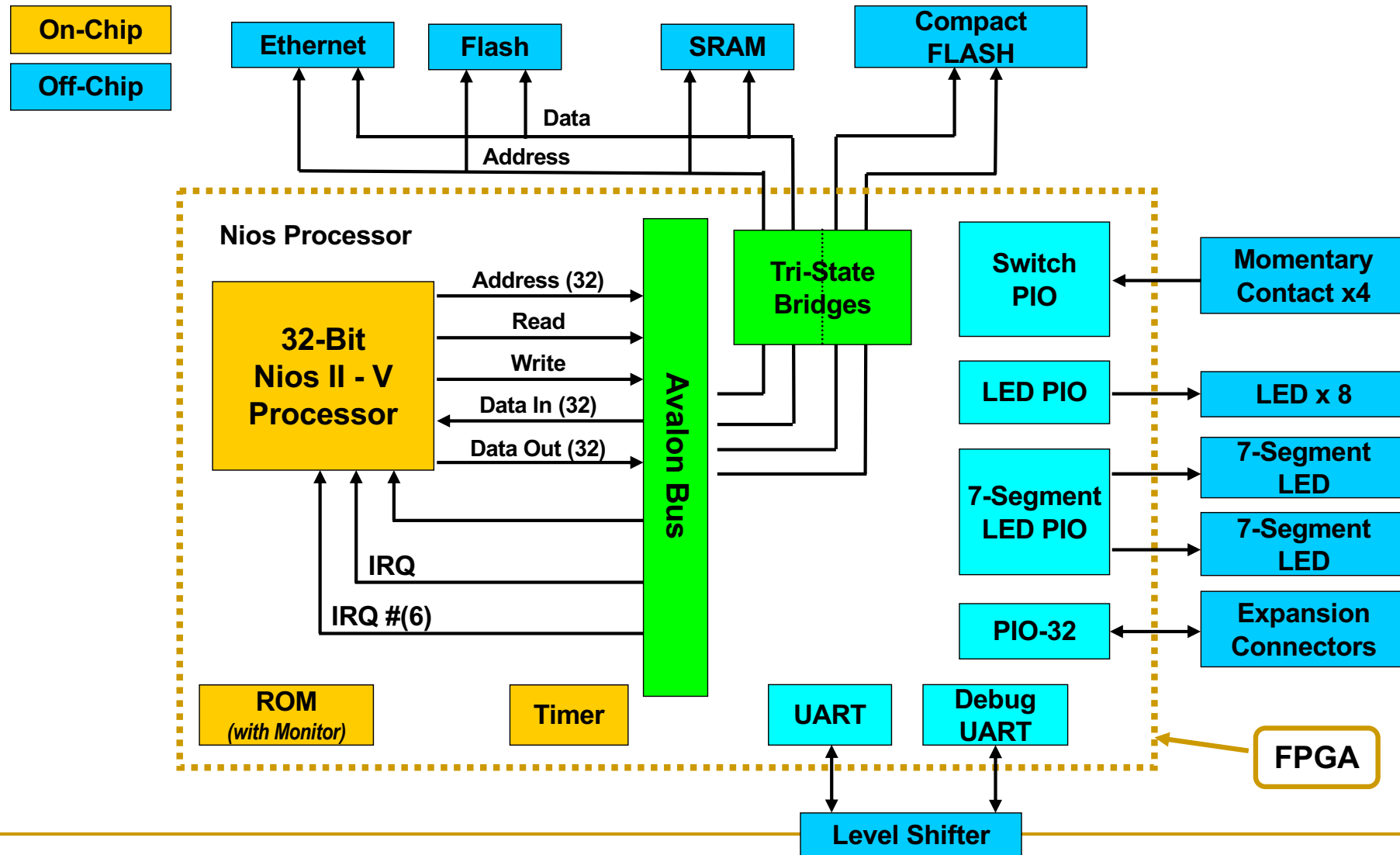


# Que peut on mettre dans le FPGA ?

- ❑ Système numérique classique décrit en HDL
- ❑ Système numérique plus complexe
  - Processeur Nios II ou 5
  - +périphériques matérielles existants
  - +propres périphériques
  - algos...



# Carte Cyclone : exemple de système



# Flot de conception

- Système numérique classique :
  - Logique combinatoire + séquentielle
- Système numérique complexe :
  - Microprocesseur Nios
  - Logique combinatoire + séquentielle
  - ...



*Flot de conception similaire*

---

# Processeur Nios II

- 
- Caractéristiques techniques
  - Flot de conception
  - Conception Matérielle
  - Conception Logicielle

# Qu'est ce qu'un Nios II

## ■ **Nios : IP logiciel d'un $\mu$ P**

- ❑ Développé par Altera
  - Première version en 1999
- ❑ Architecture de type RISC
- ❑ Entièrement configurable
  - 16 ou 32 bits, nombre de registres (128, 256 ou 512) ...
- ❑ Écrit en langage HDL
- ❑ Exploitable pour tous les FPGA de la gamme Altera
- ❑ Entièrement synthétisable



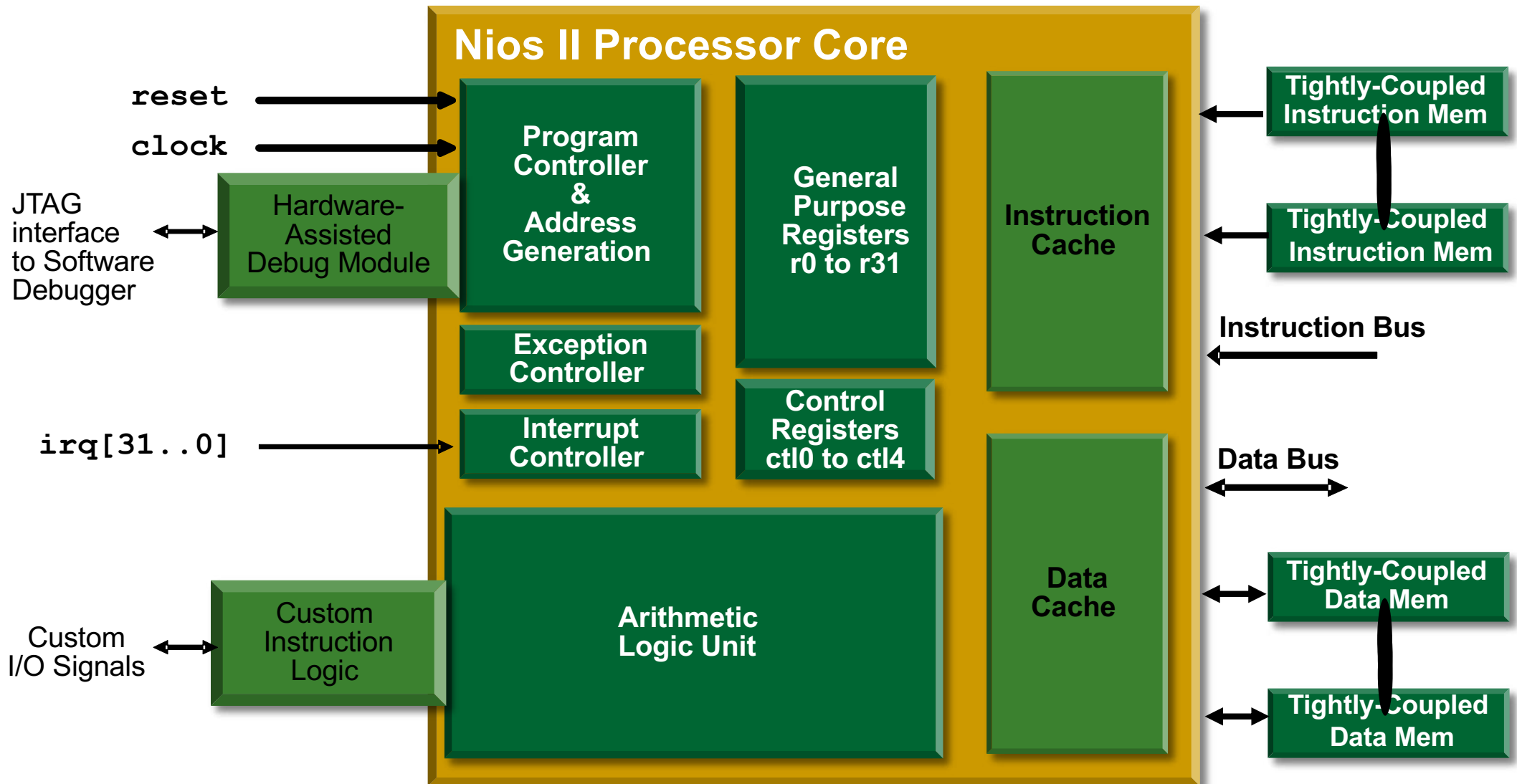
---

# Nios II Processor Architecture

## ■ Classic Pipelined RISC Machine

- ❑ 32 General Purpose Registers
  - ❑ 32-Bit Instructions
  - ❑ 32-Bit Data Path
  - ❑ 32 Prioritized Interrupts
  - ❑ On-Chip Hardware (Multiply, Shift, Rotate)
    - Single Instruction 32x32 multiply and divide
  - ❑ Custom Instructions
  - ❑ Performance up to 200 DMIPS (Dhrystone MIPS)
  - ❑ JTAG-Based Hardware Debug Unit
-

# Nios II Processor Block Diagram



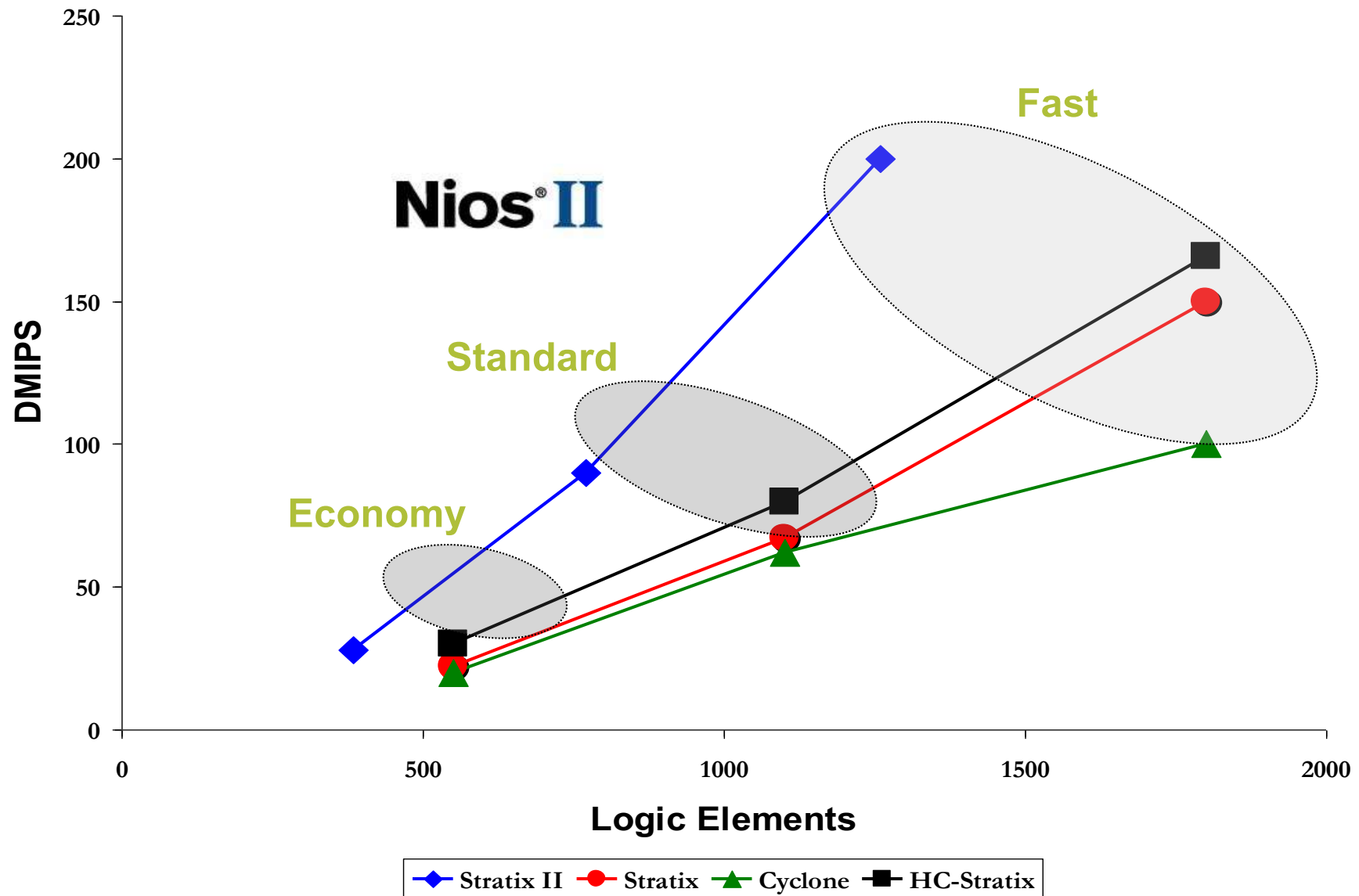
	<b>Nios II /f Fast</b>	<b>Nios II /s Standard</b>	<b>Nios II /e Economy</b>
<b>Pipeline</b>	<b>6 Stage</b>	<b>5 Stage</b>	<b>None</b>
<b>H/W Multiplier &amp; Barrel Shifter</b>	<b>1 Cycle</b>	<b>3 Cycle</b>	<b>Emulated In Software</b>
<b>Instruction Cache</b>	<b>Configurable</b>	<b>Configurable</b>	<b>None</b>
<b>Data Cache</b>	<b>Configurable</b>	<b>None</b>	<b>None</b>
<b>Logic Usage (Logic Elements)</b>	<b>1400 - 1800</b>	<b>1200 – 1400</b>	<b>600 – 700</b>
<b>Custom Instructions</b>	<b>Up to 256</b>		

# Hardware Multiplier Acceleration

- Nios II Economy version - No Multiply Hardware
  - Uses GNUPro Math Library to Implement Multiplier
- Nios II Standard - Full Hardware Multiplier
  - $32 \times 32 \rightarrow 32$  in 3 Clock Cycles if DSP block present, else uses software only multiplier
- Nios II Fast - Full Hardware Multiplier
  - $32 \times 32 \rightarrow 32$  in 1 Clock Cycles if DSP block present, else uses software only multiplier

Acceleration Hardware	Clock Cycles ( $32 \times 32 \rightarrow 32$ )
None	250
Standard MUL in Stratix	3
Fast MUL in Stratix	1

# Variation with FPGA Device



Device Family	Device OPN	Nios II/f	Nios II/e	Results are generated using
Intel Agilex	AGFA014R24A2E2VR0	400	410	Intel Quartus Prime Pro Edition software version 20.1
Intel Stratix 10	1SG250LN3F43I2LG	300	320	Intel Quartus Prime Pro Edition software version 20.1
Stratix V	5SGXEA7N2F45C1	350	410	Intel Quartus Prime Standard Edition software version 19.1
Stratix IV	EP4S100G5H40I1	240	280	Intel Quartus Prime Standard Edition software version 19.1
Intel Arria 10	10AX115U3F45I2LG	290	340	Intel Quartus Prime Pro Edition software version 20.1
Arria V GZ	5AGZME7K2F40C3	280	360	Intel Quartus Prime Standard Edition software version 19.1
Arria V	5AGXFB5K4F40I3	200	260	Intel Quartus Prime Standard Edition software version 19.1
Intel Cyclone 10 GX	10CX220YF780E5G	280	330	Intel Quartus Prime Pro Edition software version 20.1
Intel Cyclone 10 LP	10CL120YF780I7G	140	160	Intel Quartus Prime Standard Edition software version 19.1
Cyclone V	5CGXFC7D6F31C6	170	210	Intel Quartus Prime Standard Edition software version 19.1
Cyclone IV	EP4CGX30CF19C6	160	170	Intel Quartus Prime Standard Edition software version 19.1
Intel MAX® 10	10M50DAF484C6GES	160	160	Intel Quartus Prime Standard Edition software version 19.1



Device Family	Processor Core / Peripheral							Results are generated using
	Nios II/f	Nios II/e	Nios II JTAG debug module	Avalon UART	JTAG UART	SDRAM Controller <sup>(4)</sup>	Timer	
Intel Agilex	1002	596	149	57	71	5999	77	Intel Quartus Prime Pro Edition software version 20.1
Intel Stratix 10 (ALM)	1006	414	148	58	70	4429	76	Intel Quartus Prime Pro Edition software version 20.1
Stratix V (ALM)	697	296	129	62	56	2635	68	Intel Quartus Prime Standard Edition software version 19.1
Stratix IV (ALUT)	1073	527	169	95	112	3809	92	Intel Quartus Prime Standard Edition software version 19.1
Intel Arria 10 (ALM)	803	322	114	55	60	3973	60	Intel Quartus Prime Pro Edition software version 20.1
Arria V GZ (ALM)	706	290	125	55	56	2632	54	Intel Quartus Prime Standard Edition software version 19.1
Arria V (ALM)	835	310	126	56	56	2470	55	Intel Quartus Prime Standard Edition software version 19.1
Intel Cyclone 10 GX (ALM)	847	346	116	56	60	2291	55	Intel Quartus Prime Pro Edition software version 20.1
Intel Cyclone 10 LP (LE)	2326	838	464	141	163	435	148	Intel Quartus Prime Standard Edition software version 19.1
Cyclone V (ALM)	848	305	127	56	57	2473	56	Intel Quartus Prime Standard Edition software version 19.1
Cyclone IV GX (ALUT)	2221	772	352	143	160	424	138	Intel Quartus Prime Standard Edition software version 19.1
Intel MAX 10 (LE)	2211	790	364	136	157	4671	139	Intel Quartus Prime Standard Edition software version 19.1

# Nios II: Hard Numbers

	Nios II/f	Nios II/s	Nios II/e
Stratix II	<b>200 DMIPS @ 175MHz</b> <b>1180 LEs</b> 1 of 8 DSP 4K Icache, 2K Dcache Stratix 2S10-C5	<b>90 DMIPS @ 175MHz</b> <b>800 LEs</b> 4K Icache, No Dcache Stratix 2S10-C5	<b>28 DMIPS @ 190MHz</b> <b>400 LEs</b> No Icache, No Dcache Stratix 2S10-C5
Stratix	<b>150 DMIPS @ 135MHz</b> <b>1800 LEs</b> 1 of 8 DSP 4K Icache, 2K Dcache Stratix 1S10-C5	<b>67 DMIPS @ 135MHz</b> <b>1200 LEs</b> 4K Icache, No Dcache Stratix 1S10-C5	<b>22 DMIPS @ 150MHz</b> <b>550 LEs</b> No Icache, No Dcache Stratix 1S10-C5
Cyclone	<b>100 DMIPS @ 125MHz</b> <b>1800 LEs</b> 4K Icache, 1K Dcache Cyclone 1C4-C6	<b>62 DMIPS @ 125MHz</b> <b>1200 LEs</b> 2K Icache, No Dcache Cyclone 1C4-C6	<b>20 DMIPS @ 140MHz</b> <b>550 LEs</b> No Icache, No Dcache Cyclone 1C4-C6

Performance Metric	Nios II/f	Nios II/e
DMIPS/MHz Ratio	0.753	0.107
CoreMark	229.148	19.234

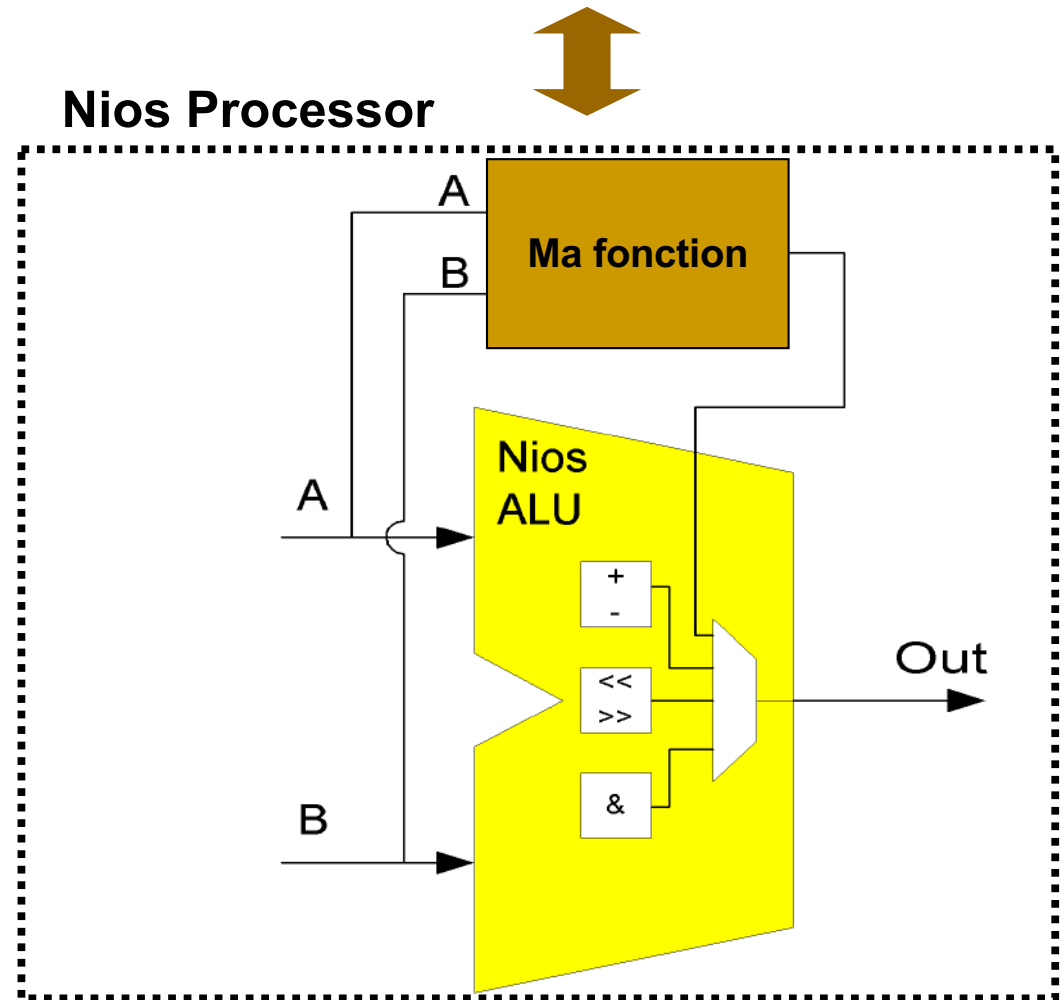
\* FMax Numbers Based Reference Design Running From On-Chip Memory (Nios II/f  $\cong$  1.15 DMIPS / MHz)

# Custom Instruction

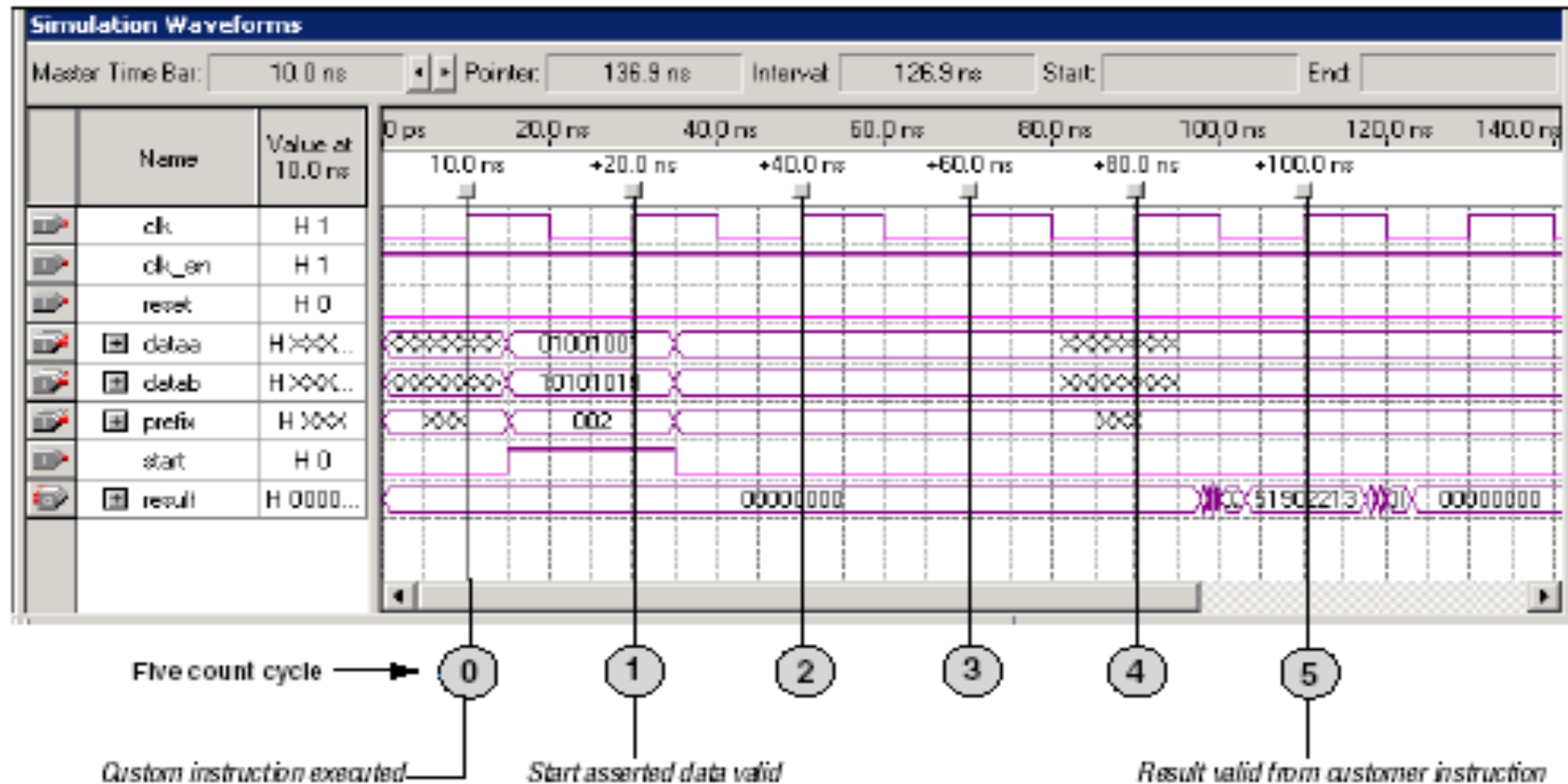
Fonctions logiques et arithmétiques

## Idées

- Augmenter les performances du Nios sans augmenter  $f_{MAX}$ 
  - **Accélération matérielle**
- Etendre le jeu d'instruction
  - Jusqu'à 5 Instructions
- Définir son propre co-processeur optimisé



# Custom Instruction



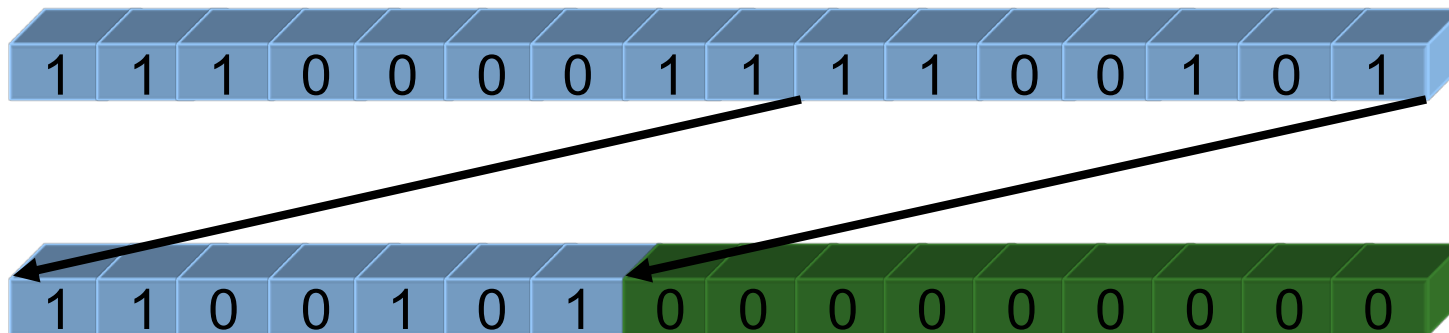
# Custom Instruction : Exemple

## ■ Multiplieur : 16bits x 16bits

- Logiciel (Librairie Math de GNU) :
  - 80 cycles d'horloge
  - 0 EL
- Matériel :
  - 1 cycle d'horloge
  - 85 EL + 2 éléments précablés dans le FPGA

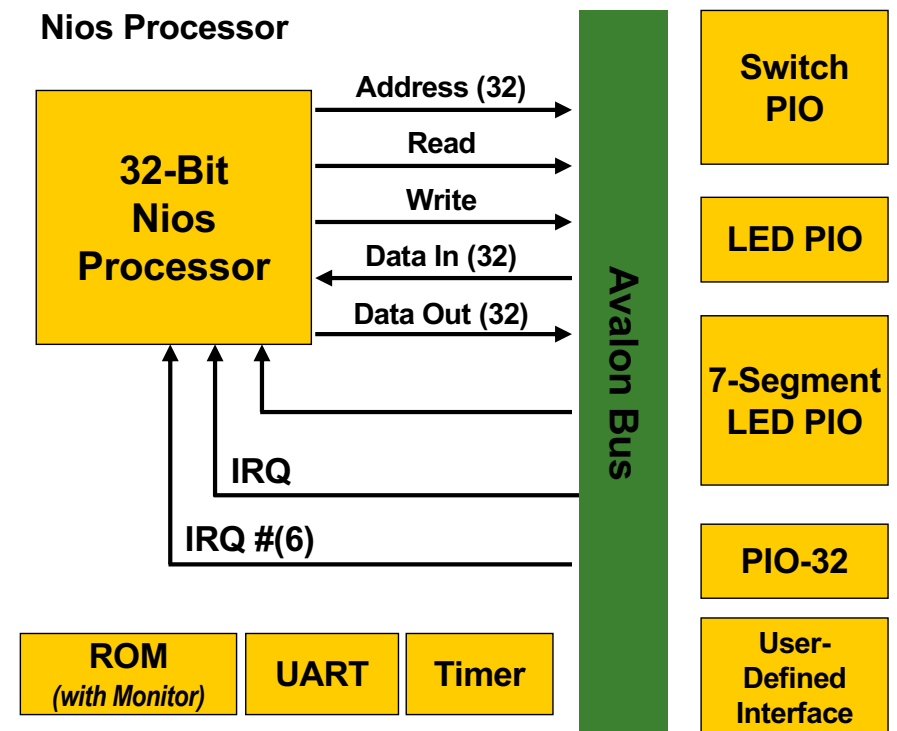
## ■ Registre à décalage : décalage de 9 bits

- Exécution en 9 cycle d'horloge
- Accélération matérielle : 1 cycle d'horloge



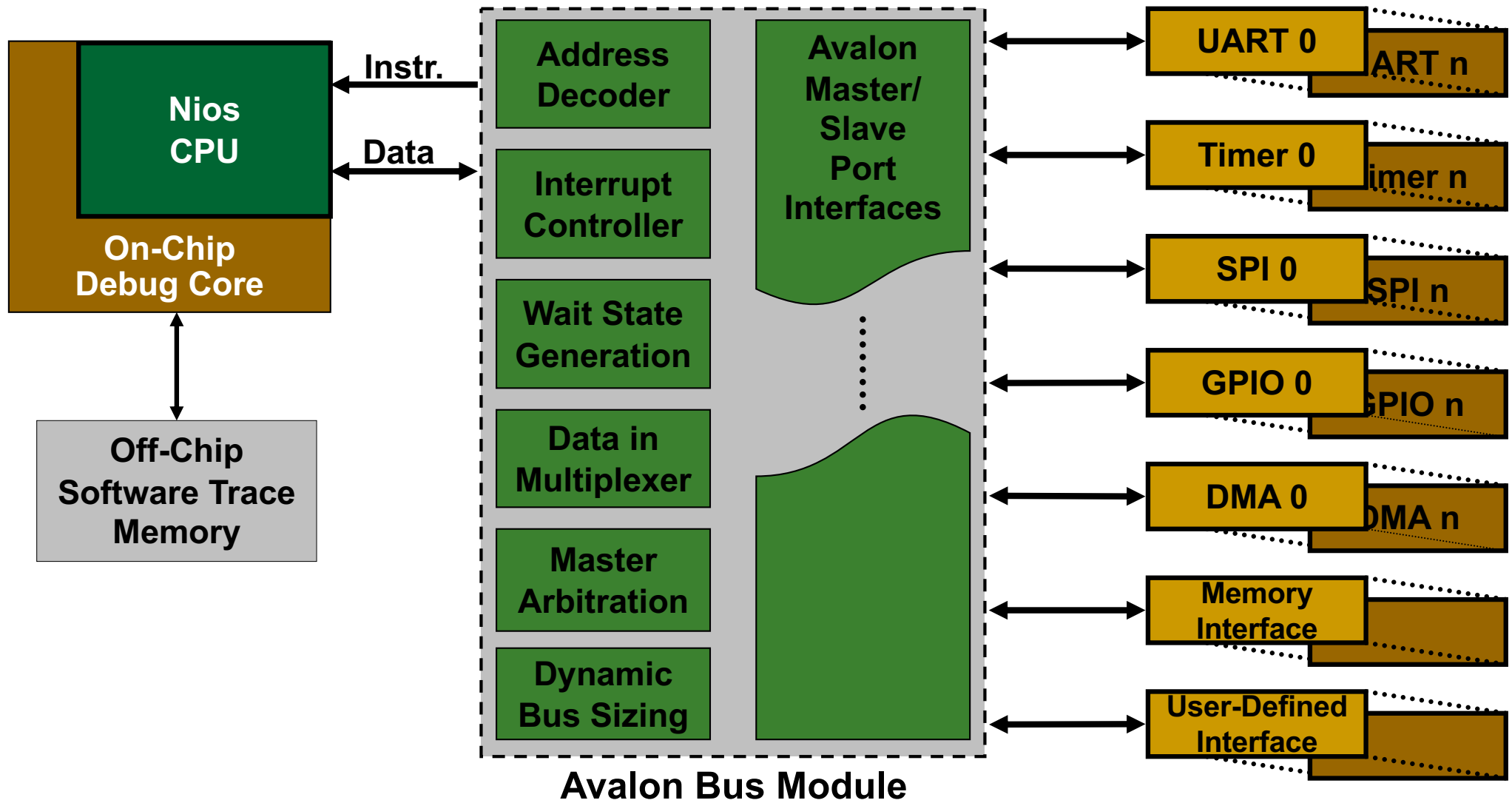
# Bus Avalon

- Bus permettant aux Nios d'accéder aux périphériques systèmes
- Bus Altera spécifique au Nios
  - Carte ARM ⇔ bus amba
- Principales caractéristiques
  - Entièrement paramétrable
    - Taille adresse, donnée, ...
  - Simplicité de mise en oeuvre
  - Opérations synchrones / horloge sys
- Types de transferts
  - Slave Transfers
  - Master Transfers
  - Streaming Transfers

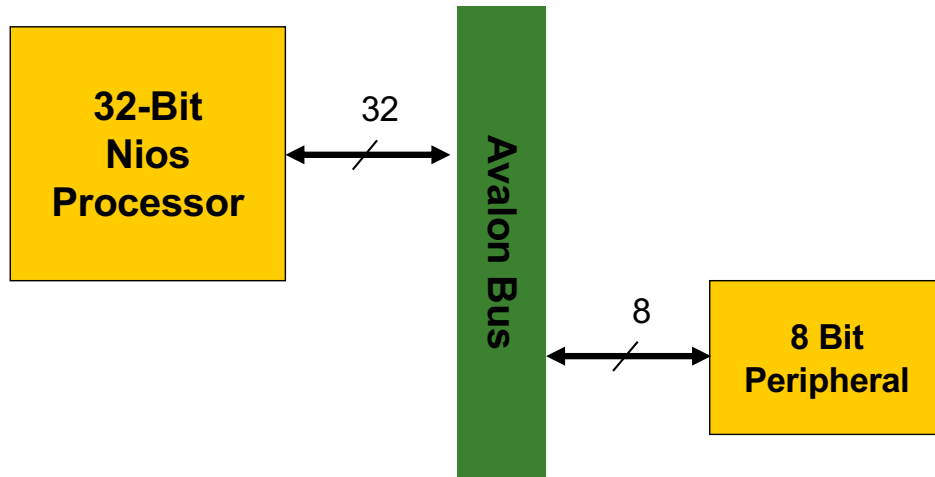




# Architecture du bus Avalon



# Dynamic bus sizing



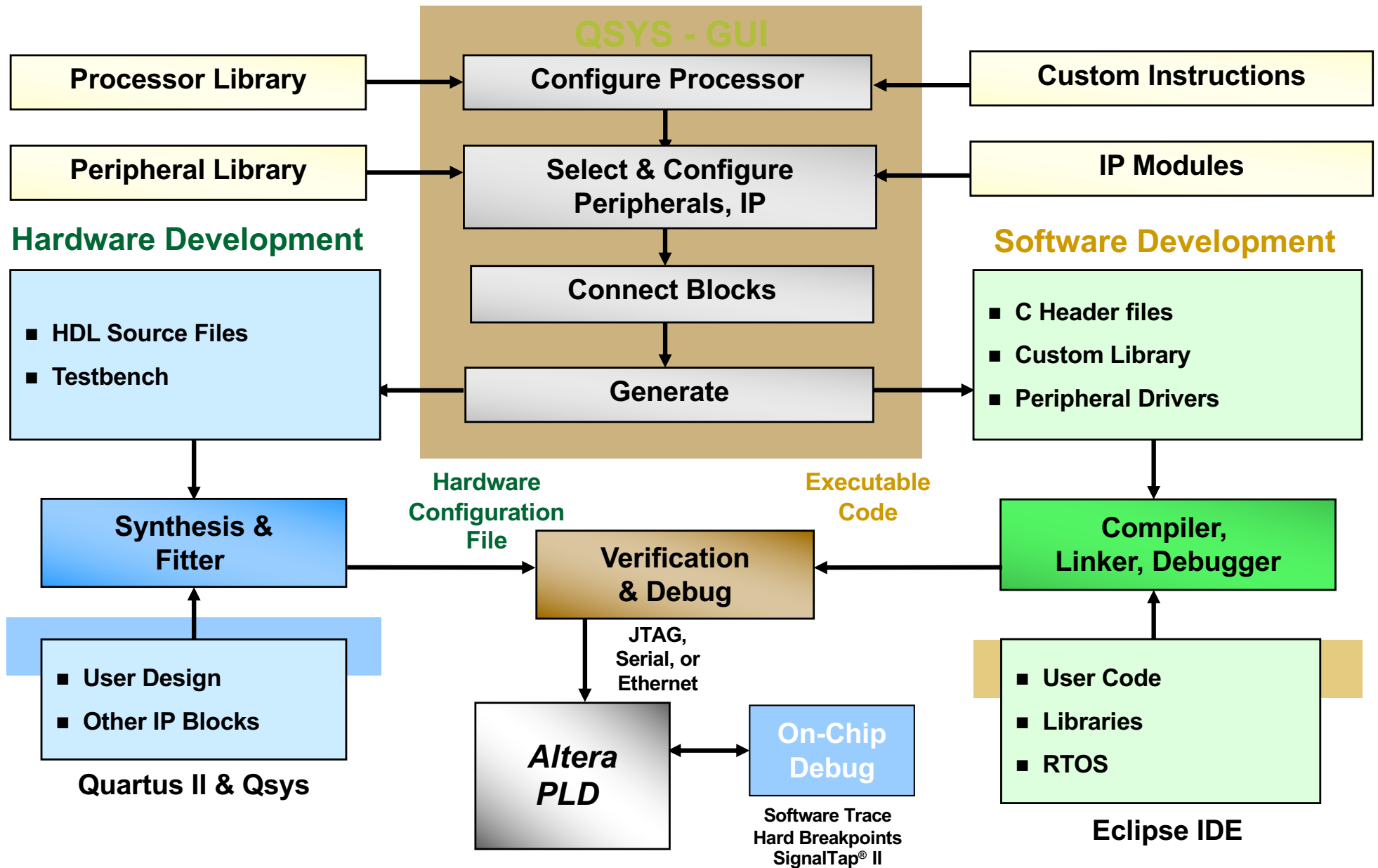
Peripheral Registers	
Base	aa
Base + 0x1	bb
Base + 0x2	cc
Base + 0x3	dd
Base + 0x4	ee

## ■ Native Address Alignment

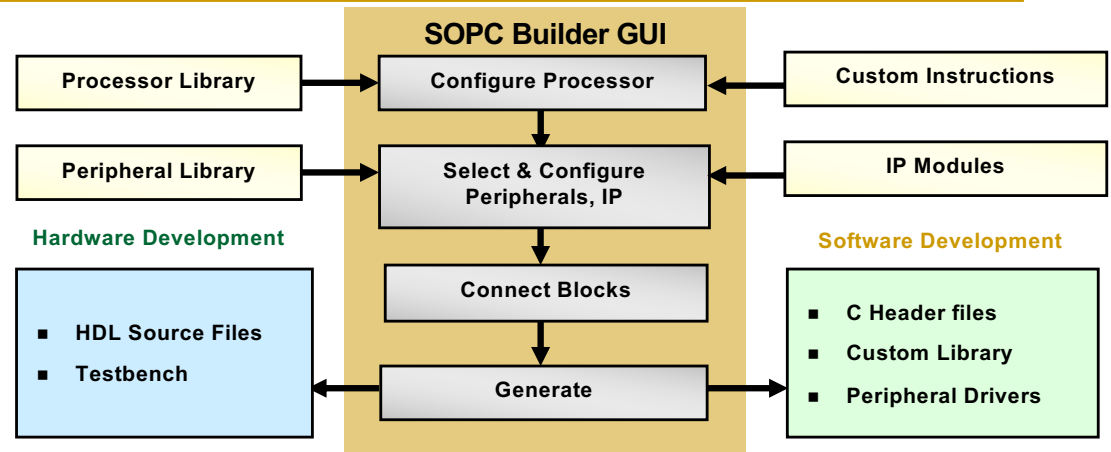
- ❑ LD from Base + 0x0: uu uu uu aa
- ❑ LD from Base + 0x4: uu uu uu bb
- ❑ LD from Base + 0x8: uu uu uu cc

# Flot de conception d'un système à base de Nios II

---



# Platform Designer ex Qsys



## Conception modulaire d'un système à $\mu$ P à base d'IP

- Sélection et configuration du  $\mu$ P (16Bits ou 32 Bits) + accélération matérielle
- Sélection et configuration des périphériques (IP) / bibliothèques
- Interconnexions des IPs / Bus
  - Assignation des adresses en mémoires des périphériques
  - Assignation des N° d'Interruption

HW : fichiers HDL + Test Bench (ModelSim ou Quartus II)

SW : fichiers Header (ex : nios.h) des périphériques pour leur programmation

# Platform Designer

The screenshot shows the Altera SOPC Builder - nios interface. The top menu bar includes File, System, Module, View, and Help. The main window is divided into two panes. The left pane, titled 'System Contents', shows a tree view of the system components. The right pane, titled 'Nios More "cpu" Settings System Generation', displays a table of system components and their properties.

System Clock Frequency: 50 MHz

Use	Module Name	Description	Base	End	IRQ
<input checked="" type="checkbox"/>	<b>cpu</b>	Nios Processor - Altera Corporation			
<input checked="" type="checkbox"/>	<b>boot_rom</b>	On-Chip Memory (RAM or ROM)	0x00900000	0x009007FF	
<input checked="" type="checkbox"/>	<b>uart1</b>	UART (RS-232 serial port)	0x00900800	0x0090081F	16
<input checked="" type="checkbox"/>	<b>seven_seg_pio</b>	PIO (Parallel I/O)	0x00900840	0x0090084F	
<input checked="" type="checkbox"/>	<b>timer1</b>	Interval timer	0x00900820	0x0090083F	17
<input type="checkbox"/>	<b>led_pio</b>	PIO (Parallel I/O)	0x00900850	0x0090085F	
<input checked="" type="checkbox"/>	<b>button_pio</b>	PIO (Parallel I/O)	0x00900860	0x0090086F	18
<input checked="" type="checkbox"/>	<b>ext_ram_bus</b>	Avalon Tri-State Bridge			
<input checked="" type="checkbox"/>	<b>ext_ram</b>	IDT71V416 SRAM for EP1S10 Nios Development Board	0x00800000	0x008FFFFFFF	
<input checked="" type="checkbox"/>	<b>ext_flash</b>	AMD 29LV065D Flash for EP1S10 Nios Development B...	0x00000000	0x007FFFFFFF	
<input checked="" type="checkbox"/>	<b>my_pwm</b>	Interface to User Logic	0x00900850	0x00900857	
<input checked="" type="checkbox"/>	<b>avalon_dma</b>	DMA	0x00900880	0x0090089F	19

Le système en cours de développement

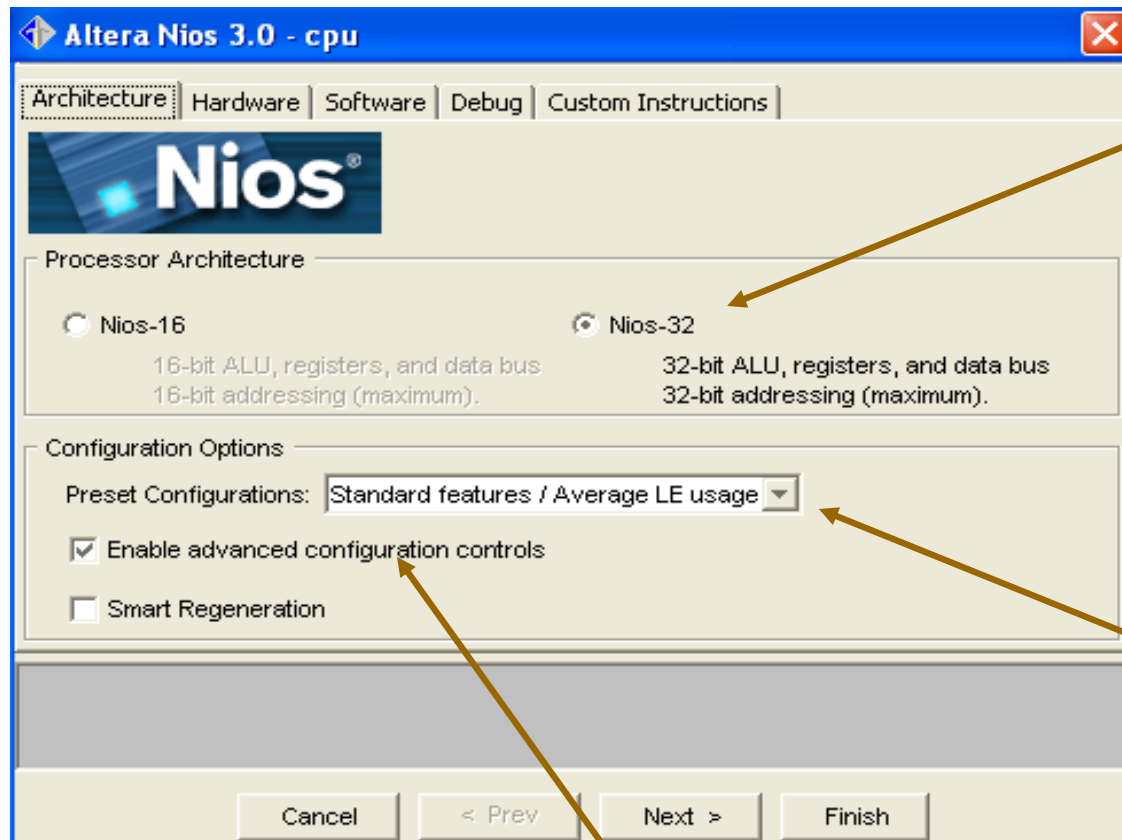
Bibliothèque d'IPs disponibles

Installed components: [Icons]

Buttons: Add..., Check, Move Up, Move Down



# Nios CPU



16 or 32 Bit

Configuration standard

Configuration avancée

# Nios CPU Hardware

Instruction & Data Cache

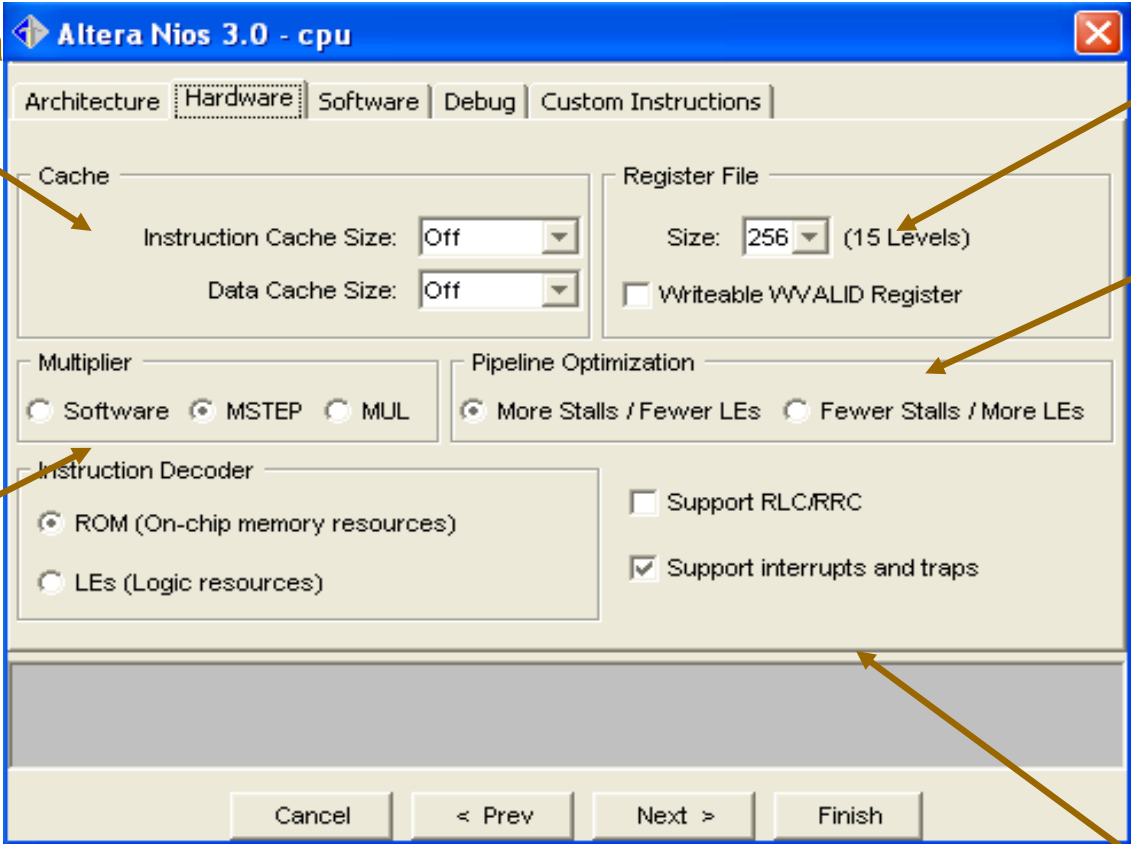
Type de Multiplieur

Nombre de registres

LE/fmax

Optimisation

Interrupt Support



The screenshot shows the 'Altera Nios 3.0 - cpu' configuration window with the 'Hardware' tab selected. The window is divided into several sections: 'Cache' with 'Instruction Cache Size' and 'Data Cache Size' both set to 'Off'; 'Register File' with 'Size' set to '256' (15 Levels) and a checkbox for 'Writeable WVALID Register'; 'Multiplier' with radio buttons for 'Software', 'MSTEP', and 'MUL'; 'Pipeline Optimization' with radio buttons for 'More Stalls / Fewer LEs' and 'Fewer Stalls / More LEs'; 'Instruction Decoder' with radio buttons for 'ROM (On-chip memory resources)' and 'LEs (Logic resources)'; and a bottom section with checkboxes for 'Support RLC/RRC' and 'Support interrupts and traps'. The bottom of the window has buttons for 'Cancel', '< Prev', 'Next >', and 'Finish'. Annotations with arrows point to specific settings: 'Instruction & Data Cache' points to the cache size dropdowns; 'Type de Multiplieur' points to the multiplier radio buttons; 'Nombre de registres' points to the register file size dropdown; 'LE/fmax' points to the pipeline optimization radio buttons; 'Optimisation' points to the 'Support interrupts and traps' checkbox; and 'Interrupt Support' points to the same checkbox.

# Bibliothèque de périphériques disponibles

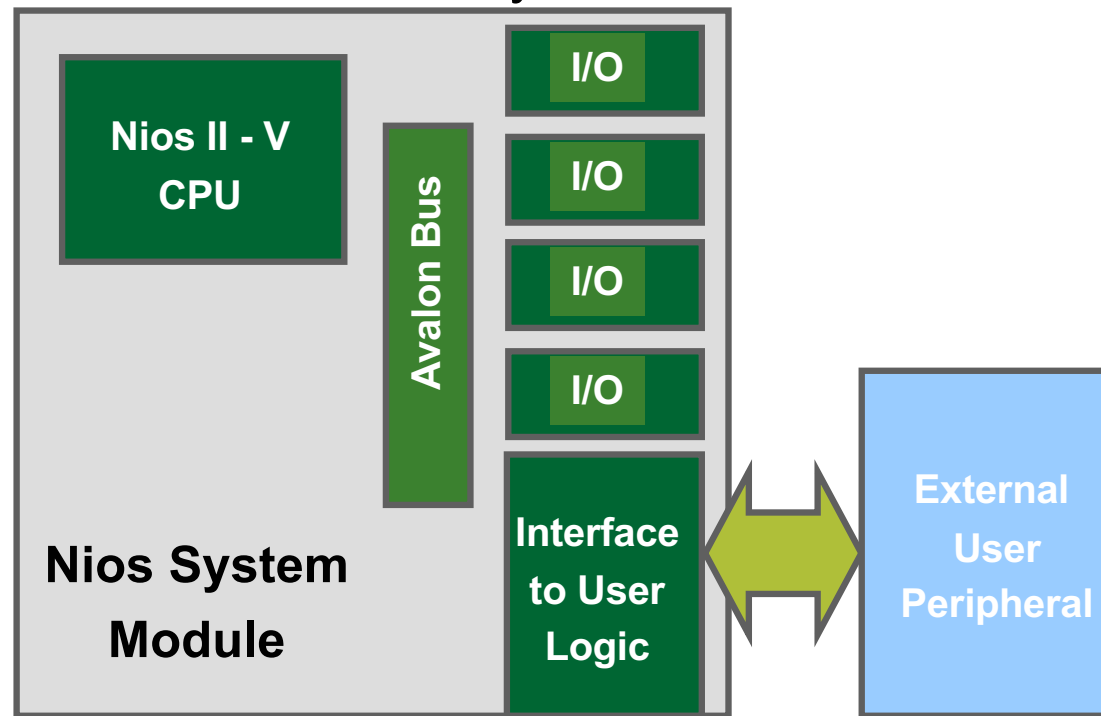
- Memory Interface
    - Active Serial Interface
    - On-Chip
      - RAM, ROM
    - Off-Chip
      - SDRAM Controller
      - SRAM
      - Flash
      - ROM
  - DMA Controller
    - Memory-Peripheral
    - Memory-Memory
    - Peripheral-Peripheral
  - Bridges
    - AHB to Avalon Bus Bridge
  - Parallel I/O (PIO) Registers
    - General-Purpose I/O Registers (PIO)
      - Input
      - Output
      - Bidirectional
    - User-Defined Interface
  - Serial I/O
    - UART
    - SPI
  - Timer
    - Simple Timer
    - Pulse Generator
    - Watchdog Timer
-

# Périphériques custom ?

- Comment faire si j'ai besoin d'utiliser dans mon système un composant qui n'existe pas dans la bibliothèque ?
  - Par exemple, une IP de type PWM ?
- Description HDL du composant + interfacage
  - 2 manières d'interfacer
    - Port PIO
      - Pas assez rapide
    - Connexion au bus Avalon

# Component editor – new component

- Encapsulation de l' IP dans un bloc “interface to user logic – New component”
  - ❑ Wrapper Avalon / interface IP
  - ❑ Définie dans le mapping mémoire du système
- Peut être “Inside” or “Outside” du système Nios



IP Catalog

Project

Library

Basic Functions

DSP

Interface Protocols

Memory Interfaces and Controllers

Processors and Peripherals

Qsys Interconnect

Tri-State Components

University Program

New... Edit... Add...

Hierarchy

Device Family

myfirstnios (myfirstnios.qsys)

clk\_sys

led\_pio external connection

reset\_sys

uart\_0

led\_pio

nios2\_proc

onchip\_ram

Connections

System Contents

Address Map

Interconnect Requirements

System: myfirstnios Path: clk\_0

Connections	Name	Description	Export	Clock	Base	End	Tags	Opco
<input checked="" type="checkbox"/>	clk_0	Clock Source						
<input checked="" type="checkbox"/>	clk_in	Clock Input	clk_sys	exported				
<input checked="" type="checkbox"/>	clk_in_reset	Reset Input	reset_sys					
<input checked="" type="checkbox"/>	clk	Clock Output	Double-click	clk_0				
<input checked="" type="checkbox"/>	clk_reset	Reset Output	Double-click					
<input checked="" type="checkbox"/>	led_pio	PIO (Parallel I/O) Intel FPGA IP	Double-click					
<input checked="" type="checkbox"/>	clk	Clock Input	Double-click	clk_0				
<input checked="" type="checkbox"/>	reset	Reset Input	Double-click	[clk]				
<input checked="" type="checkbox"/>	s1	Avalon Memory-Mapped Slave			0x0010	0x0015		
<input checked="" type="checkbox"/>	external_connection	Conduit						
<input checked="" type="checkbox"/>	onchip_ram	On-Chip						
<input checked="" type="checkbox"/>	clk1	Clock Input						
<input checked="" type="checkbox"/>	s1	Avalon I						
<input checked="" type="checkbox"/>	reset1	Reset In						
<input checked="" type="checkbox"/>	nios2_proc	Nios II						
<input checked="" type="checkbox"/>	clk	Clock In						
<input checked="" type="checkbox"/>	reset	Reset In						
<input checked="" type="checkbox"/>	data_master	Avalon I						
<input checked="" type="checkbox"/>	instruction_master	Avalon I						
<input checked="" type="checkbox"/>	irq	Interrupt						
<input checked="" type="checkbox"/>	debug_reset_request	Reset O						
<input checked="" type="checkbox"/>	debug_mem_slave	Avalon I						
<input checked="" type="checkbox"/>	custom_instruction_master	Custom						
<input checked="" type="checkbox"/>	jtag_uart_0	JTAG UI						
<input checked="" type="checkbox"/>	clk	Clock In						

Current filter:

Messages

Type	Path	Message
Info	1	Info Message
Warning	myfirstnios.jtag_uart_0	JTAG UART IP input clock nee

0 Errors, 0 Warnings

Component Editor - new\_component\_hw.tcl\*

File Templates Beta View

Component Type Block Symbol Files Parameters Signals & Interfaces

About Files

Synthesis Files

These files describe this component's implementation, and will be created when a Quartus synthesis model is generated.

The parameters and signals found in the top-level module will be used for this component's parameters and signals.

Output Path	Source File	Type	Attributes
(No files)			

Add File... Remove File Analyze Synthesis Files Create Synthesis File from Signals

Top-level Module: (Analyze files to select module)

Verilog Simulation Files

These files will be produced when a Verilog simulation model is generated.

Output Path	Source File	Type	Attributes
-------------	-------------	------	------------

Messages

To Do: Add HDL files on the Files tab, or add signals on the Signals tab.

Help Prev Next Finish...

filled IP  
 Object Directory  
 System  
 Library  
 Basic Functions  
 DSP  
 Interface Protocols  
 Memory Interfaces and Controllers  
 Processors and Peripherals  
 University Program  
 Search for Partner IP

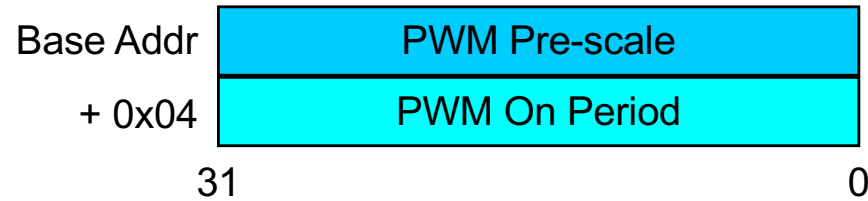
Activer Windows  
 Accédez aux paramètres pour activer Windows.

Messages

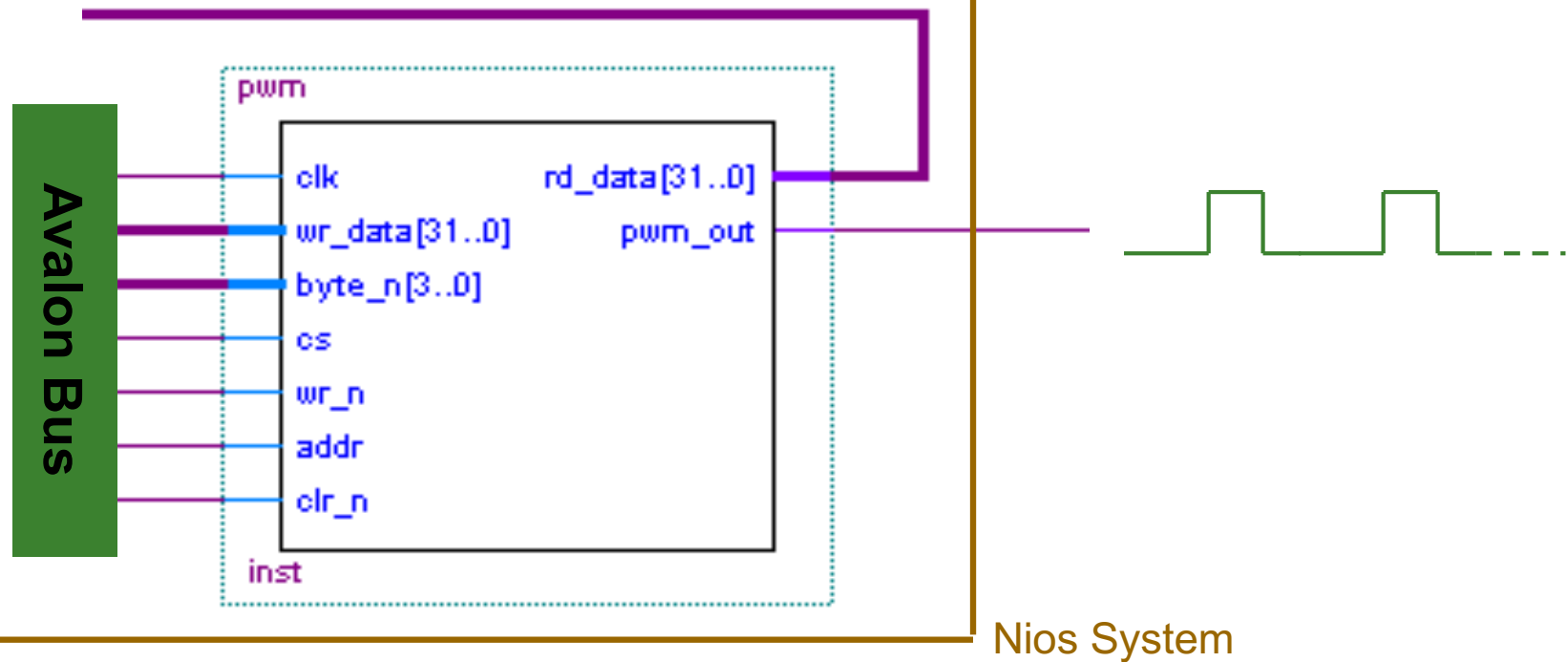
System Processing

0% 00:00:00

# Interface to user logic : example

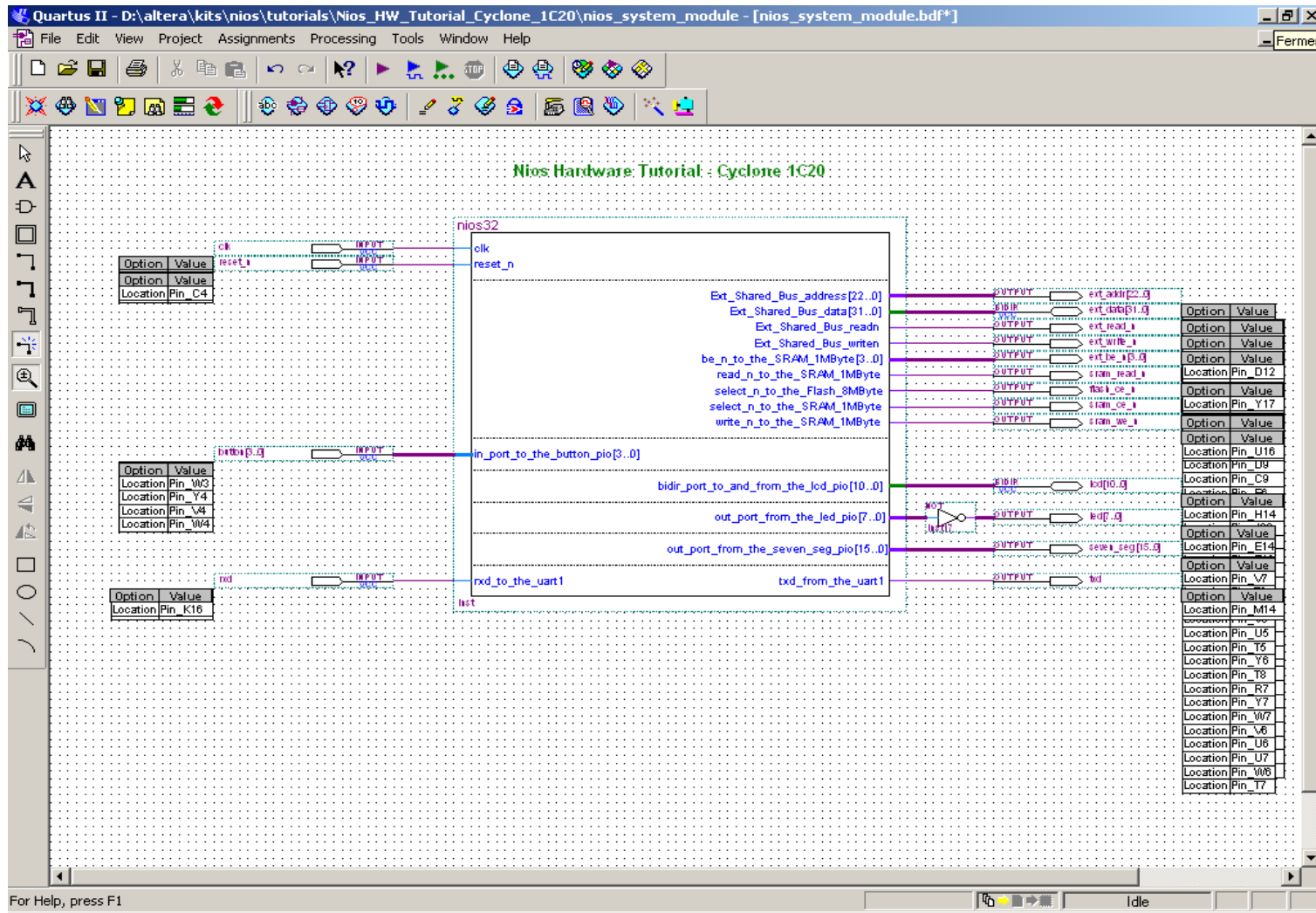


$$\text{PWM Duty Cycle} = \frac{\text{On Period}}{\text{Pre-scale}}$$





# Génération du Système



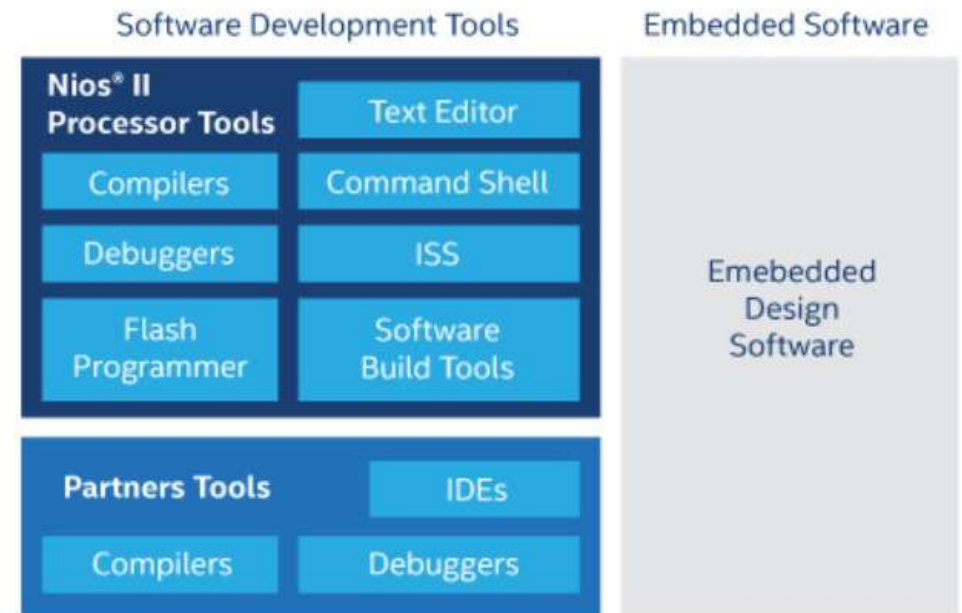
---

# Conception Logicielle

- 
- Environnement de développement
  - Commande de compilation et d'exécution

# Nios II IDE

- Leading Edge Software Development Tool
- Target Connections
  - Hardware (JTAG)
  - Instruction Set Simulator
  - ModelSim®-Altera Software
- Advanced Hardware Debug Features
  - Software and Hardware Break Points, Data Triggers, Trace
- Flash Memory Programming Support



\* Based on Eclipse Project

# Eclipse

Nios II - mythirdnios\_bsp/drivers/inc/altera\_avalon\_pio\_regs.h - Eclipse

File Edit Source Refactor Navigate Search Project Run Nios II Window Help

Project Explorer

- mythirdnios
  - Binaries
  - Includes
  - obj
  - system
  - hello\_world\_small.c
  - mythirdnios.eif - [alteranios2/le]
  - create-this-app
  - Makefile
  - mythirdnios.map
  - mythirdnios.objdump
  - readme.txt
- mythirdnios\_bsp [myfirstnios2]
  - Includes
  - drivers
    - inc
      - altera\_avalon\_jtag\_uart\_fd.h
      - altera\_avalon\_jtag\_uart\_regs.h
      - altera\_avalon\_jtag\_uart.h
      - altera\_avalon\_pio\_regs.h
    - src
  - HAL
  - alt\_sys\_init.c
  - linker.h
  - system.h
  - create-this-bsp
  - linker.x
  - Makefile
  - mem\_init.mk
  - memory.gdb
  - public.mk
  - settings.bsp
  - summary.html

hello\_world\_small.c system.h altera\_avalon\_pio\_regs.h

```
30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66
License Agreement
#ifndef ALTERA_AVALON_PIO_REGS_H
#define ALTERA_AVALON_PIO_REGS_H
#include <io.h>
#define IOADDR_ALTERA_AVALON_PIO_DATA(base) __IO_CALC_ADDRESS_NATIVE(base, 0)
#define IORD_ALTERA_AVALON_PIO_DATA(base) IORD(base, 0)
#define IOWR_ALTERA_AVALON_PIO_DATA(base, data) IOWR(base, 0, data)
#define IOADDR_ALTERA_AVALON_PIO_DIRECTION(base) __IO_CALC_ADDRESS_NATIVE(base, 1)
#define IORD_ALTERA_AVALON_PIO_DIRECTION(base) IORD(base, 1)
#define IOWR_ALTERA_AVALON_PIO_DIRECTION(base, data) IOWR(base, 1, data)
#define IOADDR_ALTERA_AVALON_PIO_IRQ_MASK(base) __IO_CALC_ADDRESS_NATIVE(base, 2)
#define IORD_ALTERA_AVALON_PIO_IRQ_MASK(base) IORD(base, 2)
#define IOWR_ALTERA_AVALON_PIO_IRQ_MASK(base, data) IOWR(base, 2, data)
#define IOADDR_ALTERA_AVALON_PIO_EDGE_CAP(base) __IO_CALC_ADDRESS_NATIVE(base, 3)
#define IORD_ALTERA_AVALON_PIO_EDGE_CAP(base) IORD(base, 3)
#define IOWR_ALTERA_AVALON_PIO_EDGE_CAP(base, data) IOWR(base, 3, data)
#define IOADDR_ALTERA_AVALON_PIO_SET_BIT(base) __IO_CALC_ADDRESS_NATIVE(base, 4)
#define IORD_ALTERA_AVALON_PIO_SET_BITS(base) IORD(base, 4)
#define IOWR_ALTERA_AVALON_PIO_SET_BITS(base, data) IOWR(base, 4, data)
#define IOADDR_ALTERA_AVALON_PIO_CLEAR_BITS(base) __IO_CALC_ADDRESS_NATIVE(base, 5)
#define IORD_ALTERA_AVALON_PIO_CLEAR_BITS(base) IORD(base, 5)
#define IOWR_ALTERA_AVALON_PIO_CLEAR_BITS(base, data) IOWR(base, 5, data)
/* Definitions for direction-register operation with bi-directional PIOs */
#define ALTERA_AVALON_PIO_DIRECTION_INPUT 0
#define ALTERA_AVALON_PIO_DIRECTION_OUTPUT 1
```

Outline


- \_ALTERA\_AVALON\_PIO\_REGS\_H
- io.h
- IOADDR\_ALTERA\_AVALON\_PIO\_DATA()
- IORD\_ALTERA\_AVALON\_PIO\_DATA()
- IOWR\_ALTERA\_AVALON\_PIO\_DATA()
- IOADDR\_ALTERA\_AVALON\_PIO\_DIRECTION()
- IORD\_ALTERA\_AVALON\_PIO\_DIRECTION()
- IOWR\_ALTERA\_AVALON\_PIO\_DIRECTION()
- IOADDR\_ALTERA\_AVALON\_PIO\_IRQ\_MASK()
- IORD\_ALTERA\_AVALON\_PIO\_IRQ\_MASK()
- IOWR\_ALTERA\_AVALON\_PIO\_IRQ\_MASK()
- IOADDR\_ALTERA\_AVALON\_PIO\_EDGE\_CAP()
- IORD\_ALTERA\_AVALON\_PIO\_EDGE\_CAP()
- IOWR\_ALTERA\_AVALON\_PIO\_EDGE\_CAP()
- IOADDR\_ALTERA\_AVALON\_PIO\_SET\_BITS()
- IORD\_ALTERA\_AVALON\_PIO\_SET\_BITS()
- IOWR\_ALTERA\_AVALON\_PIO\_SET\_BITS()
- IOADDR\_ALTERA\_AVALON\_PIO\_CLEAR\_BITS()
- IORD\_ALTERA\_AVALON\_PIO\_CLEAR\_BITS()
- IOWR\_ALTERA\_AVALON\_PIO\_CLEAR\_BITS()
- ALTERA\_AVALON\_PIO\_DIRECTION\_INPUT
- ALTERA\_AVALON\_PIO\_DIRECTION\_OUTPUT

Problems Tasks Console Nios II Console Properties

mythirdnios Nios II Hardware configuration - cable: DE SoC on localhost [USB-1] device ID: 2 instance ID: 0 name: jtag\_uart/jtag

Hello from Nios II!

Writable



# HAL References

- Each HAL project references library routines and drivers for the components included in your Nios II system

