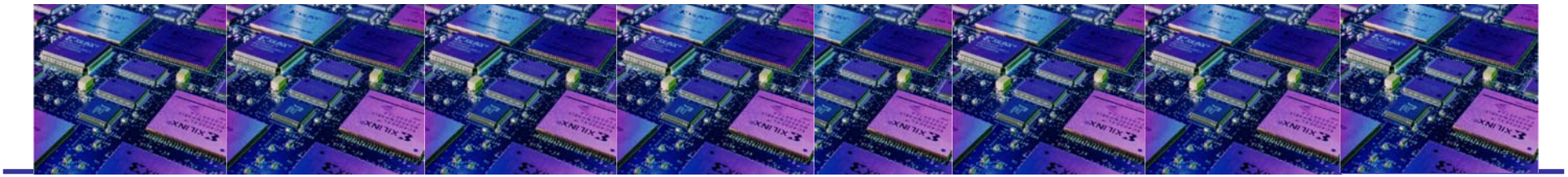


Les Circuits Reconfigurables : Passé, Présent, Futur ….

Lionel TORRES
LIRMM/POLYTECH, Université de Montpellier II
torres@lirmm.fr
http://www.lirmm.fr

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

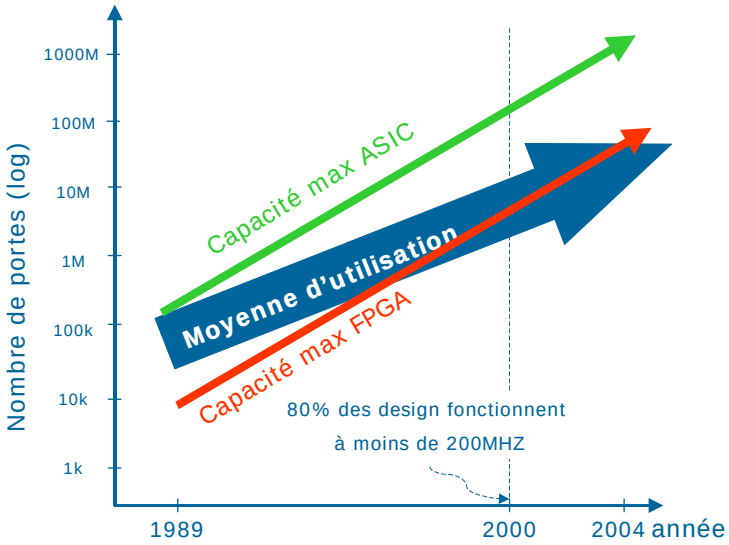
Polytech'Montpellier
Réseau Polytech'

LIRMM

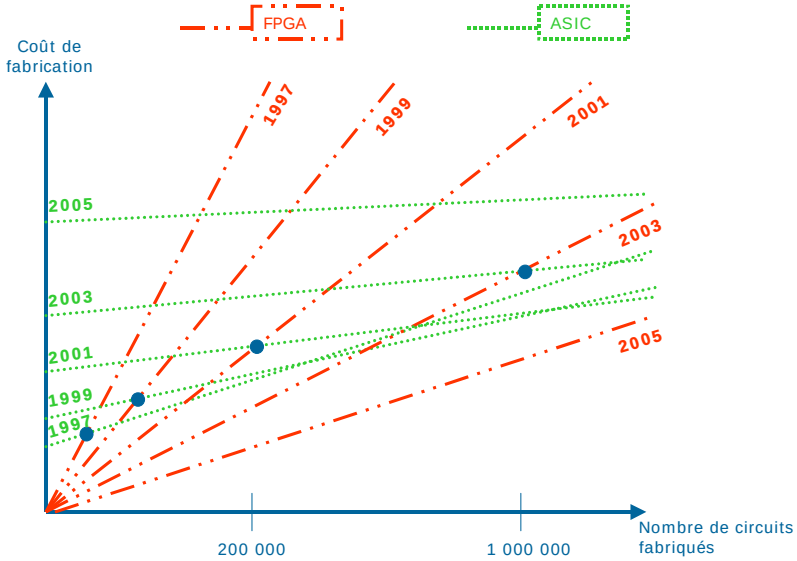
UNIVERSITÉ MONTPELLIER II
SCIENCES ET TECHNIQUES DU LANGUEDOC

Pourquoi les architectures reconfigurables ?

- Les architectures reconfigurables
  - importance des travaux du domaine
    - FPGA, Systolic Ring, DART, aSoC, Garp, PipeRench, Chameleon, RAW, KressArray, Morphosys, FPFA ...
  - les FPGA sont aujourd'hui de réelles alternatives aux ASIC (capacité, prix et performances en adéquation avec les besoins)



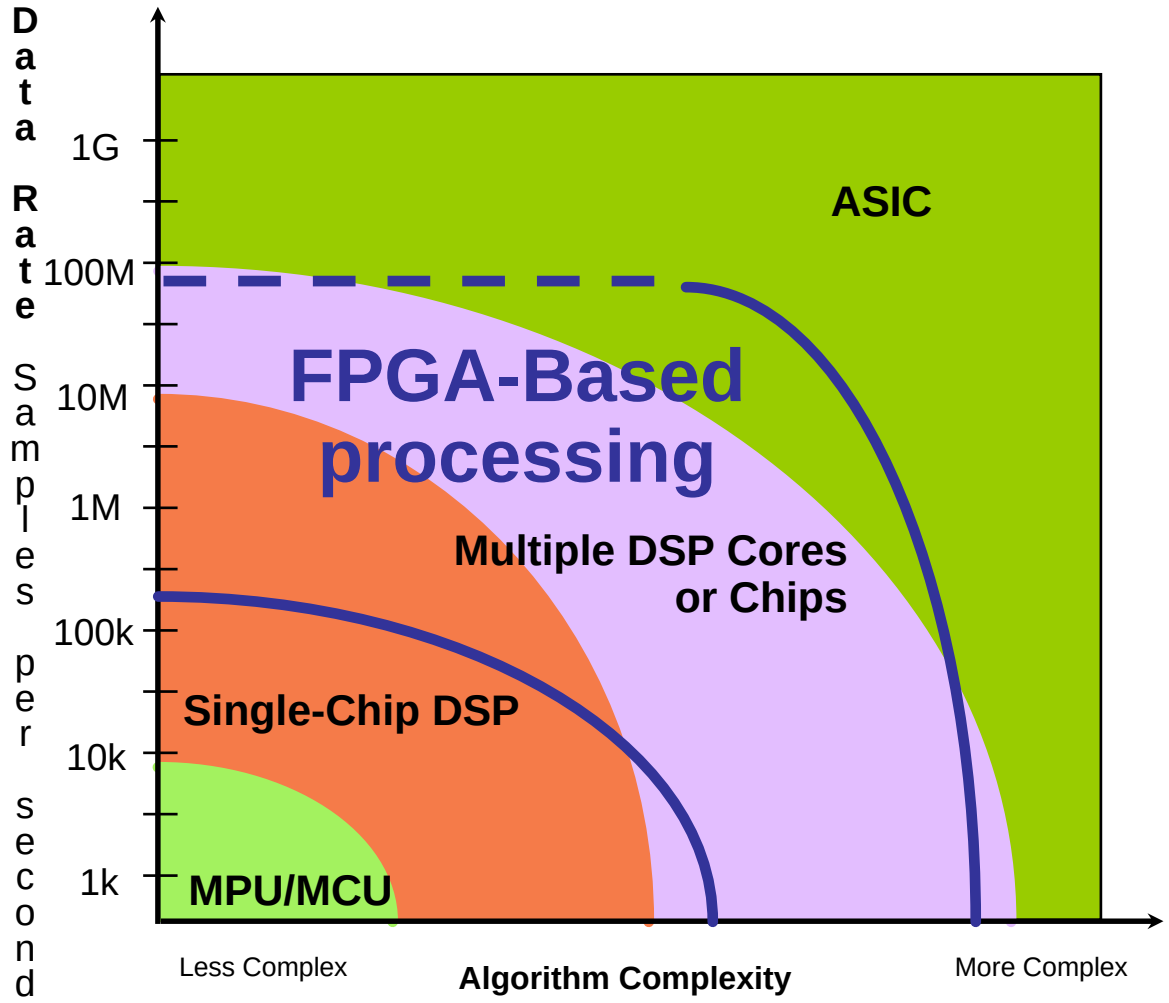
Source : Altera, SOPC World Seminar 2001



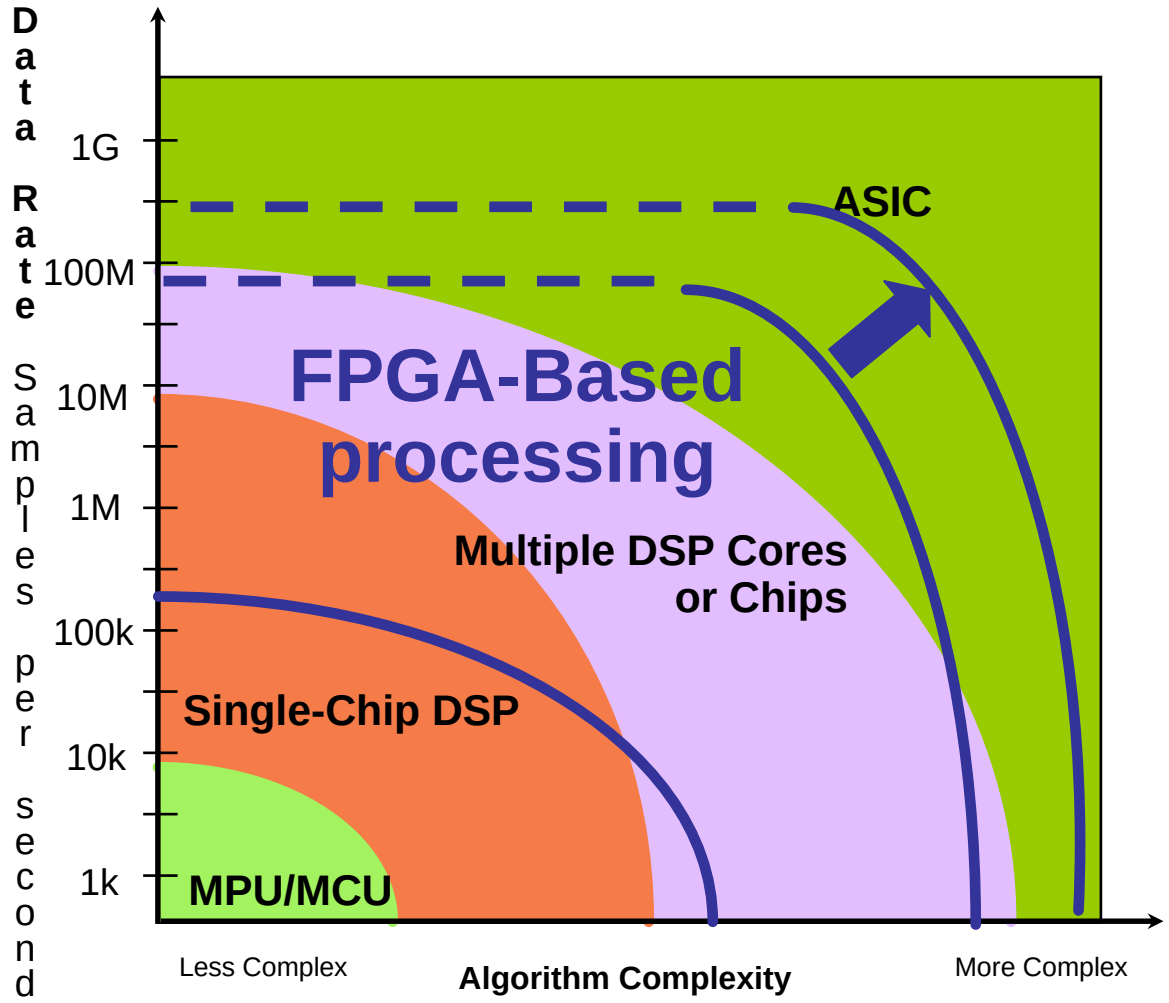
S. Tredennick, The Rise of Reconfigurable Systems, ERSA 2003

Source : L. Bossuet, G. Gogniat - LESTER

Pourquoi les architectures reconfigurables ?



Pourquoi les architectures reconfigurables ?



Sommaire

Partie 1 : Circuits reconfigurables « standards »

1- PRINCIPES et CLASSIFICATION
2- TECHNOLOGIES
3- ARCHITECTURES ET CIRCUITS
4- CONCEPTION

Partie 2 : Les architectures reconfigurables dynamiquement

1- INTRODUCTION & CONTEXTE
2- CLASSIFICATION D'ARCHITECTURES
3- QUELQUES EXEMPLES
4- EXEMPLE DE GESTION DYNAMIQUE
5- CONCLUSIONS & PERSPECTIVES

1- PRINCIPES et CLASSIFICATION

DEFINITION :

COMPOSANTS STANDARDS PROGRAMMABLES ELECTRIQUEMENT –

UNE SEULE FOIS (FUSIBLES)

OU

PLUSIEURS FOIS

RE-PROGRAMMABLES (RECONFIGURATION)

1- PRINCIPES et CLASSIFICATION

PRINCIPES DES ARCHITECTURES:

ENSEMBLE DE RESSOURCES LOGIQUES (PORTES, BASCULES, ...ETC) QUI PEUVENT ÊTRE INTERCONNECTÉES DE DIFFÉRENTES FAÇONS.

REALISATION DE FONCTIONS BOOLEENNES SOUS FORME D'UNE SOMME LIMITEE DE MONOMES (PAL, PLD, EPLD,...) OU D'UN RESEAU DE CELLULES (FPGA)

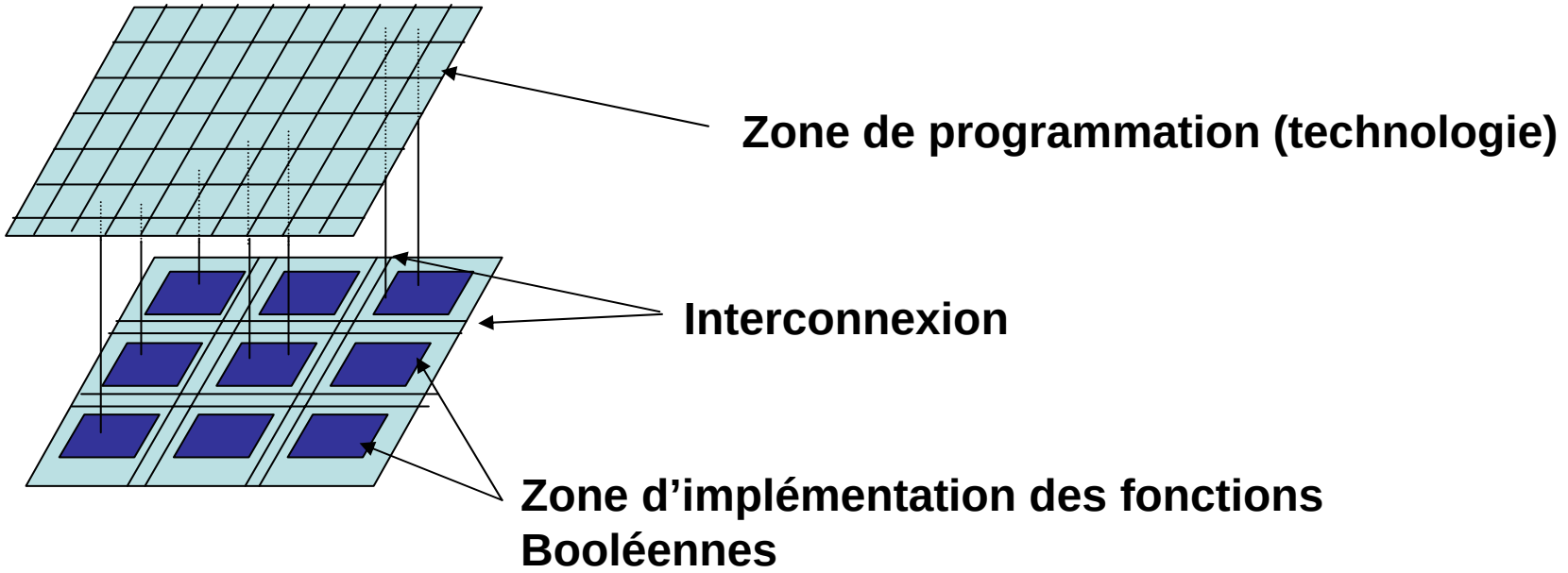
1- PRINCIPES et CLASSIFICATION

TECHNOLOGIES DE PROGRAMMATION :

FUSIBLES (METAL), ANTIFUSIBLES (CAPACITE MOS), TRANSISTOR MOS A GRILLE FLOTANTE (EPLD), RAM STATIQUE (FPGA-SRAM),...

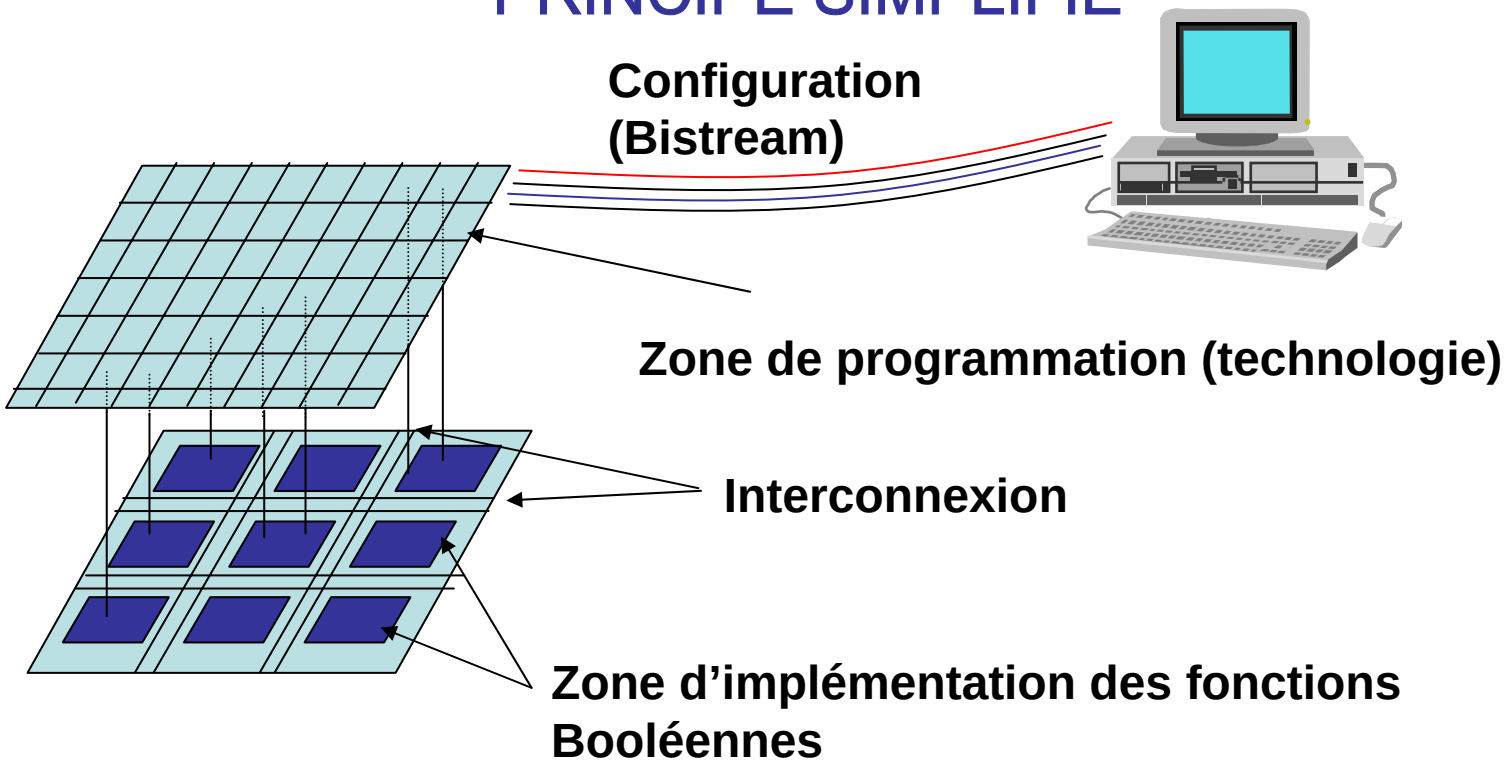
1- PRINCIPES et CLASSIFICATION

PRINCIPE SIMPLIFIE



1- PRINCIPES et CLASSIFICATION

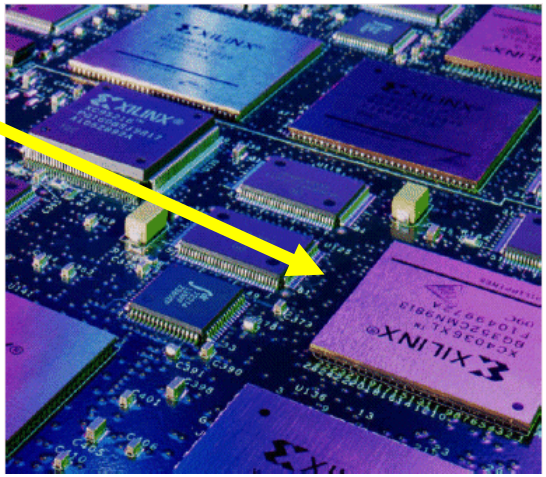
PRINCIPE SIMPLIFIE



1- PRINCIPES et CLASSIFICATION

- Le bitstream décrit la configuration de tous les éléments configurables du circuits
- Un transfert de bitstream est nécessaire lors de la mise sous tension et à chaque reconfiguration

10010010011110010110



1- PRINCIPES et CLASSIFICATION

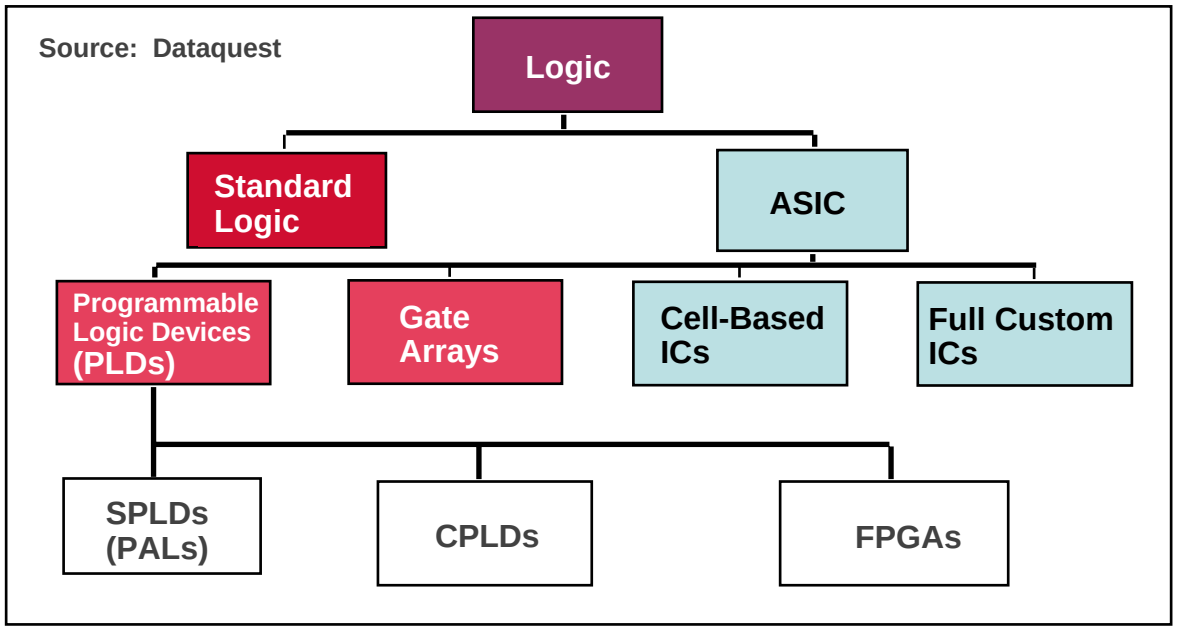
GLOSSAIRE

ASIC : Application Spécific Integrated Circuit
circuit intégré pour application spécifique
SOC : System On Chip
Système sur puce
IP : Intellectual Propertie (bloc IP ou « IP core ») ou Composant virtuel
propriété intellectuelle
FPGA : Field Programmable Gate Array
réseau de portes programmables
PLD : Programmable Logic Devices

CPLD : Complex Programmable Logic Devices

1- PRINCIPES et CLASSIFICATION

CLASSIFICATION



Acronyms

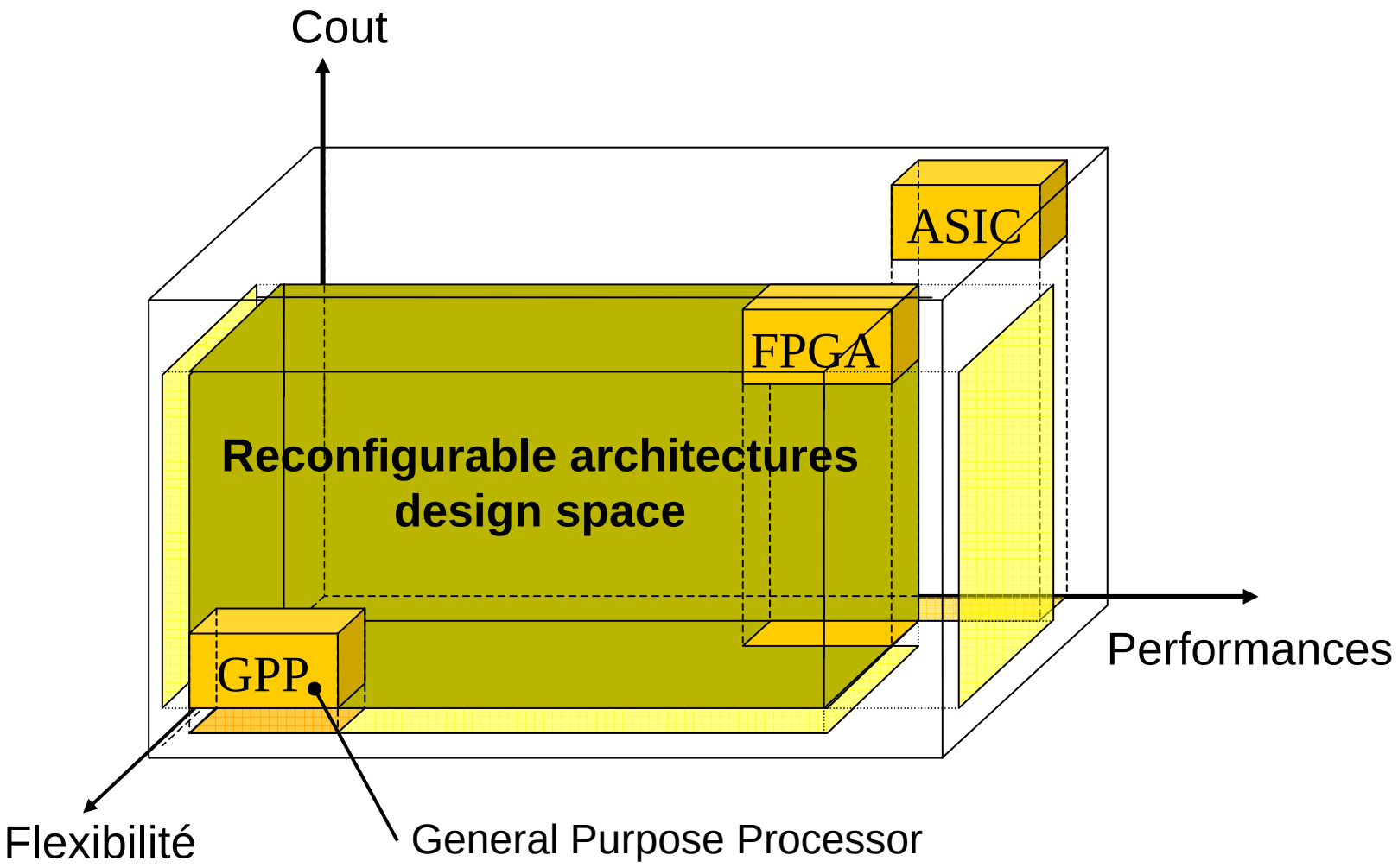
SPLD = Simple Prog. Logic Device
PAL = Prog. Array of Logic
CPLD = Complex PLD
FPGA = Field Prog. Gate Array

Common Resources

Configurable Logic Blocks (CLB)
- Memory Look-Up Table
- AND-OR planes
- Simple gates

Input / Output Blocks (IOB)
- Bidirectional, latches, inverters

1- PRINCIPES et CLASSIFICATION



Sommaire

Partie 1 : Circuits reconfigurables « standards »

1- PRINCIPES et CLASSIFICATION
2- TECHNOLOGIES
3- ARCHITECTURES ET CIRCUITS
4- CONCEPTION

2- TECHNOLOGIES

Technologies de programmation

- Différentes technologies pour stocker la configuration

- Fuse
- Antifuse

} Programmable une seule fois (configurable)

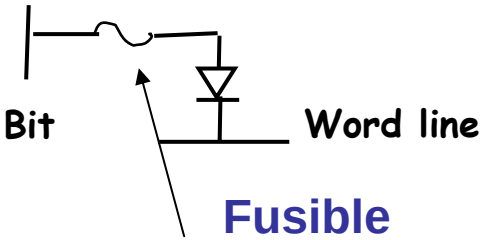
- (E)EPROM } Nombre de configuration limité (x.10^3 à x. 10^6)
- SRAM } Configuration réalisée à chaque mise sous tension

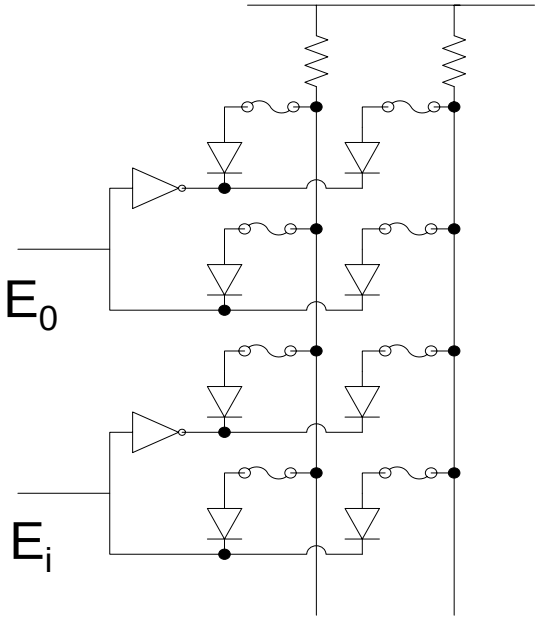
2- TECHNOLOGIES

Technologie Fusible

Historique (préhistoire : il y a 25 ans) : PLD Bipolaires

- ✓ Technologie comparable à celle des PROMs
- ✓ réseaux de portes avec connexions programmables
- ✓ programmation par fusibles (comme les PROMs)



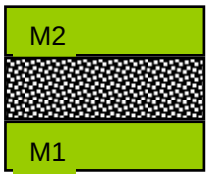


- ✓ non reprogrammables
- ✓ fonctions logiques simples (structures de type *Sum-of-Products*)
- ✓ coût unitaire élevé (faibles séries, prototypes, …)

2- TECHNOLOGIES

- Anti fusible = condensateur

  Actel

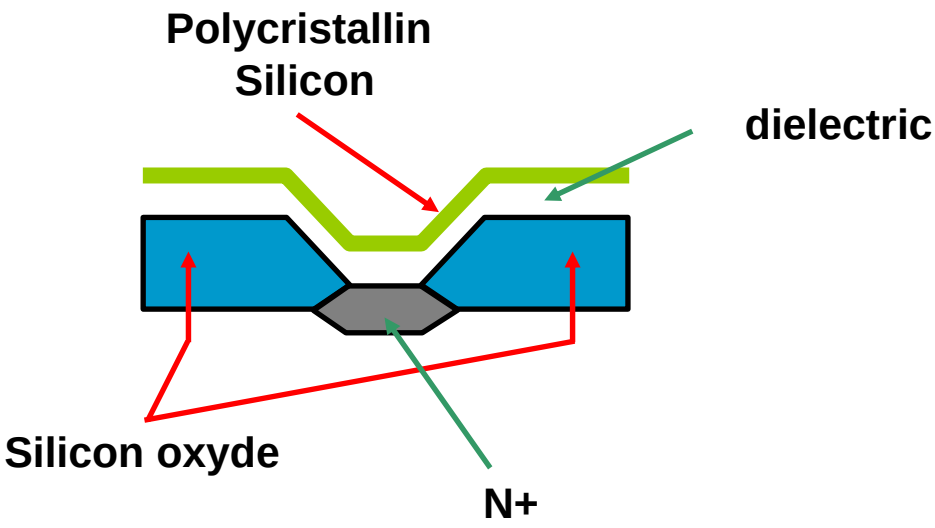


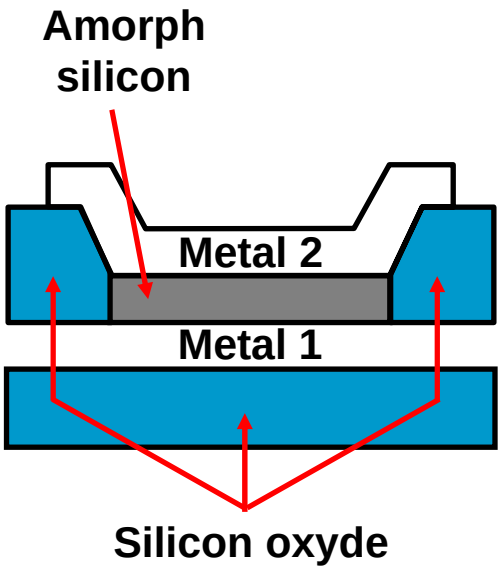
  - Création d'un court circuit entre deux lignes de métal: claquage
  - Programmation définitive
  - Très peu de place occupée sur le circuit, mais étapes de fabrication supplémentaires
  - performances électriques supérieures à la technologie « SRAM » (minimisation de la surface et des effets RC)

2- TECHNOLOGIES

Antifuse (PLDs, CPLDs)

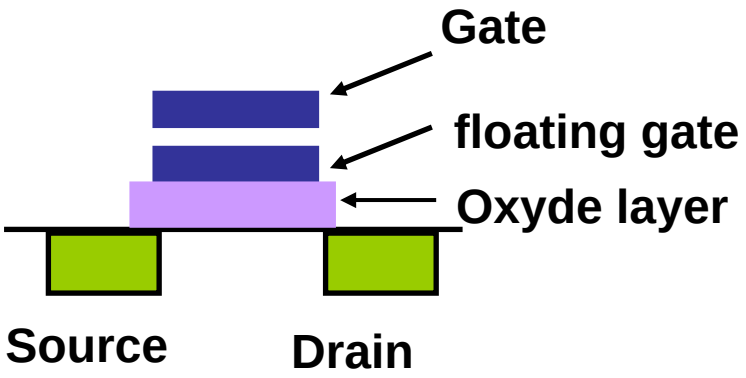
Programmable une fois





2- TECHNOLOGIES

Electrically Erasable Programmable Read-Only Memory (EPLD, CPLD)



transistor à grille flottante

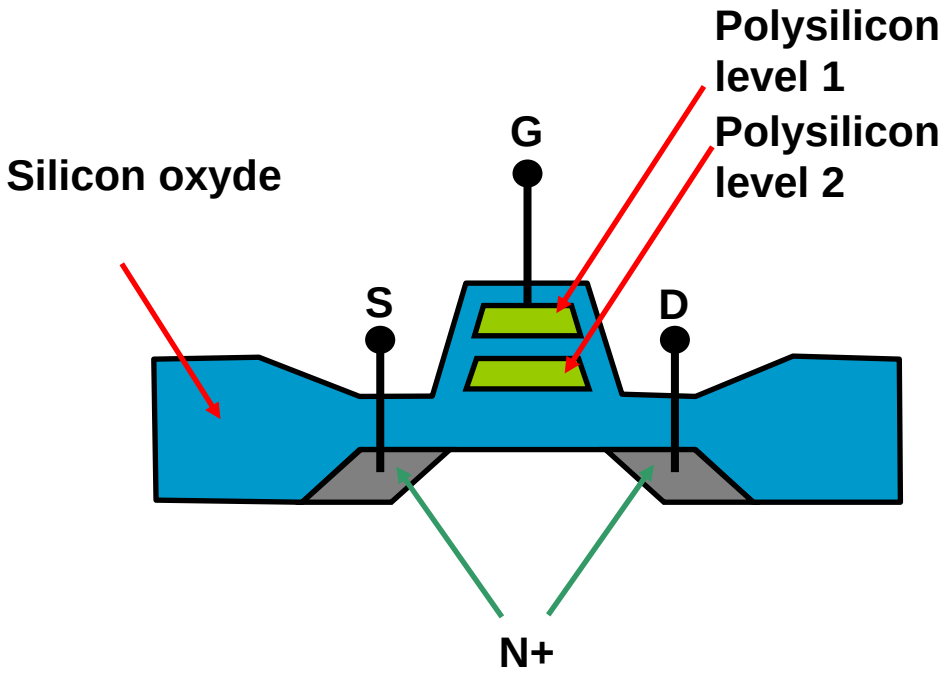
L 'application d 'un potentiel sur la grille supérieure provoque le passage d 'une partie des électrons du canal à travers la mince couche d 'oxyde, ce qui charge la grille flottante. Lors de la lecture, une tension appliquée sur la grille supérieure est complètement masquée par la charge négative emmagasinée sur la grille flottante. Cela équivaut à un transistor toujours bloqué.

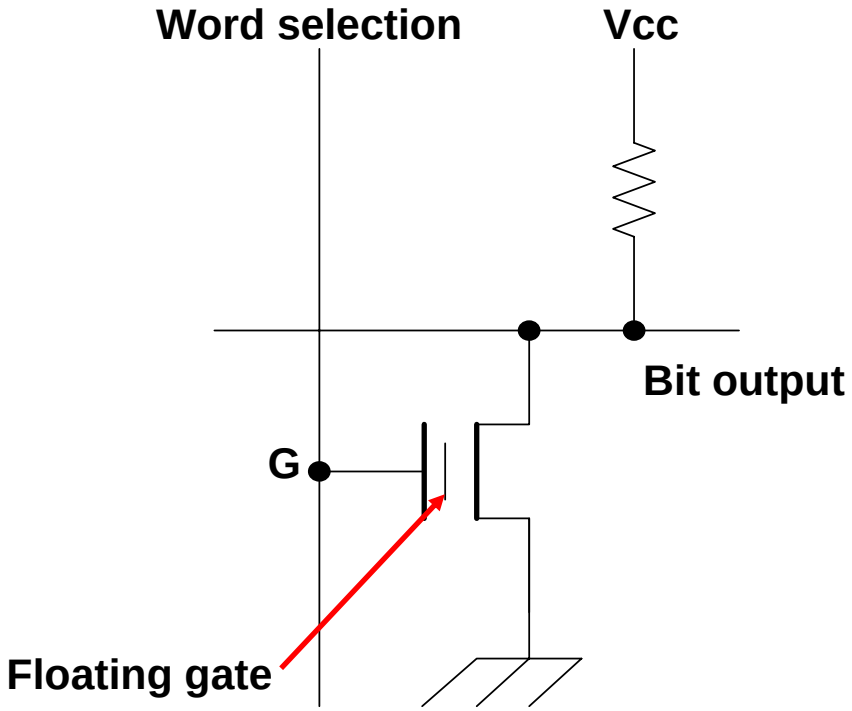
- même technologie que celle des mémoires EPROM
- transistor à double grille
- reprogrammable (effacement par UV ou électriquement)

2- TECHNOLOGIES

Electrically Erasable Programmable Read-Only Memory

Nombre limité de programmation

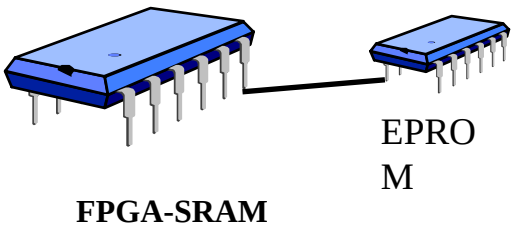


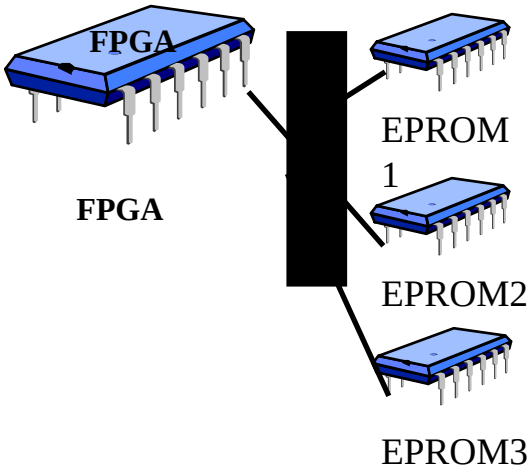


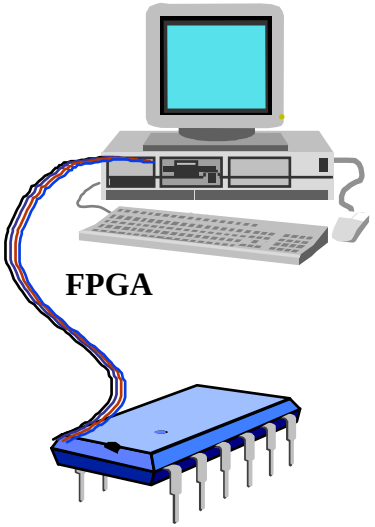
2- TECHNOLOGIES

Technologie SRAM (FPGA)

- **Technologie CMOS standard**
- Portes de transmission ou multiplexeurs commandés par des cellules SRAM
- Les mémoires SRAM permettent de configurer les interconnexions et de programmer les cellules
- le FPGA doit être configuré à chaque mise sous tension à partir d'une mémoire externe (EPROM)



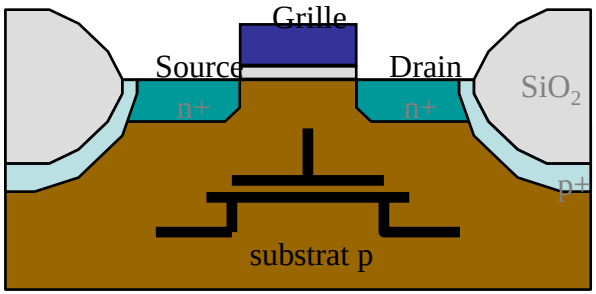




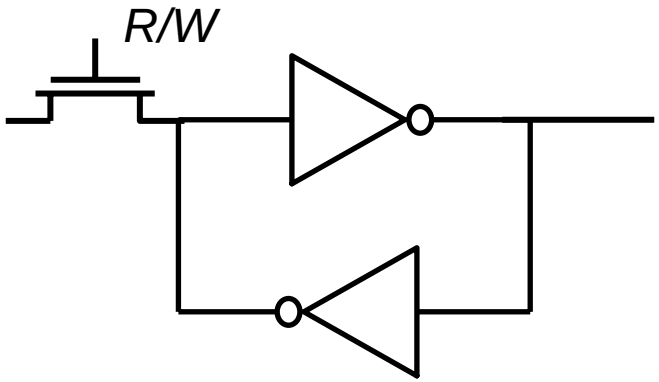
2- TECHNOLOGIES

Technologie SRAM (FPGA)

- Programmation illimitée
- Programmation à chaque mise en tension

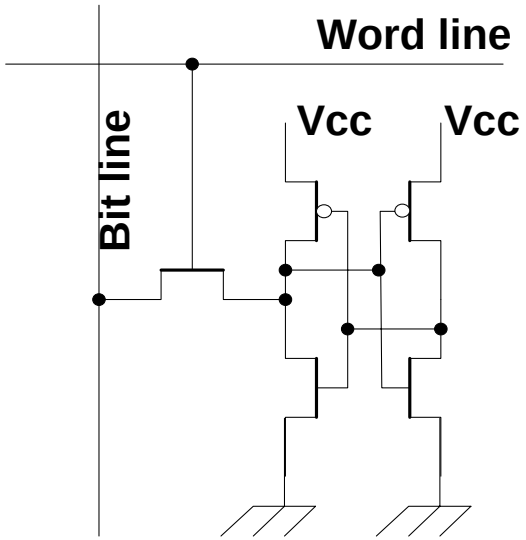


SRAM

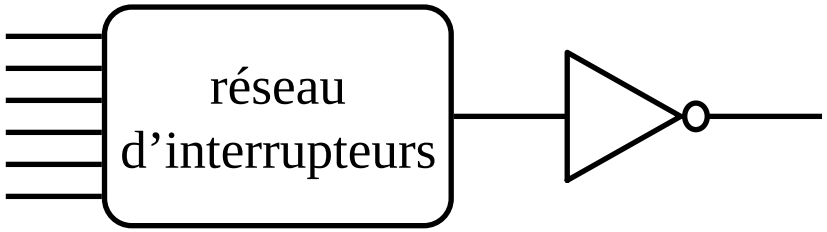


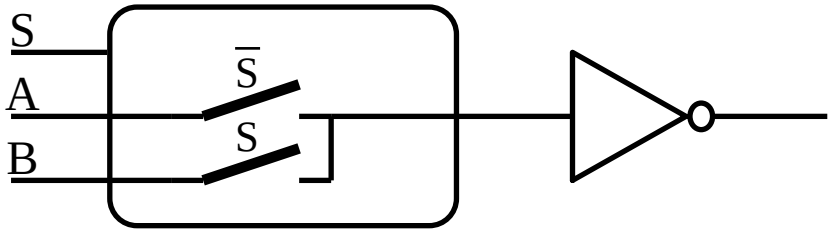
Point mémoire

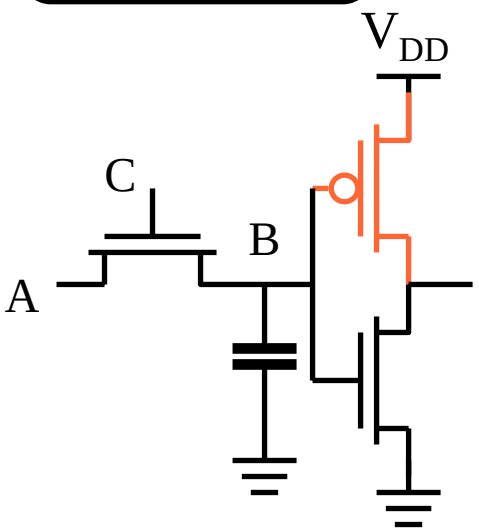
5 transistors SRAM

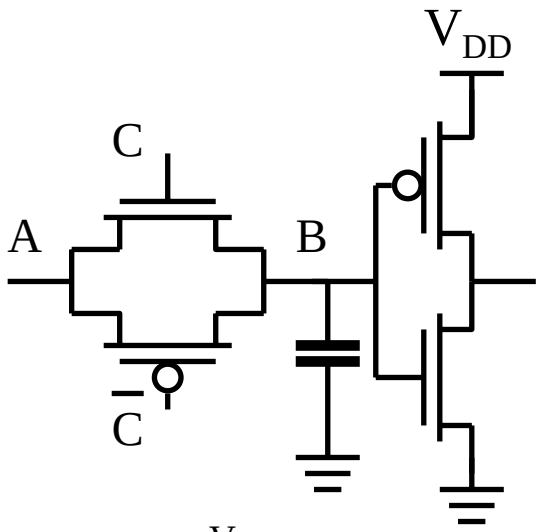


2- TECHNOLOGIES

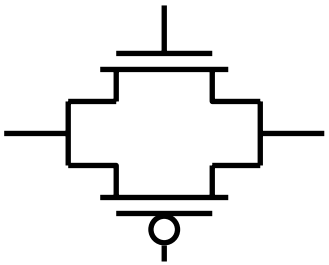


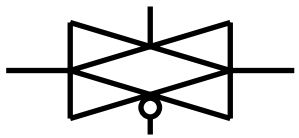


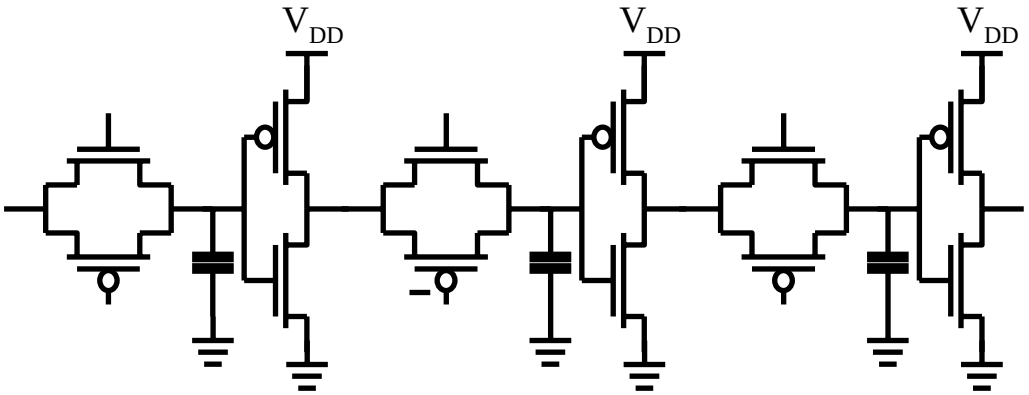




Porte de transmission



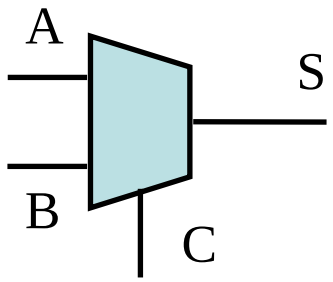




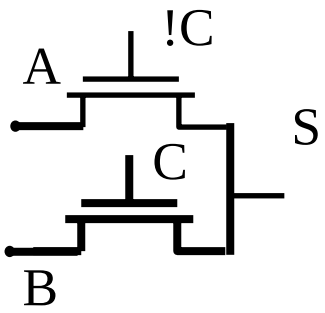
Registre à décalage

2- TECHNOLOGIES

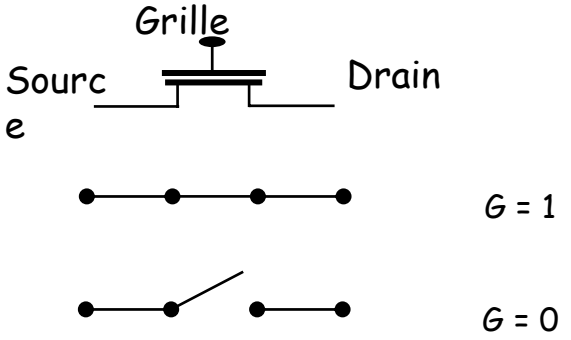
Rappel : Réalisation de fonctions logiques avec des multiplexeurs MOS



Si C=0, alors S=A
Si C=1, alors S=B



Exemples:
S=X.Y si C= X, A =0, B = Y
S=X+Y si C= !X, A =1, B = X



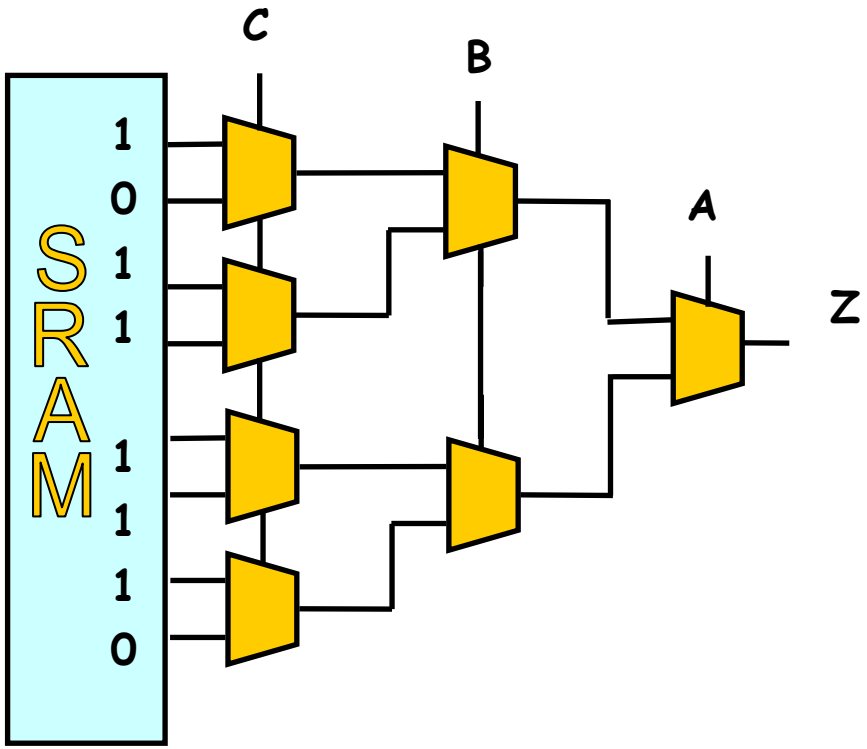
Transistor NMOS = commutateur

Réalisation d'une LUT

A	B	C	Z
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

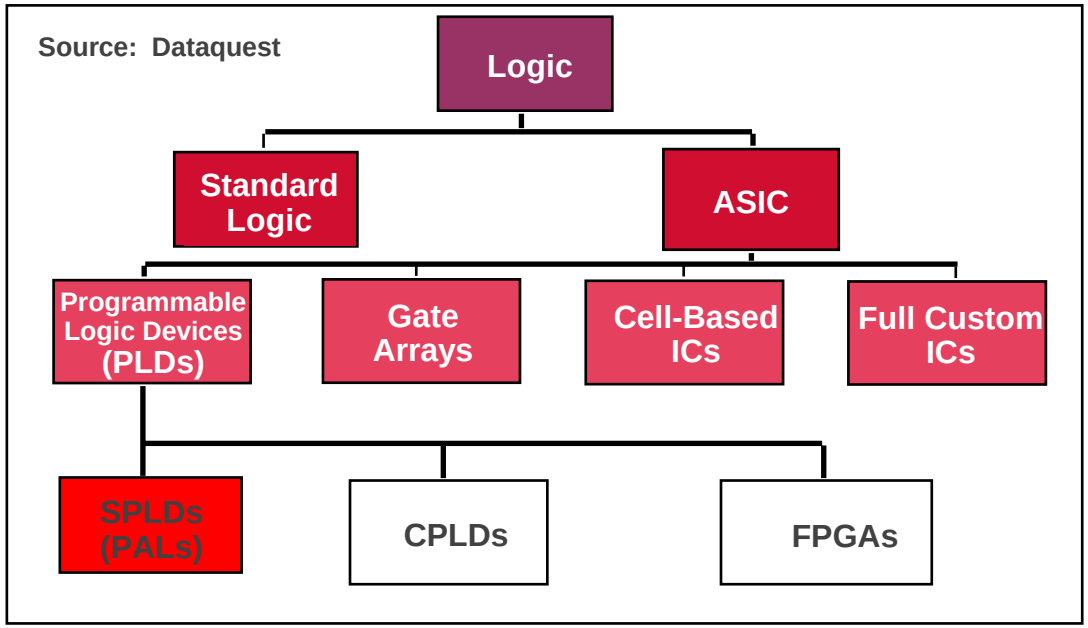
« Look Up Table »

LUT 3



3- ARCHITECTURE ET CIRCUITS

Programmable Logic Device Families



Acronyms

SPLD = Simple Prog. Logic Device
PAL = Prog. Array of Logic
CPLD = Complex PLD
FPGA = Field Prog. Gate Array

Common Resources

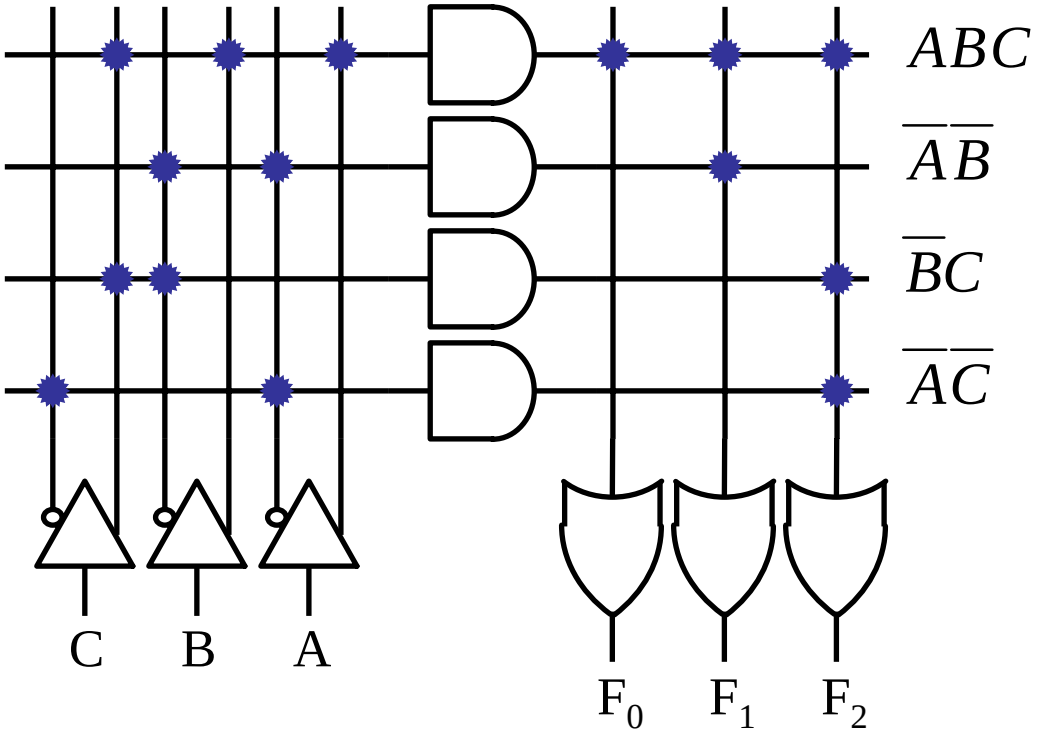
Configurable Logic Blocks (CLB)
- Memory Look-Up Table
- AND-OR planes
- Simple gates

Input / Output Blocks (IOB)
- Bidirectional, latches, inverters

3- ARCHITECTURE ET CIRCUITS

PLAs : Programmable Logic Arrays (préhistoire)

- Plans AND-OR : Somme de produits



$$F_0 = ABC$$

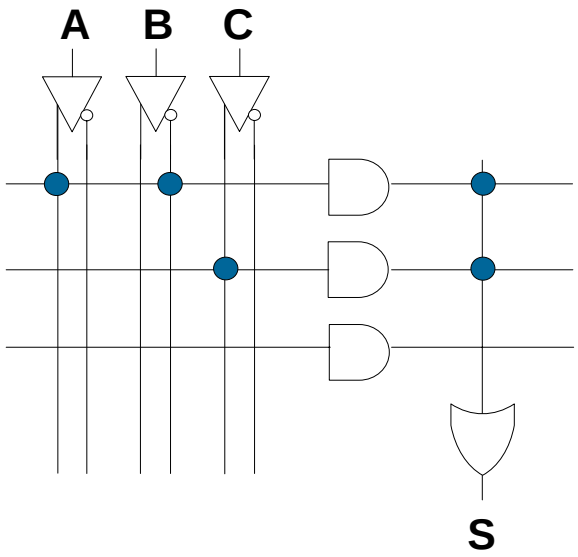
$$F_1 = ABC + \overline{A}\overline{B}$$

$$F_2 = ABC + \overline{B}C + \overline{A}\overline{C}$$

PLAs : Examples

AND-OR plane

S = A.B+C

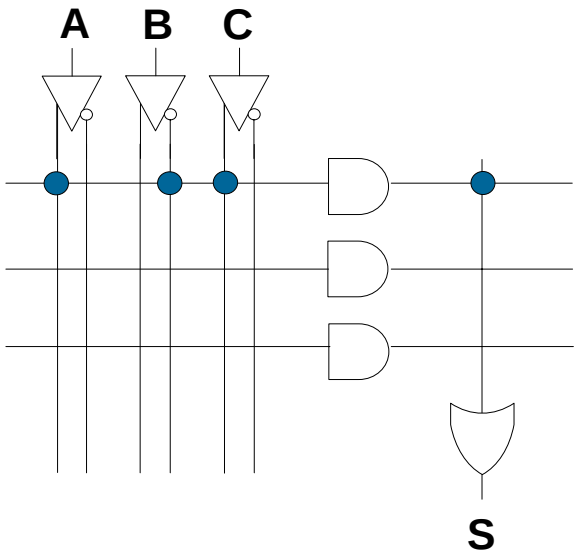


A	B	C	S
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

PLAs : Examples

AND-OR plane

S = A.B.C

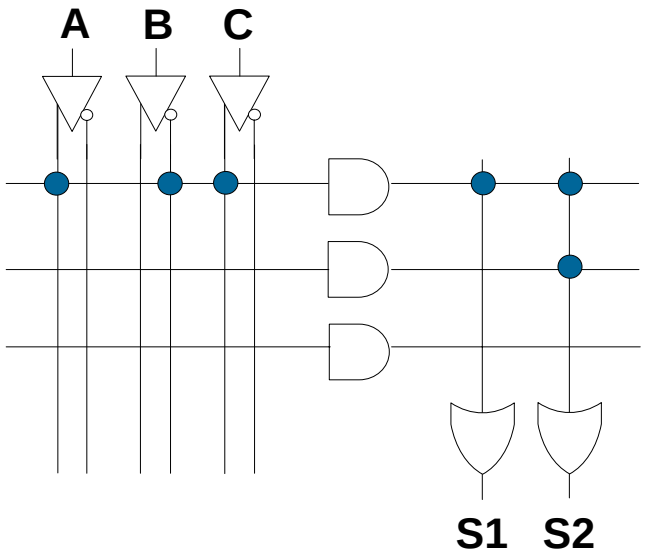


A	B	C	S
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

PLAs : Examples

AND-OR plane

S1 = A.B.C , S2 = A.B+C

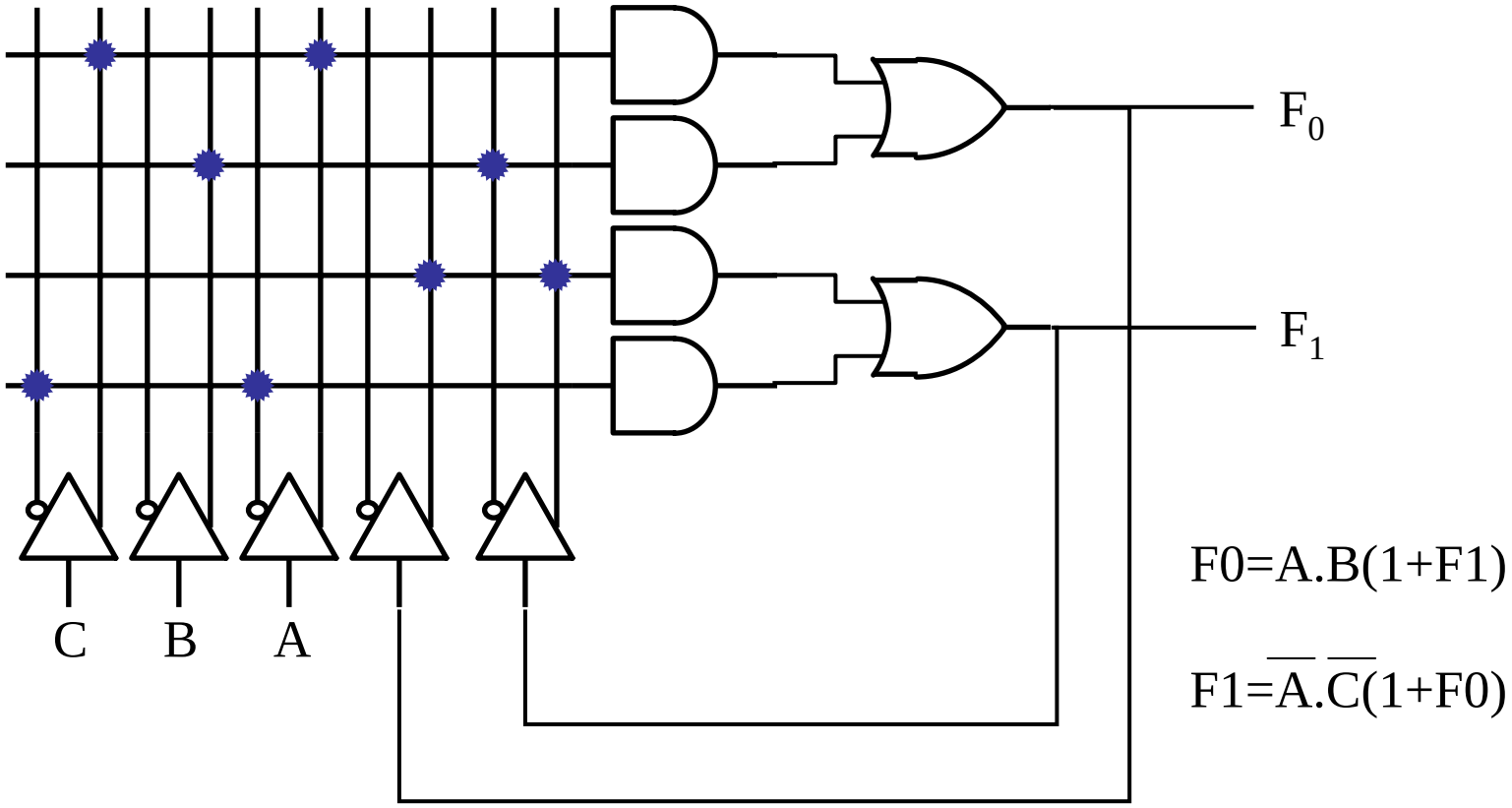


A	B	C	S1
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

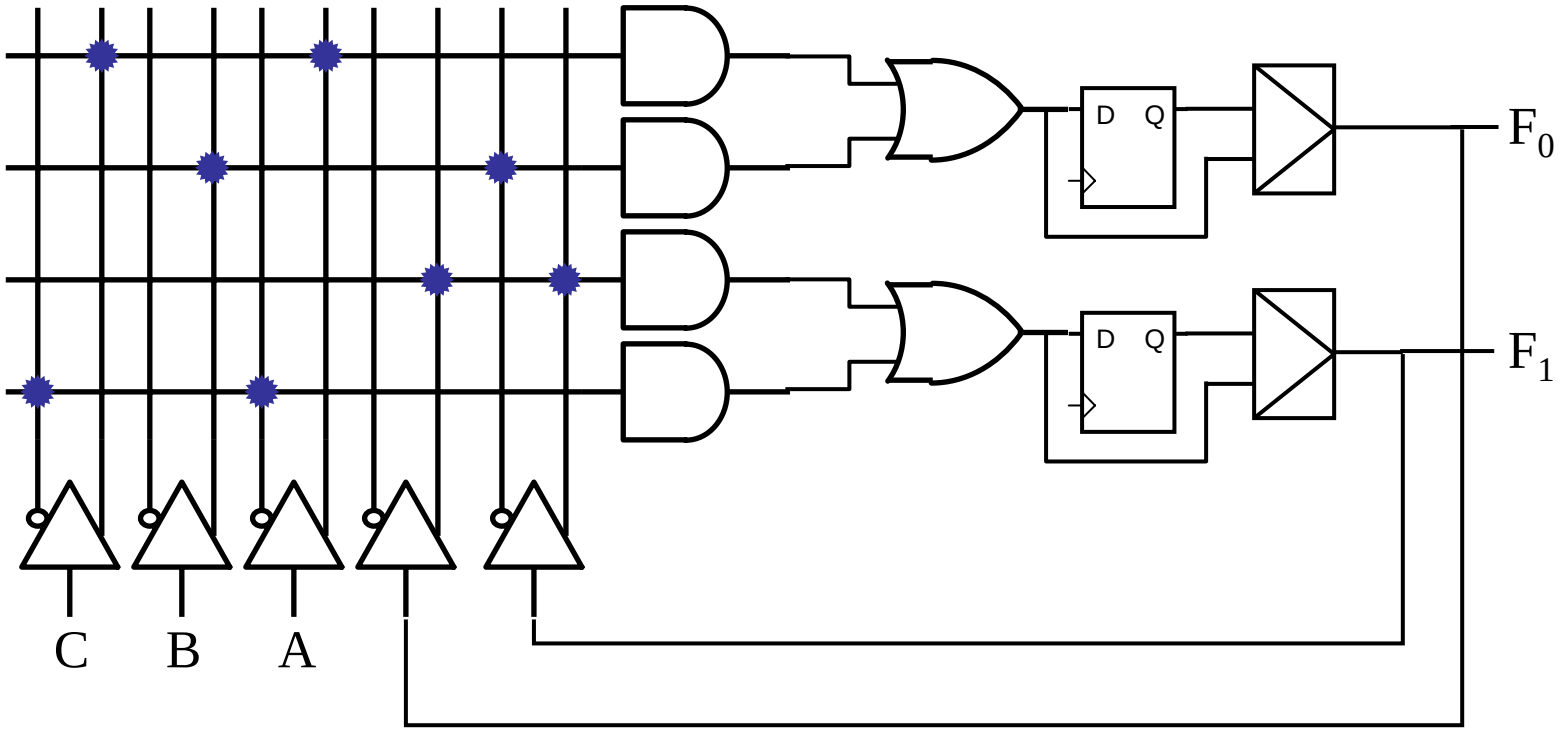
A	B	C	S2
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

3- ARCHITECTURE ET CIRCUITS

PALs : Programmable Arrays of Logic



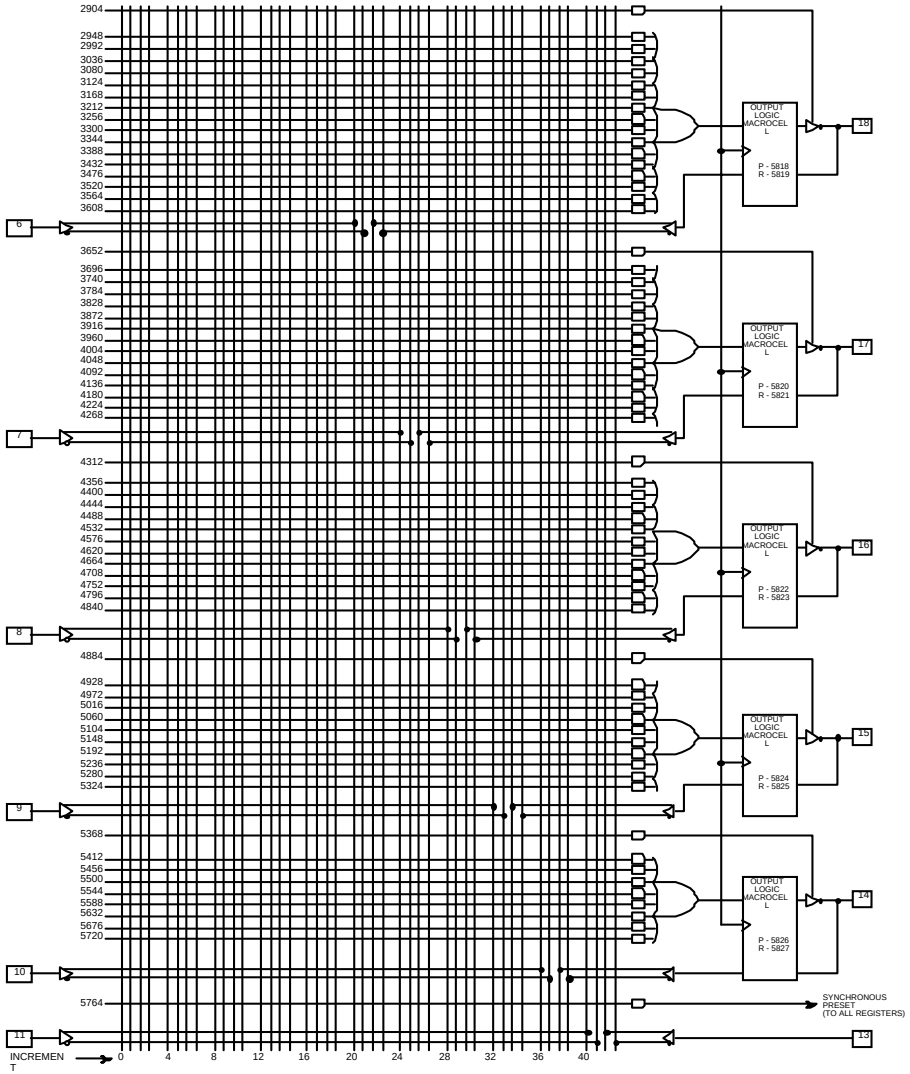
SPLDs : Simple Programmable Logic Devices



Feedback terms

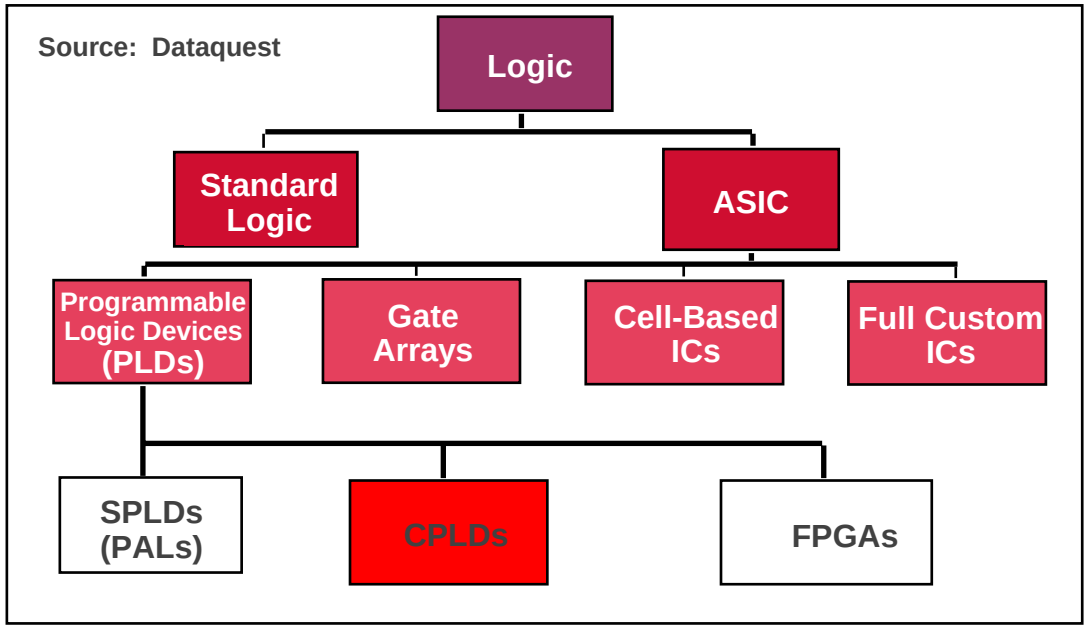
3- ARCHITECTURE ET CIRCUITS

SPLDs : Simple Programmable Logic Devices



3- ARCHITECTURE ET CIRCUITS

Programmable Logic Device Families



Acronyms

SPLD = Simple Prog. Logic Device
PAL = Prog. Array of Logic
CPLD = Complex PLD
FPGA = Field Prog. Gate Array

Common Resources

Configurable Logic Blocks (CLB)
- Memory Look-Up Table
- AND-OR planes
- Simple gates

Input / Output Blocks (IOB)
- Bidirectional, latches, inverters

3- ARCHITECTURE ET CIRCUITS

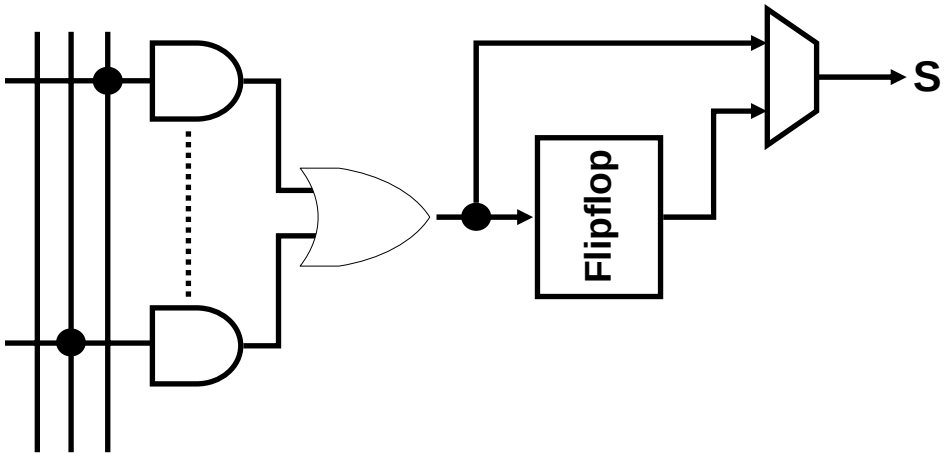
CPLDs : Complex Programmable Logic Devices

Macrocells : Somme de produits + sequ.elements (flipflops)

Utilisation de PLD

Surface importante mais :

- Nombre d'entrées
- Nombre de somme de produits

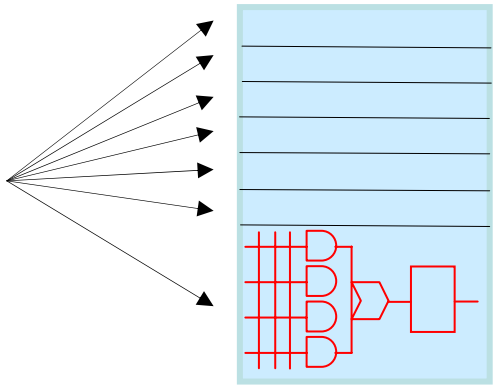


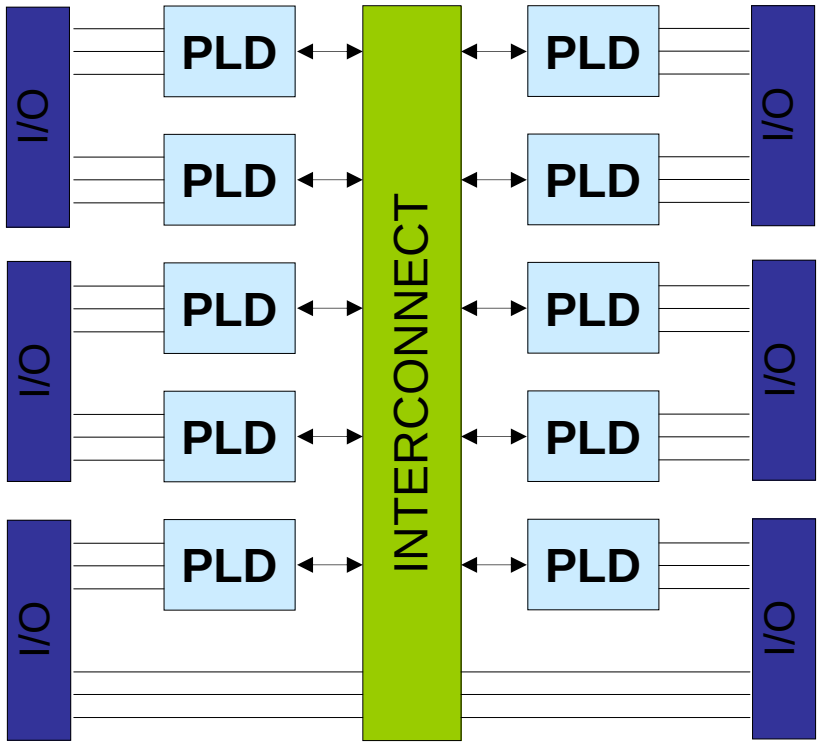
3- ARCHITECTURE ET CIRCUITS

CPLDs : Complex Programmable Logic Devices

Plusieurs PLD
Interconnexion via a *switch matrix*

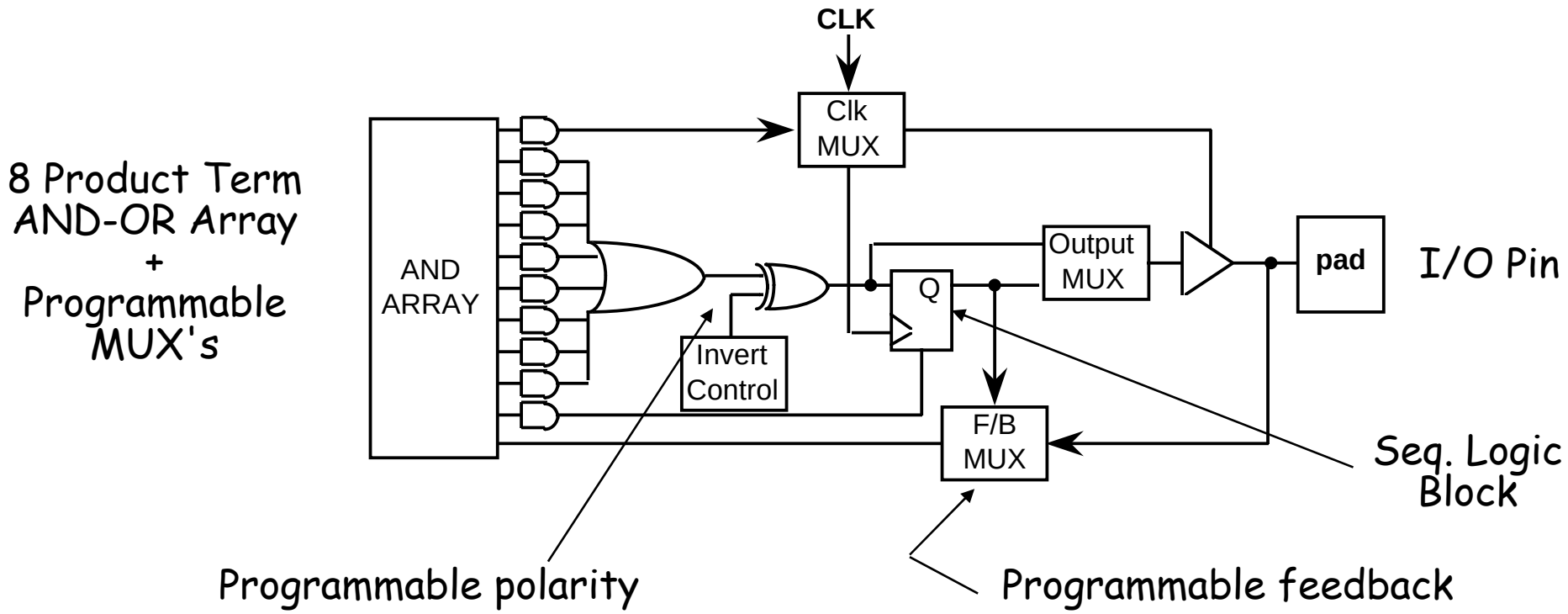
1 block:
Plusieurs
Macrocells





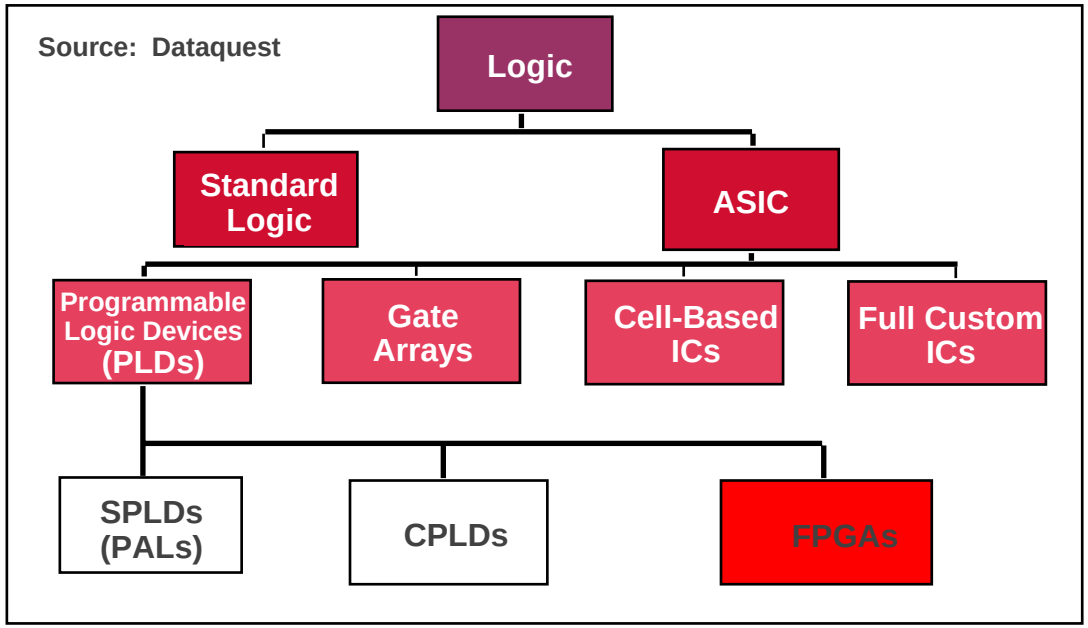
CPLDs : Complex Programmable Logic Devices

Altera MAX 7000 Macrocell structure



3- ARCHITECTURE ET CIRCUITS

Programmable Logic Device Families



Acronyms

SPLD = Simple Prog. Logic Device
PAL = Prog. Array of Logic
CPLD = Complex PLD
FPGA = Field Prog. Gate Array

Common Resources

Configurable Logic Blocks (CLB)
- Memory Look-Up Table
- AND-OR planes
- Simple gates

Input / Output Blocks (IOB)
- Bidirectional, latches, inverters

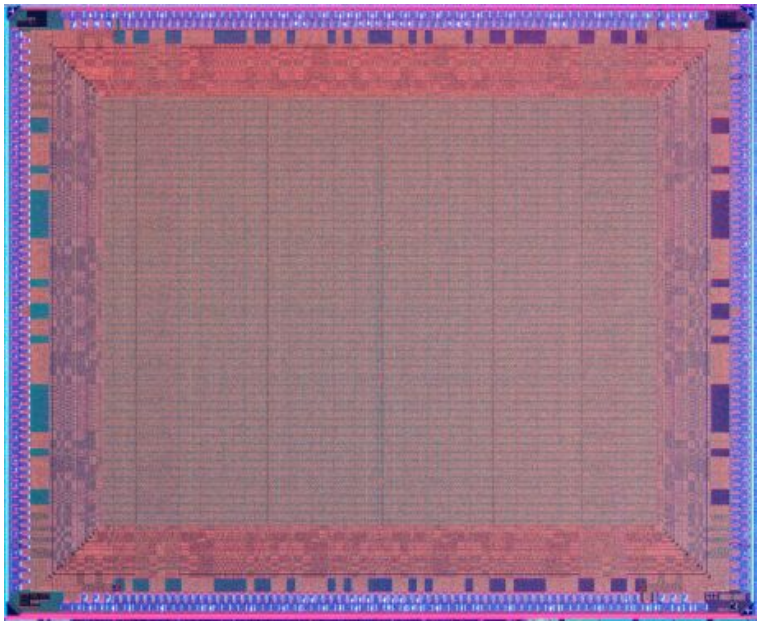
3- ARCHITECTURES ET CIRCUITS

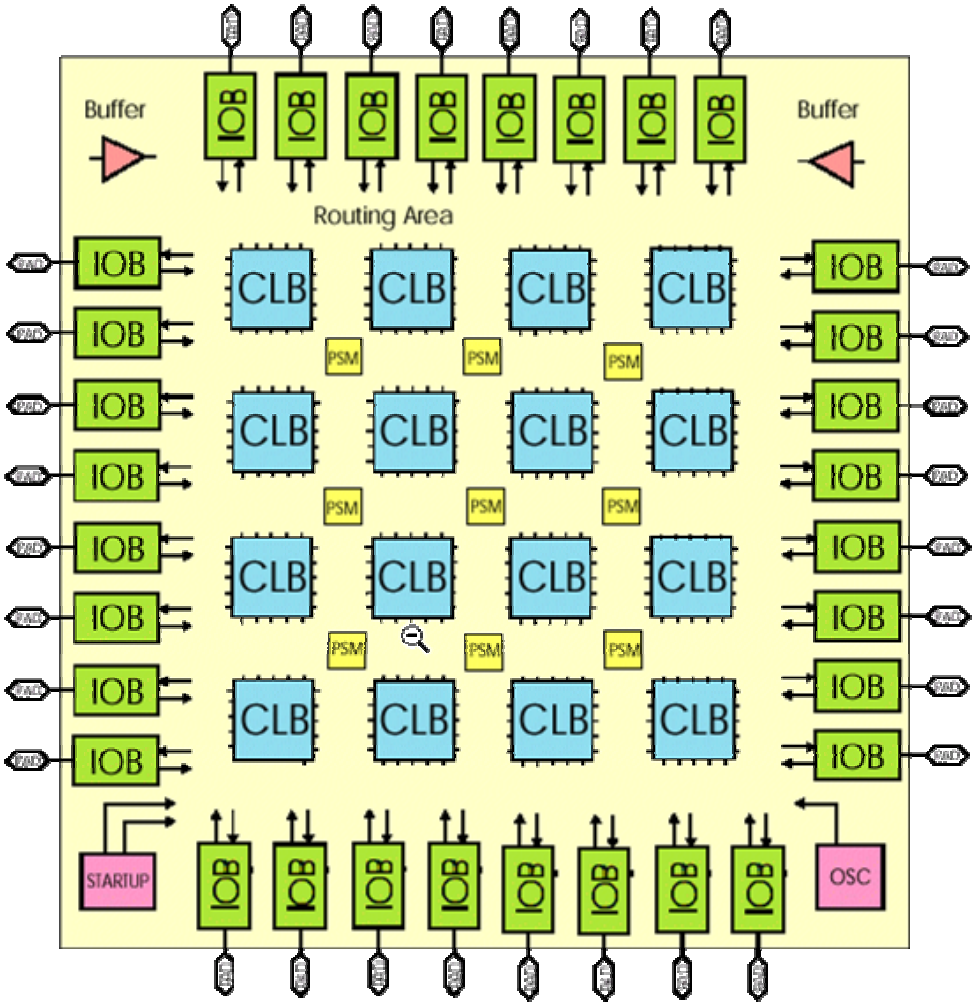
FPGAs : Field Programmable Gate Arrays

Matrice de blocs reconfigurables

Interconnexion reconfigurable

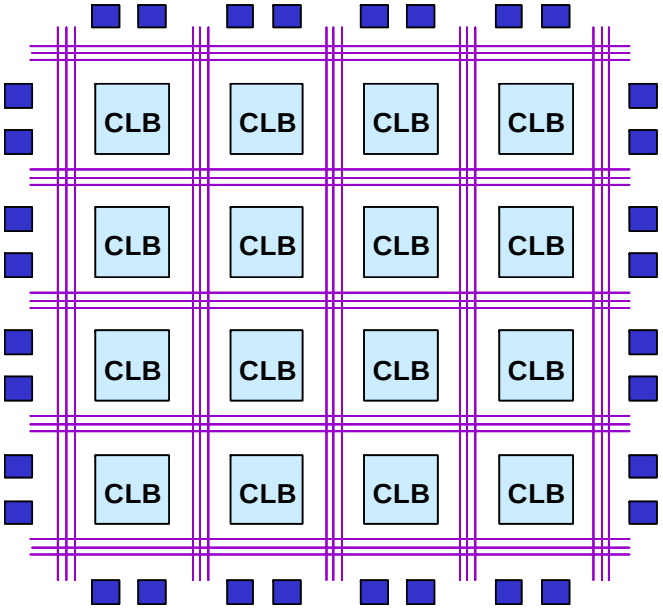
CLB = LUT + Sequentiel éléments





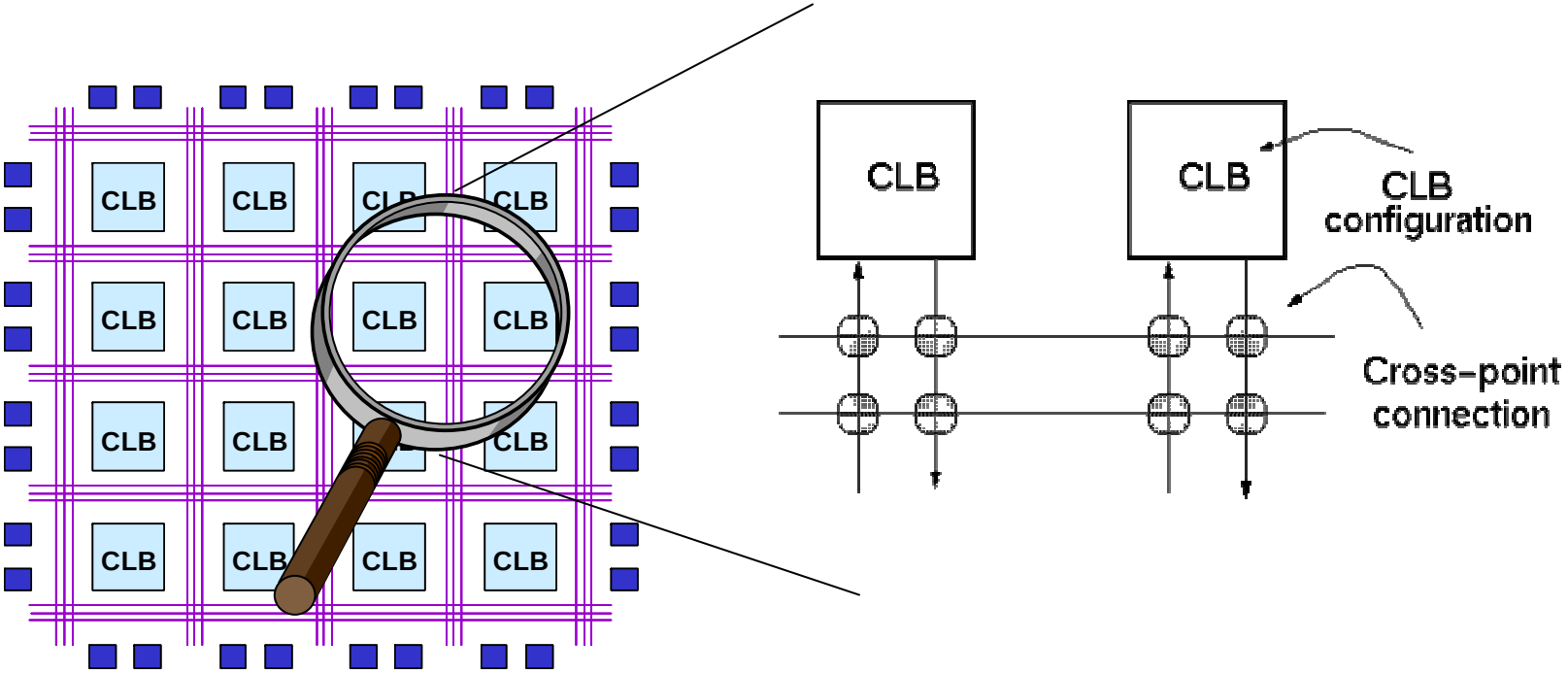
FPGAs : Reconfigurable

- Arrangement Matriciel de blocs logiques avec configuration des :
    1. Interconnexions entre les blocs logiques,
    2. La fonction de chaque bloc.



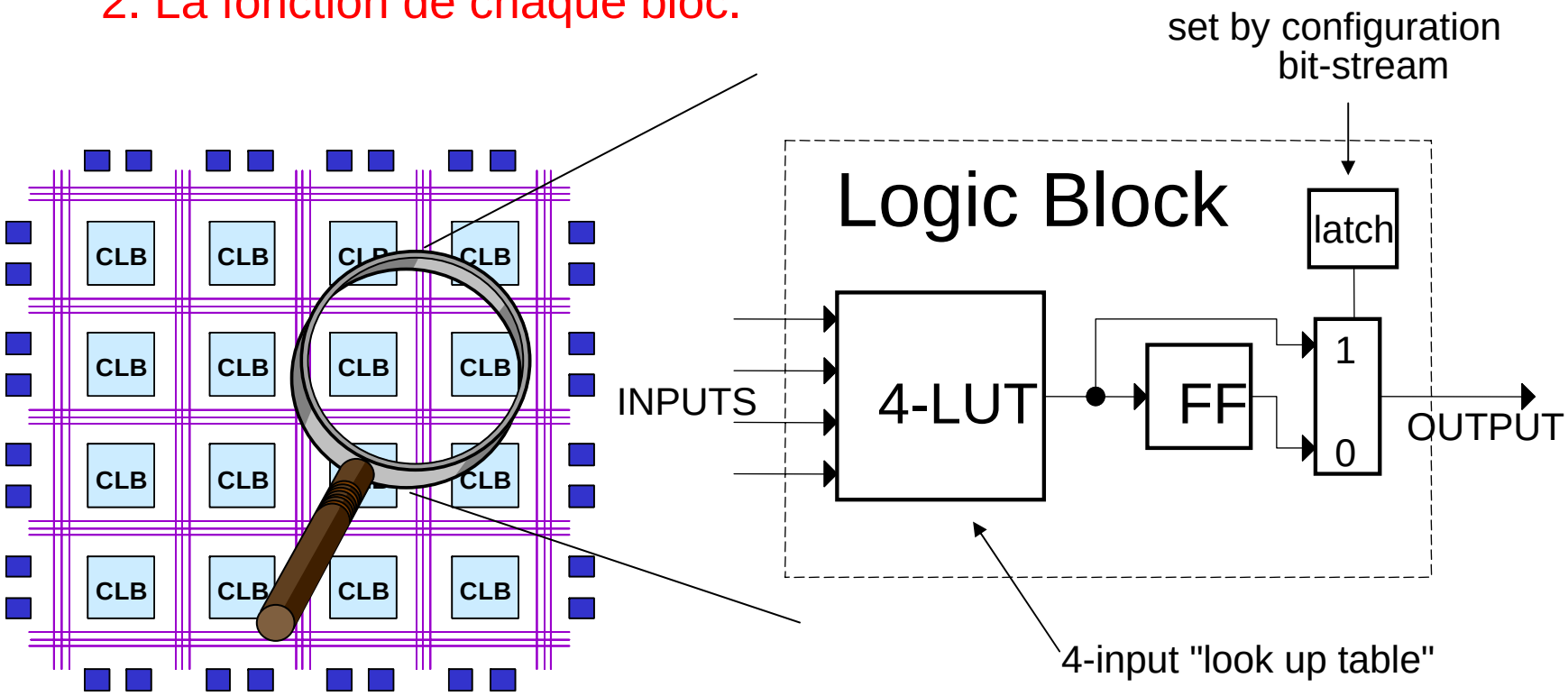
FPGAs : Reconfigurable

- Arrangement Matriciel de blocs logiques avec configuration des :
  1. Interconnexions entre les blocs logiques,
  2. La fonction de chaque bloc.

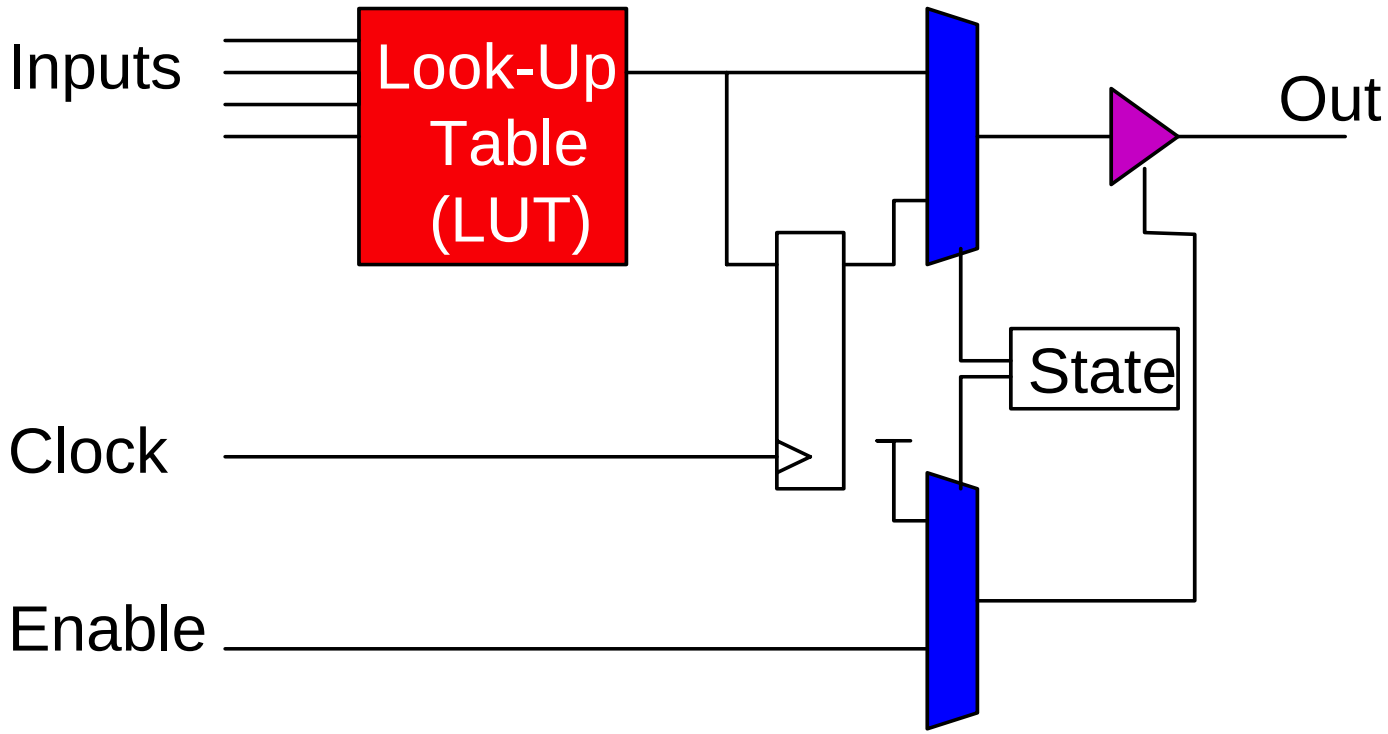


FPGAs : Reconfigurable

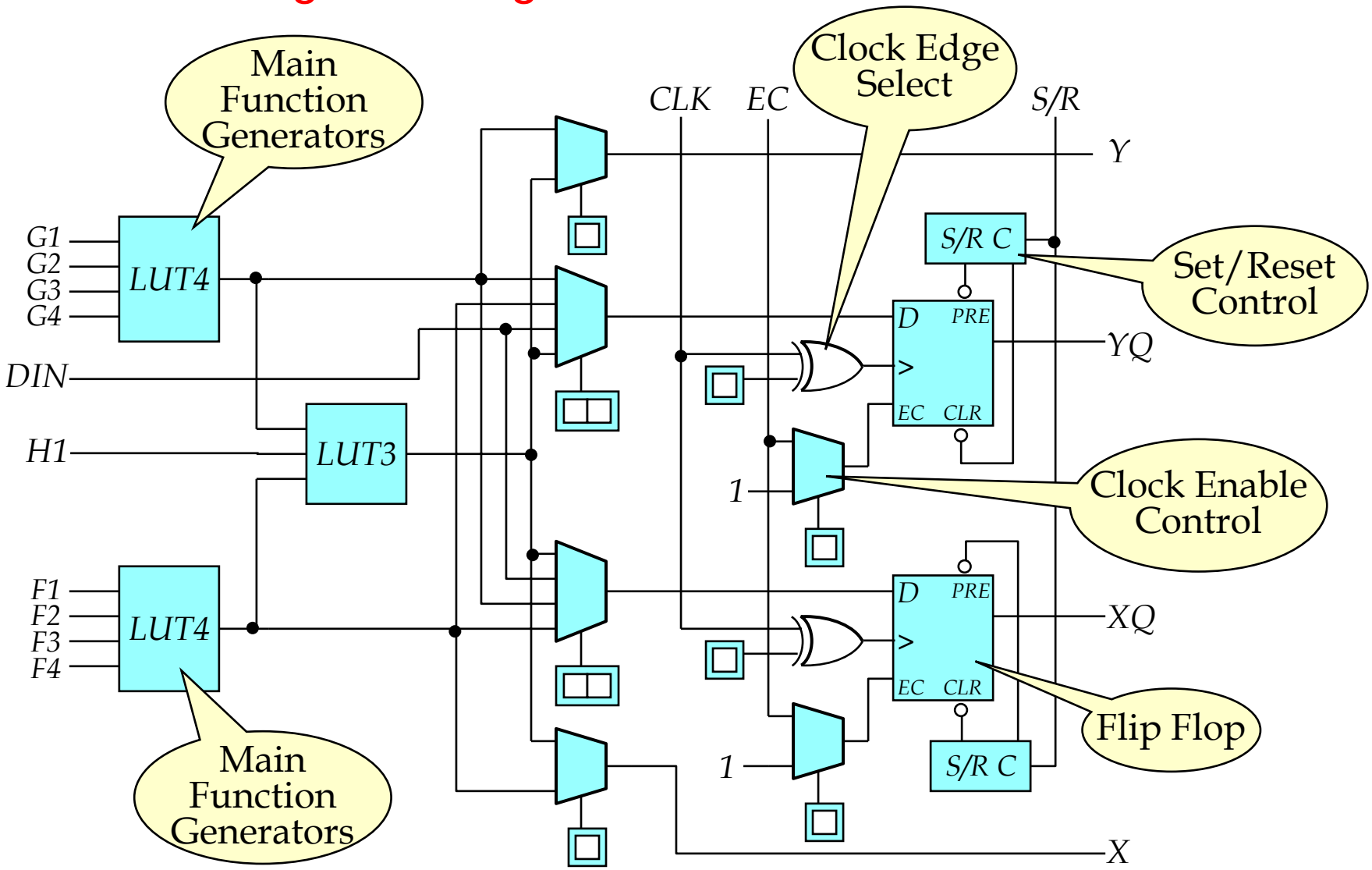
- Arrangement Matriciel de blocs logiques avec configuration des :
  1. Interconnexions entre les blocs logiques,
  2. La fonction de chaque bloc.



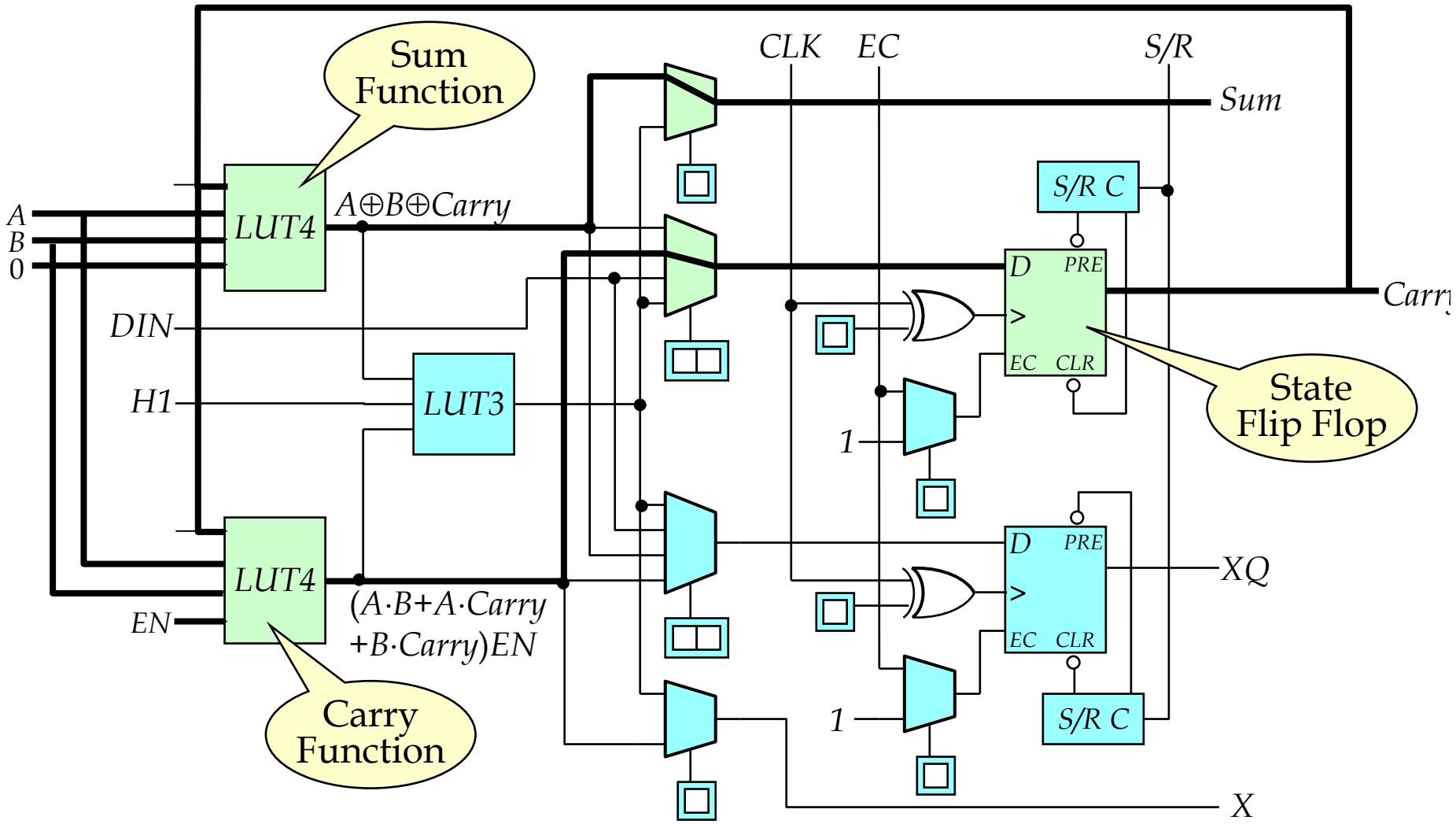
FPGAs : Generic reconfigurable block



FPGAs : Configurable logic block



FPGAs : Serial adder implementation example



FPGAs : Xilinx CLB (Configurable Logic Block)

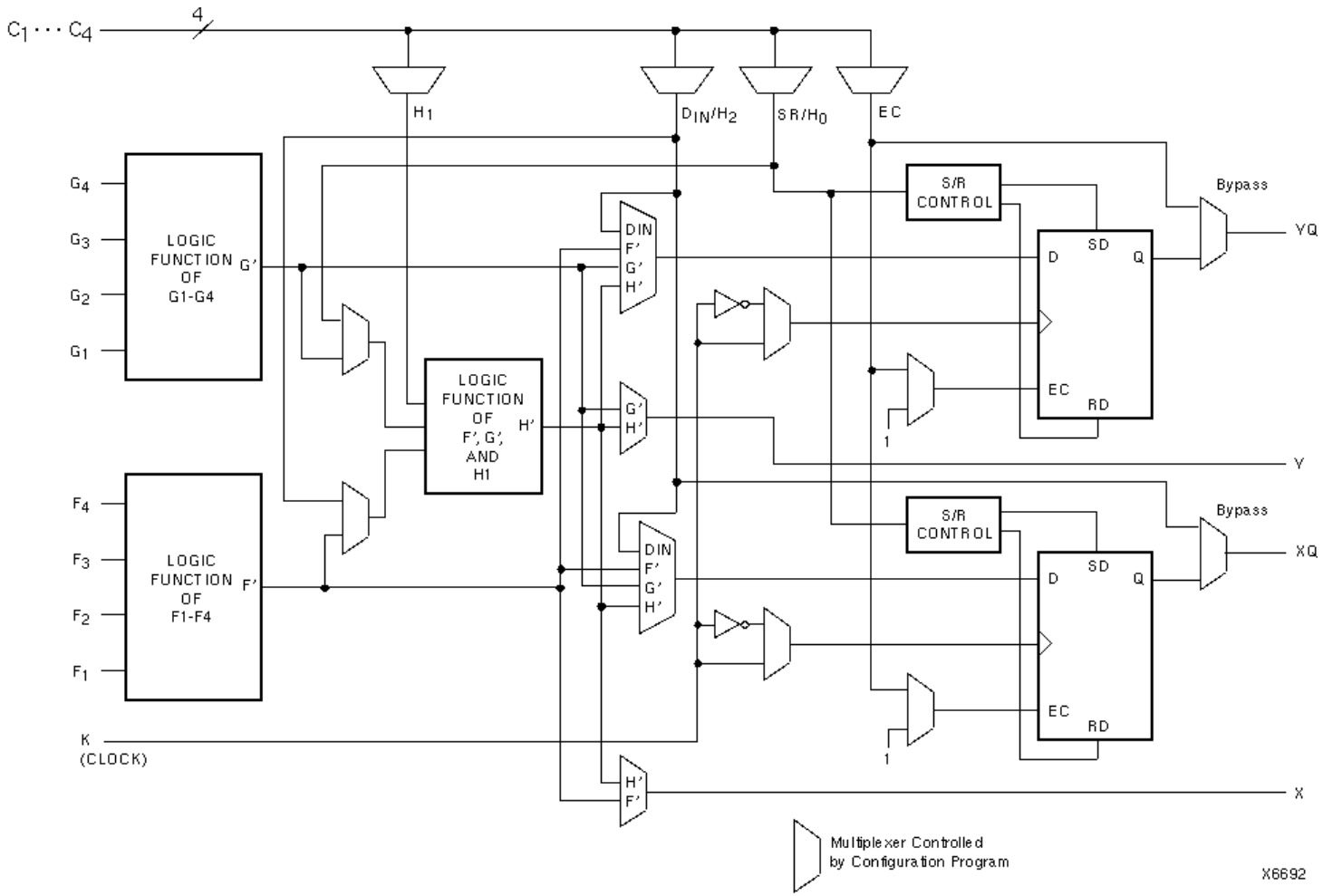
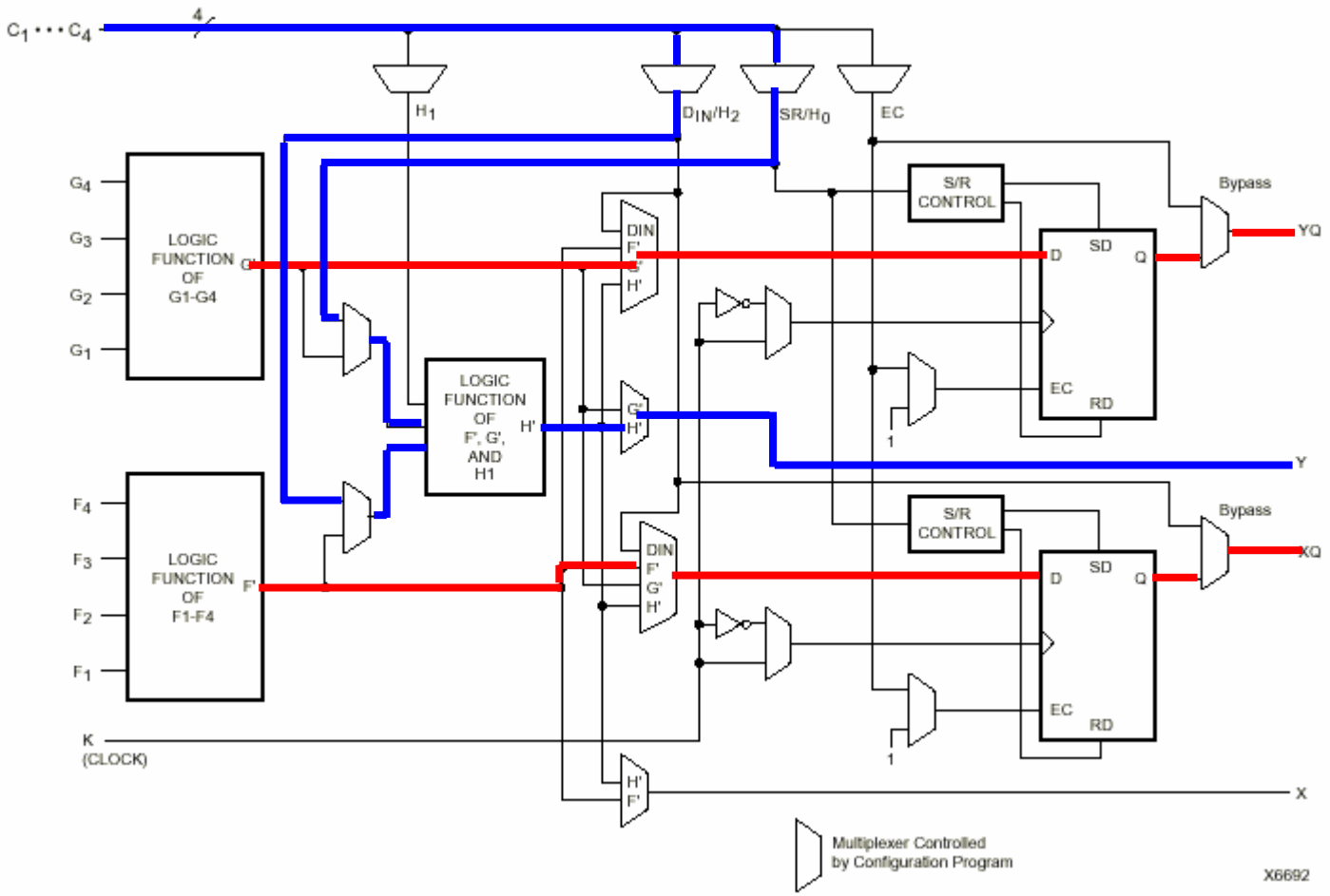


Figure 1: Simplified Block Diagram of XC4000 Series CLB (RAM and Carry Logic functions not shown)

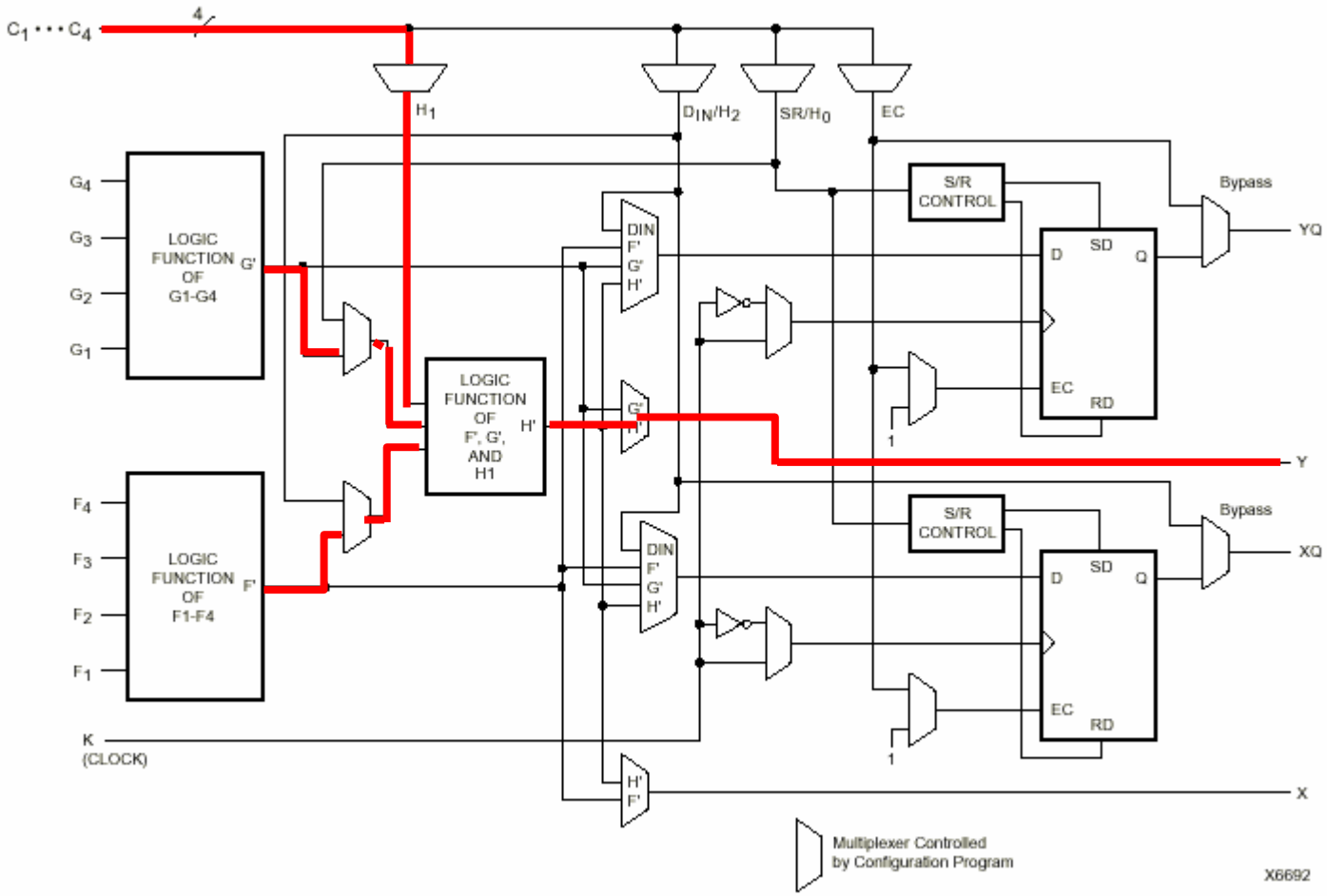
FPGAs : Xilinx CLB (Configurable Logic Block)



Two 4-input functions, registered output

3- ARCHITECTURES ET CIRCUITS

FPGAs : Xilinx CLB (Configurable Logic Block)



5-input function, combinational output

FPGAs : Xilinx CLB used as single port RAM

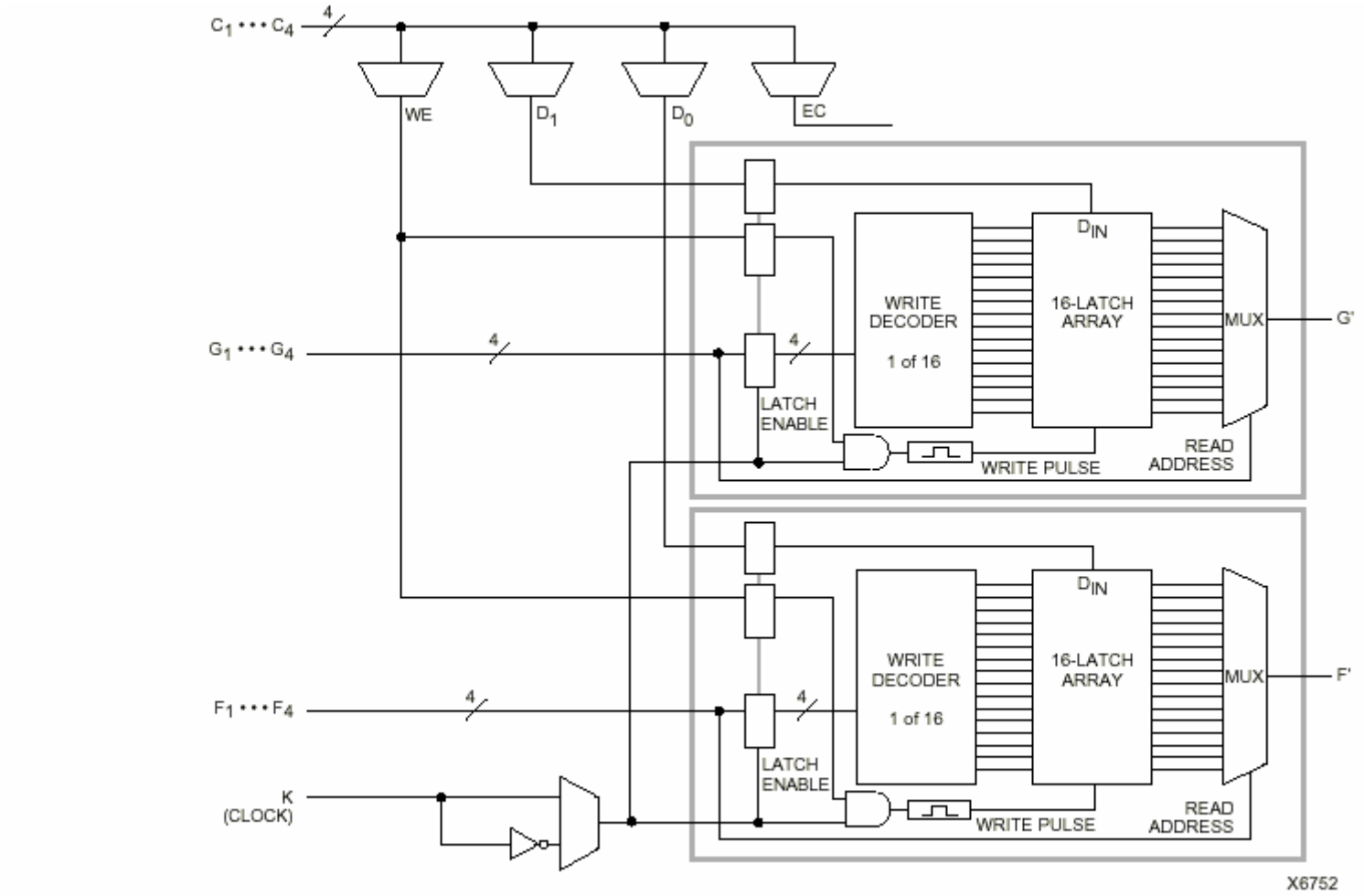
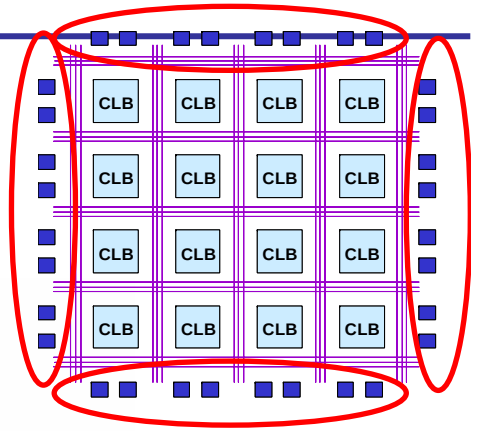
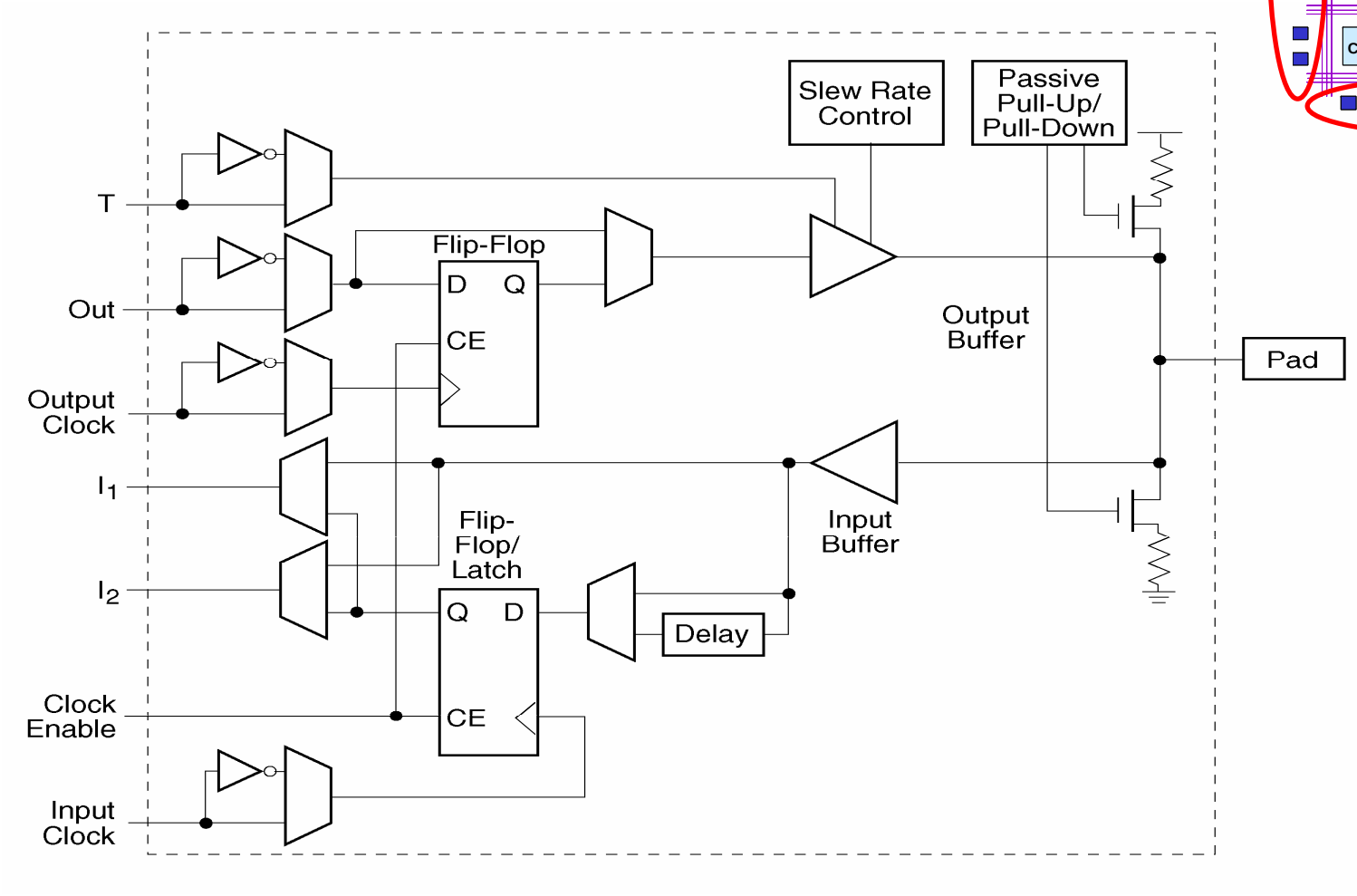


Figure 4: 16x2 (or 16x1) Edge-Triggered Single-Port RAM

3- ARCHITECTURES ET CIRCUITS

FPGAs : Xilinx IOB (Input Output Block)

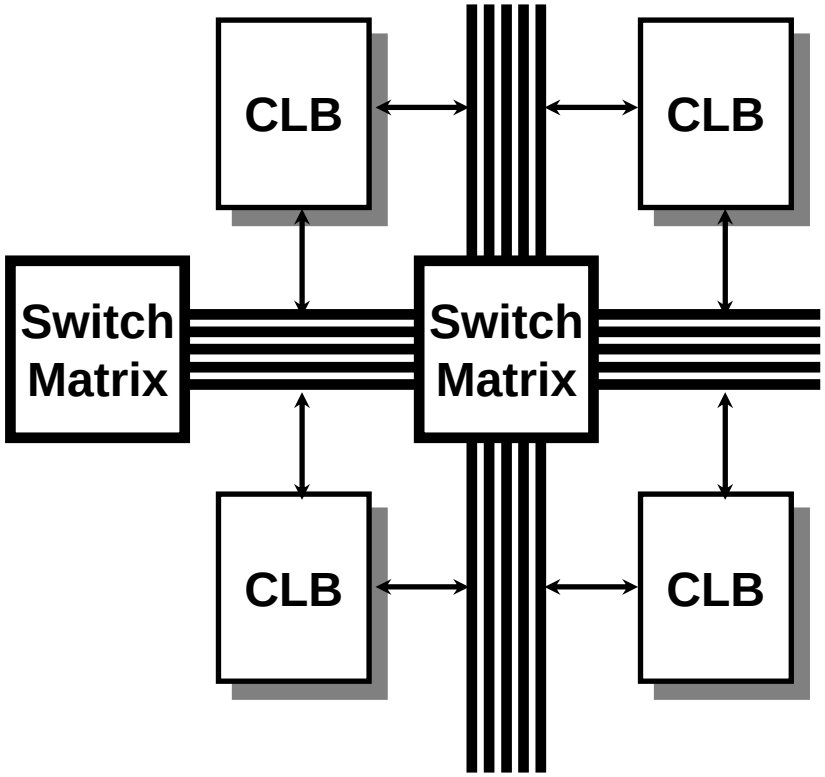




FPGAs : Xilinx interconnect

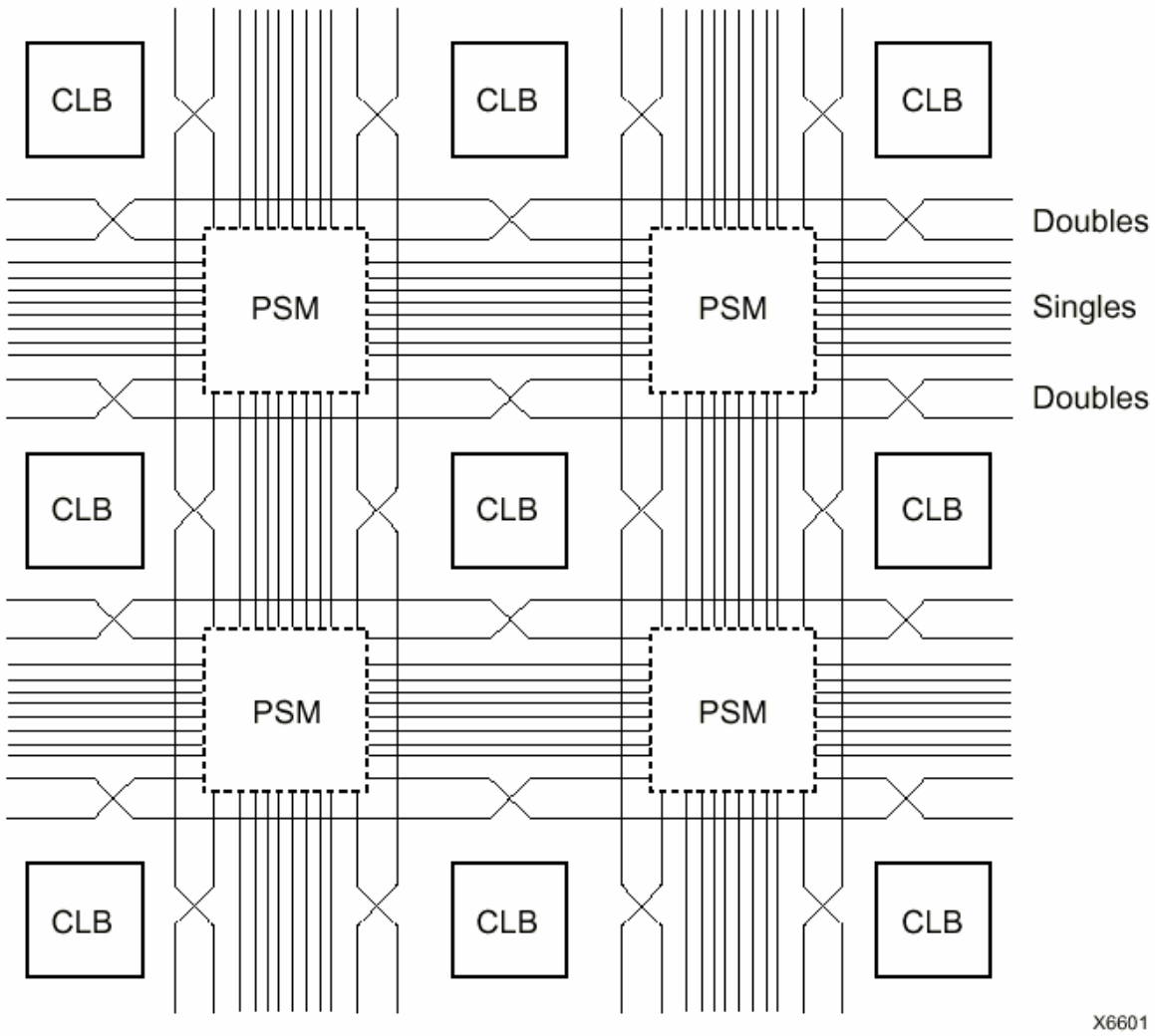
- Interconnexion directe - CLB to CLB
- Utilisation de la switch matrix

- Lignes longues
  - Horloge (skew)
  - Lignes segmentées
  - Bus (Tri state)

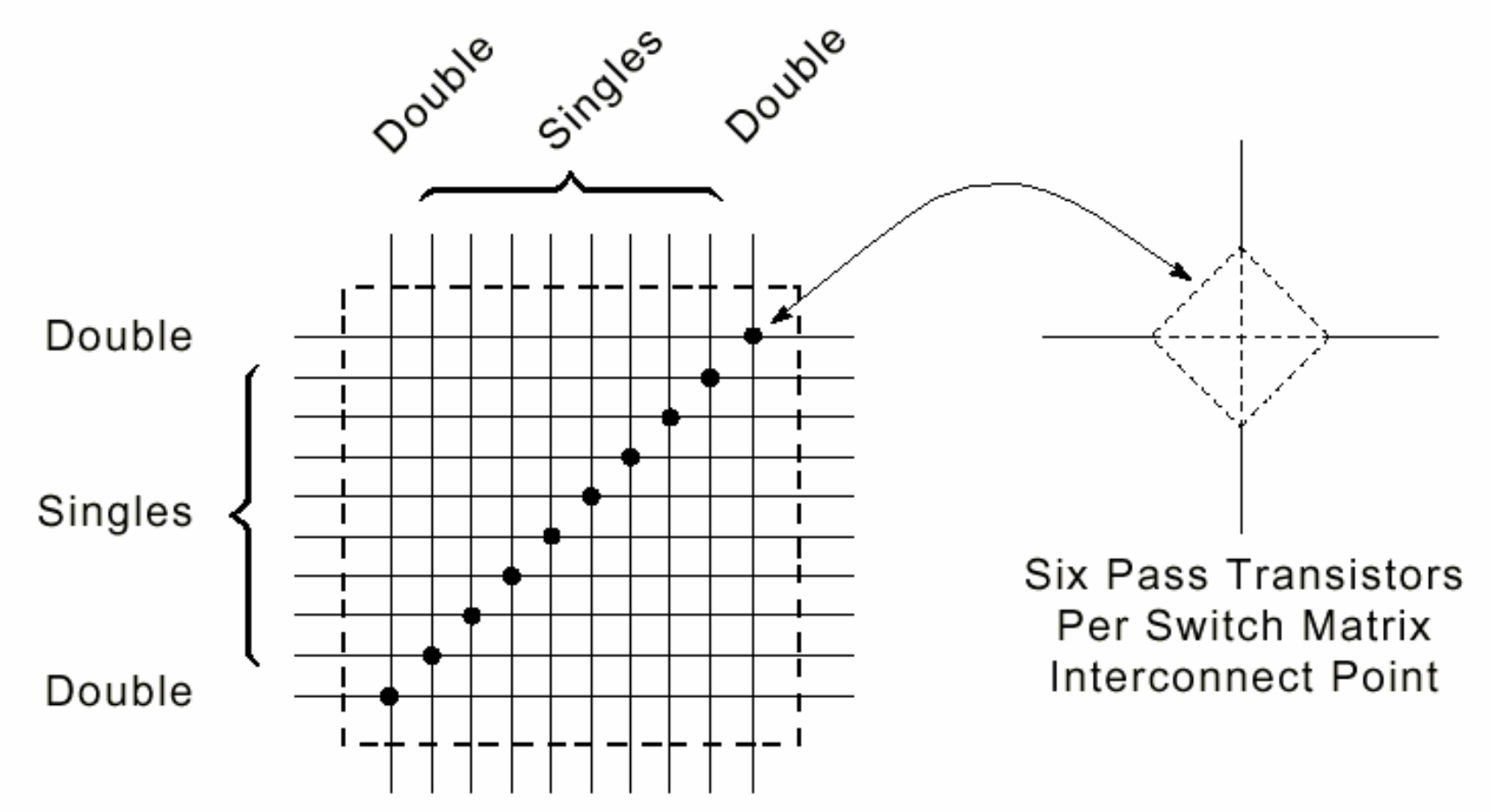


3- ARCHITECTURES ET CIRCUITS

FPGAs : Xilinx interconnect

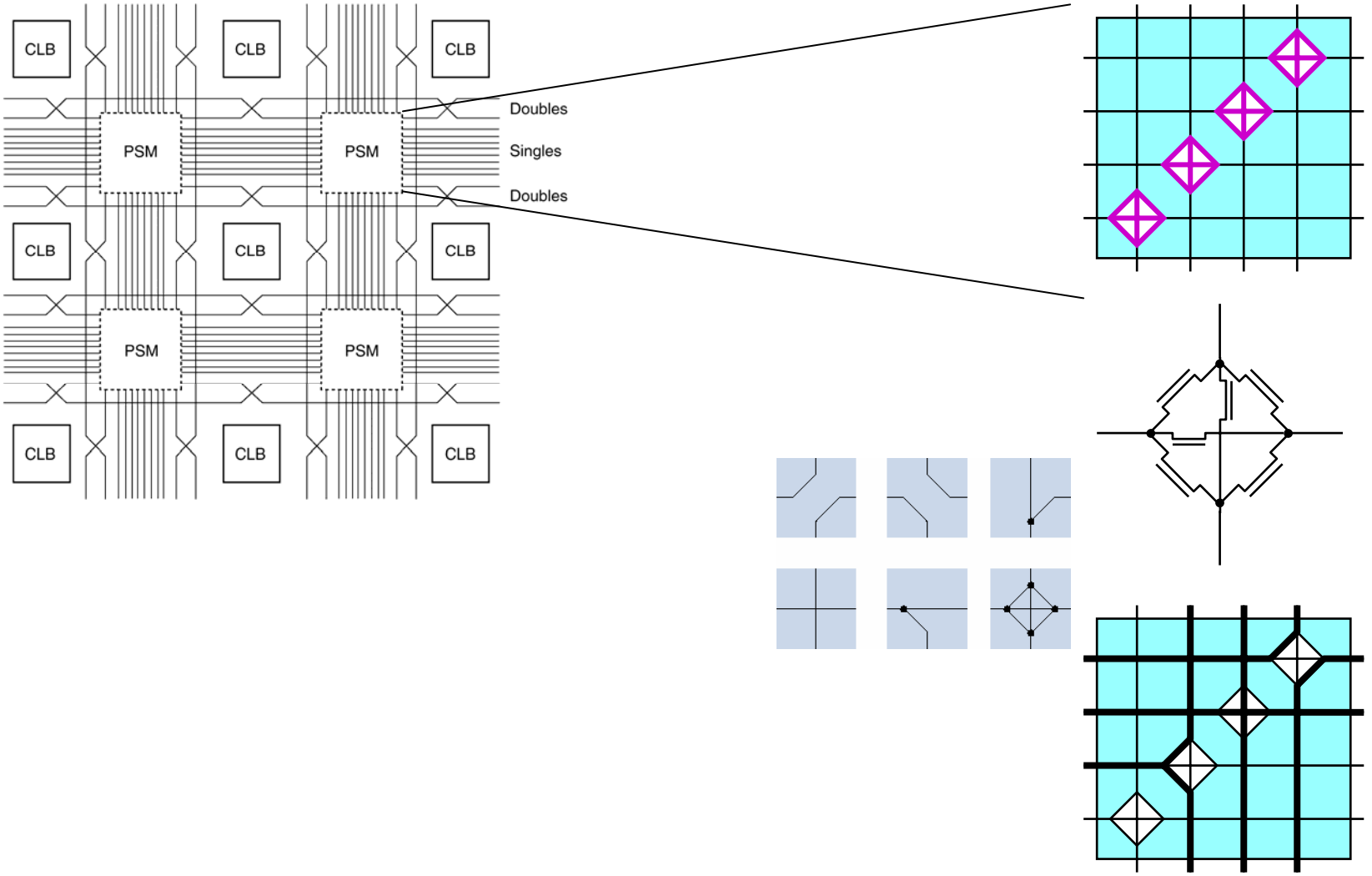


FPGAs : Xilinx Switch matrix

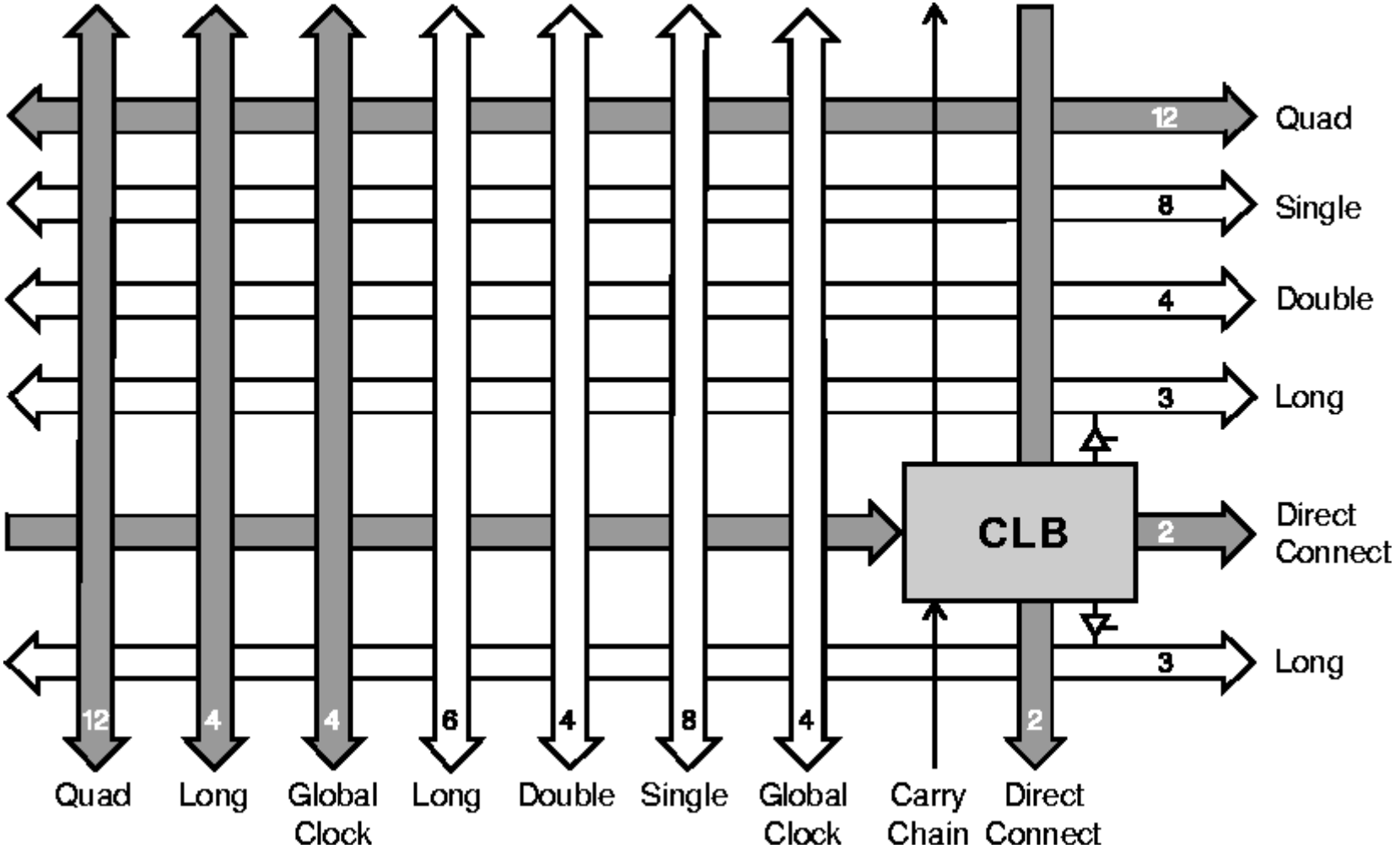


3- ARCHITECTURES ET CIRCUITS

FPGAs : Switch matrix architecture

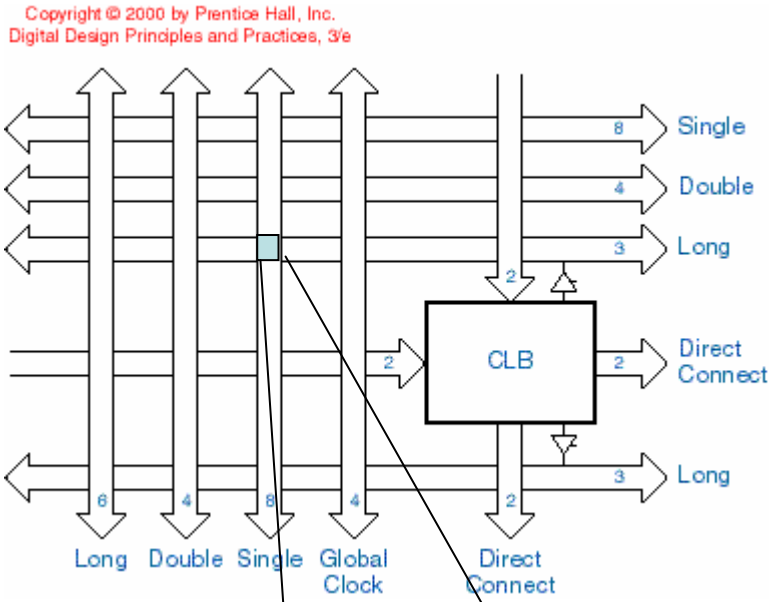


FPGAs : Xilinx Interconnect



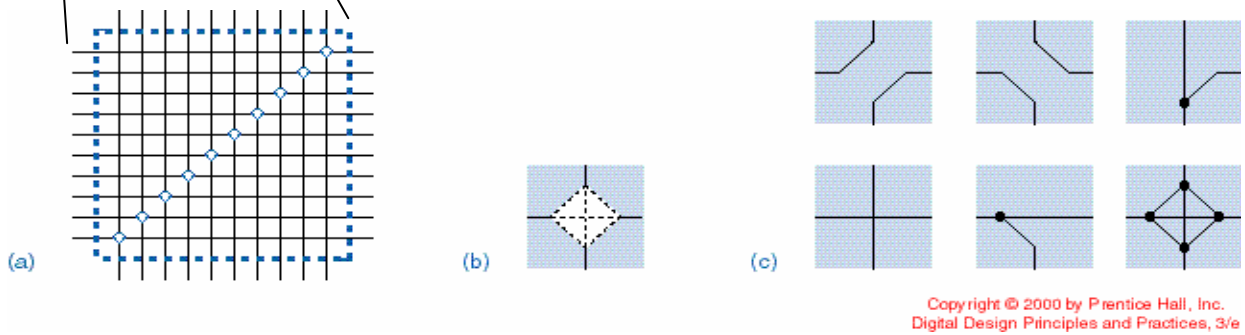
3- ARCHITECTURES ET CIRCUITS

FPGAs : Xilinx Interconnect



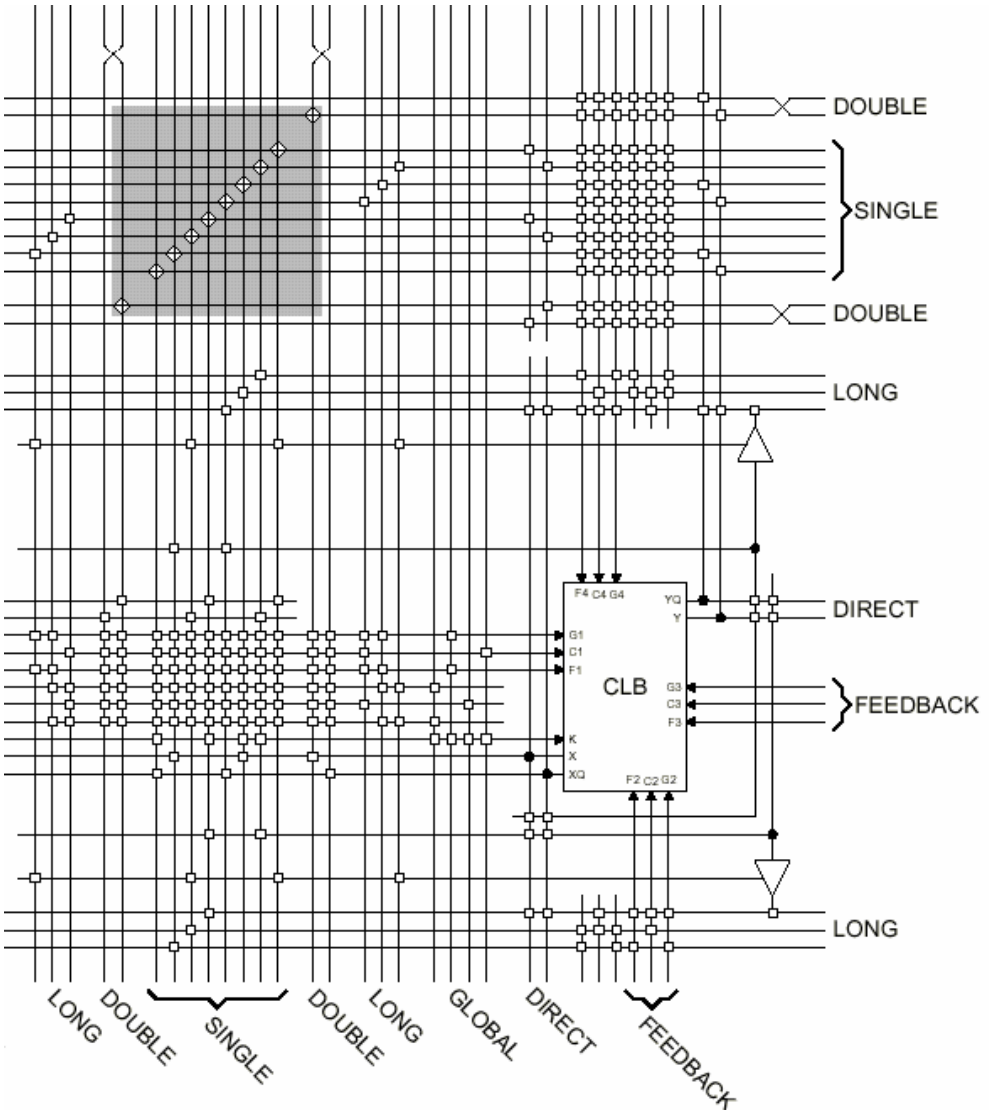
- **Single**
  Entre 2 CLBs voisins
- **Doubles**
  Entre CLB éloignés
- **Long**
  Traversée complète
- **Global clock**
  Optimisée pour limitée le skew

Programmable switches



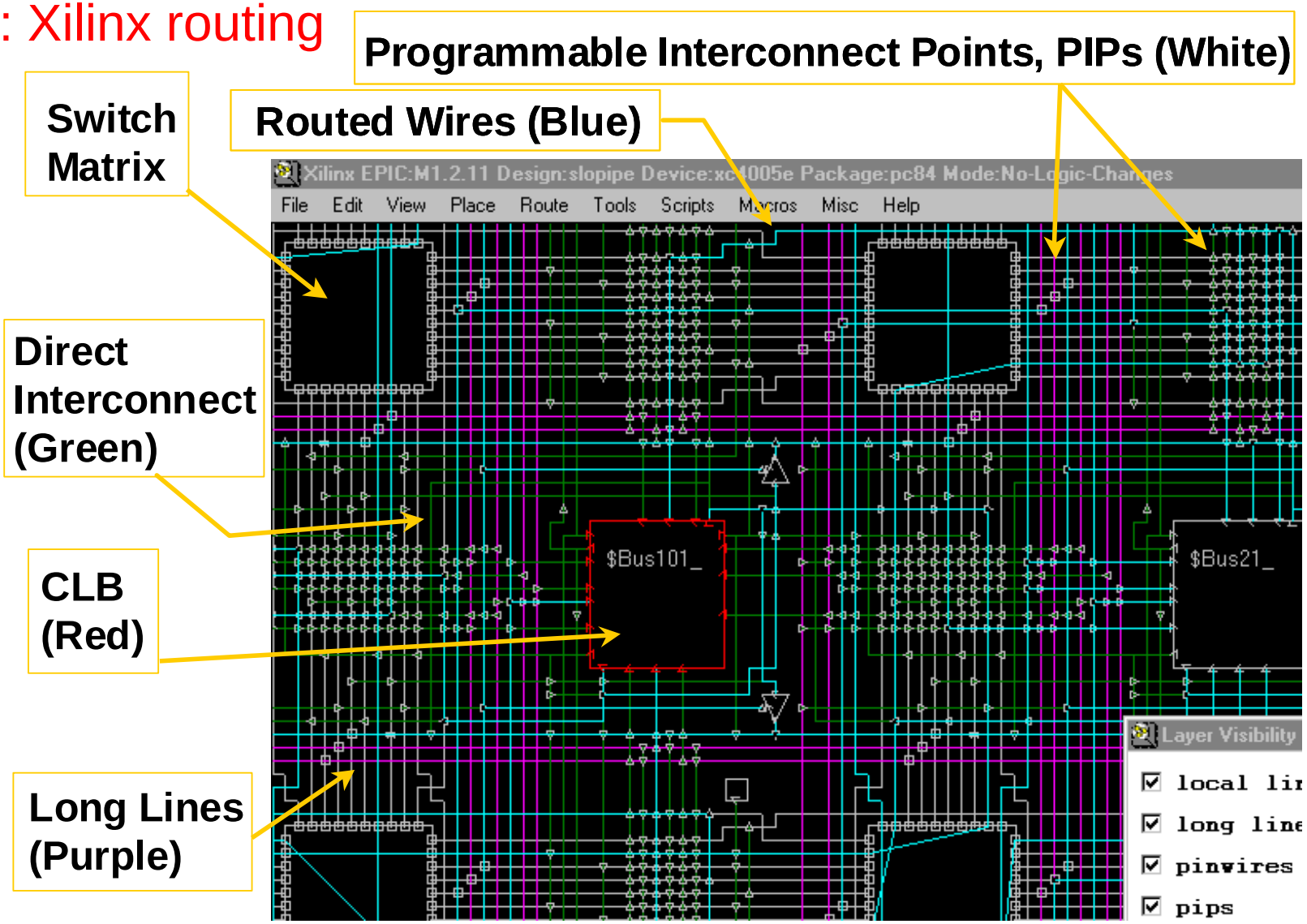
3- ARCHITECTURES ET CIRCUITS

FPGAs : Xilinx Interconnect



3- ARCHITECTURES ET CIRCUITS

FPGAs : Xilinx routing



3- ARCHITECTURES ET CIRCUITS

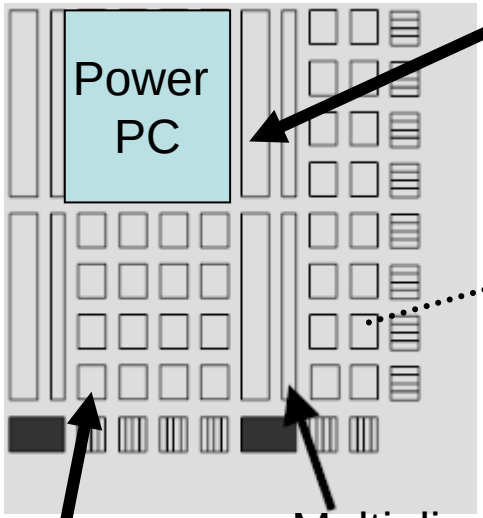
FPGAs : Xilinx Virtex-II et IV architecture

- Virtex 4, 17 Familles
- www.xilinx.com/ipcenter
- > 500 Mhz de fréquence d'horloge
- 90 nm, 9-Metal layers
- Fonctions arithmétiques cablées (Multiplieur, MAC, etc …)
- 2 à 4 Processeurs type Power PC
- **50 Millions de bits de configuration !!**

Devices	*Array*	*CLBs*	*Slices*	*Distributed RAM (Kb)*	*DSP*	*Max Block RAM*
XC4VLX15	64 x 24	13,824	6,144	96	32	864
XC4VLX25	96 x 28	24,192	10,752	168	48	1,296
XC4VLX40	128 x 36	41,472	18,432	288	64	1,728
XC4VLX60	128 x 52	59,904	26,624	416	64	2,880
XC4VLX80	160 x 56	80,640	35,840	560	80	3,600
XC4VLX100	192 x 64	110,592	49,152	768	96	4,320
XC4VLX160	192 x 88	152,064	67,584	1056	96	5,184
XC4VLX200	192 x 116	200,448	89,088	1392	96	6,048

3- ARCHITECTURES ET CIRCUITS

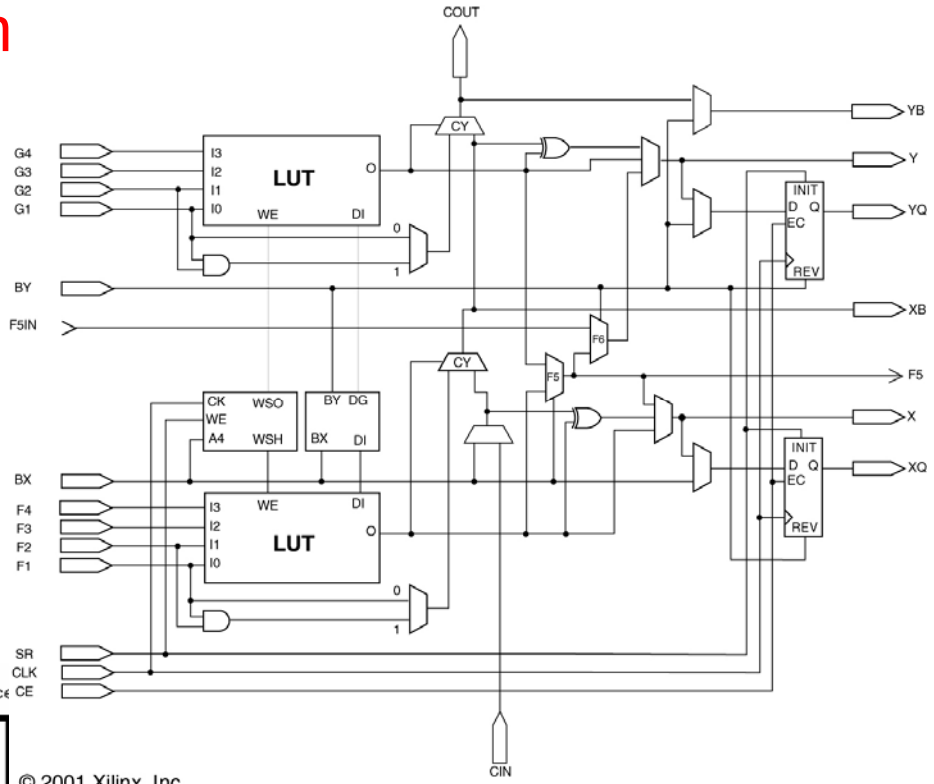
FPGAs : Xilinx Virtex-II et IV arch

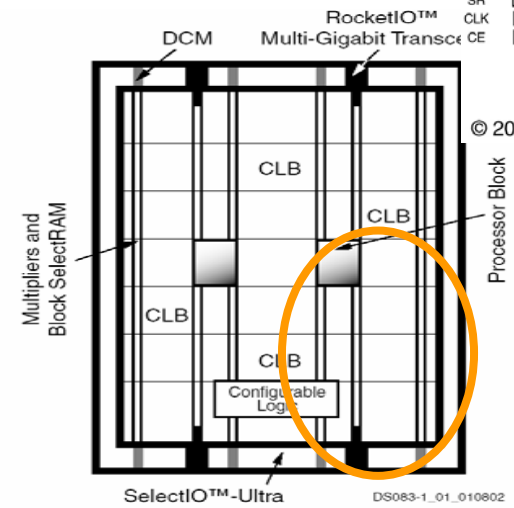


Mémoire

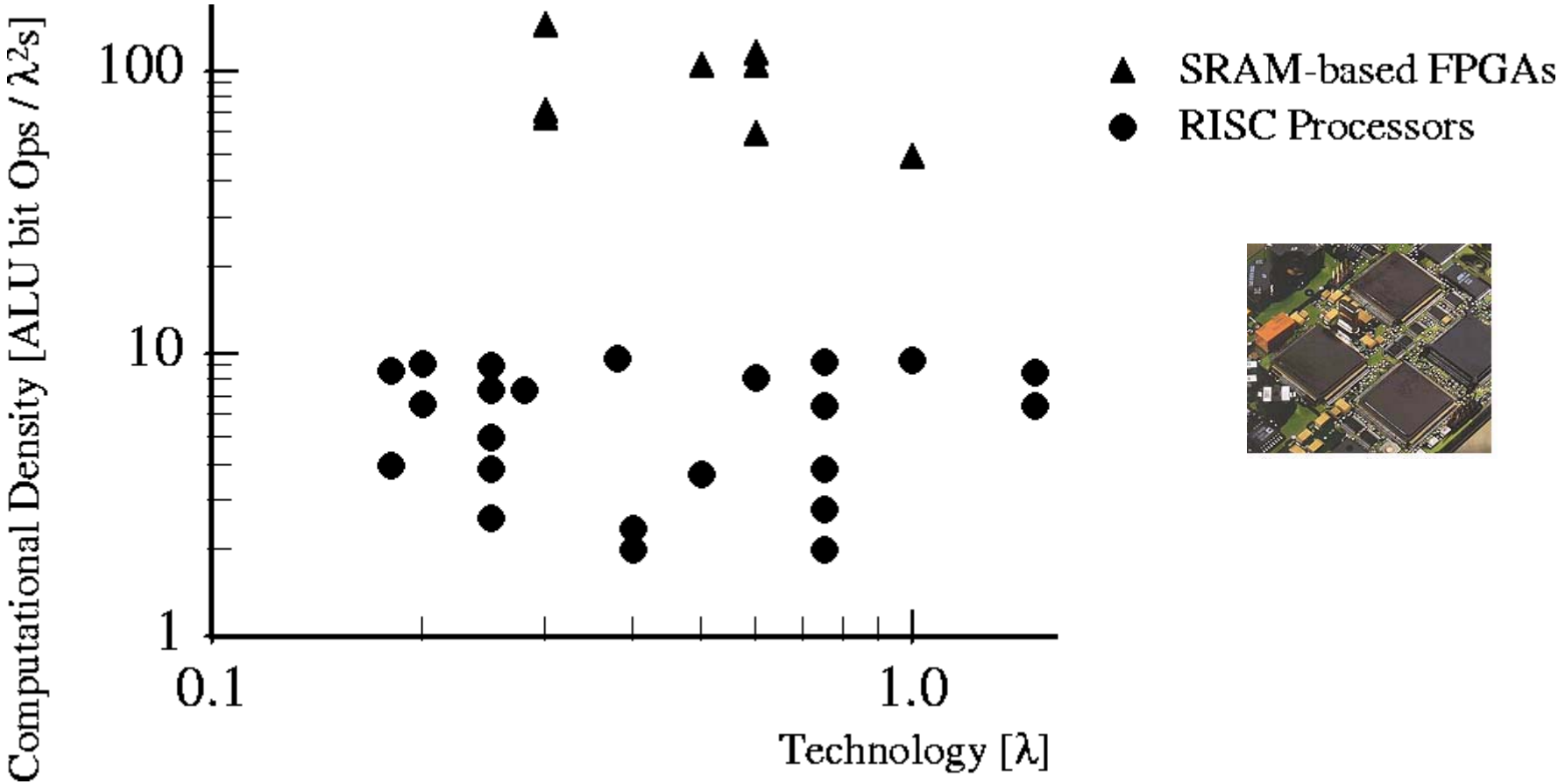
Multiplieurs 18x18

Matrice
de Slices



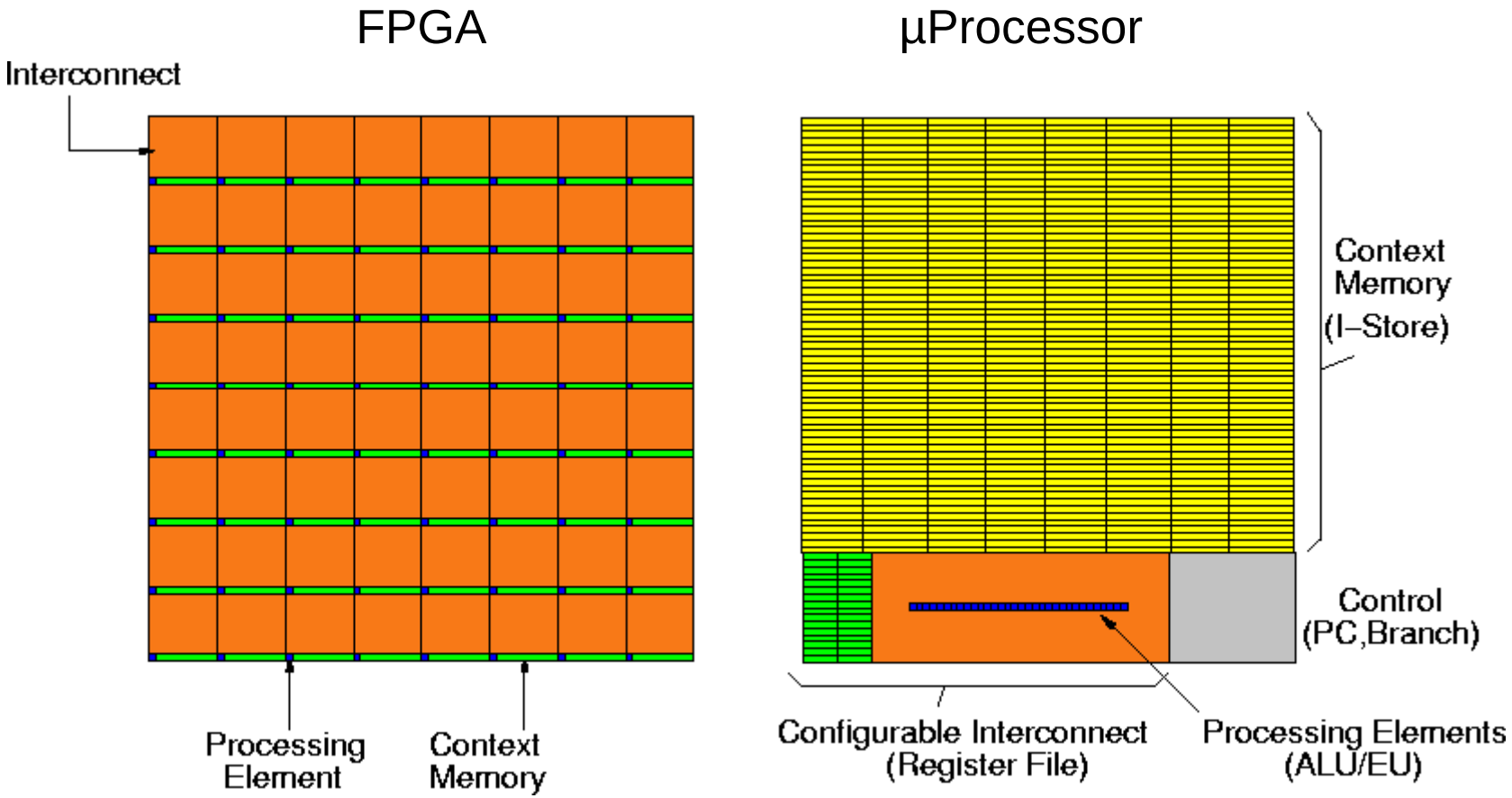


FPGAs : Comparaison en densité (VS RISC)



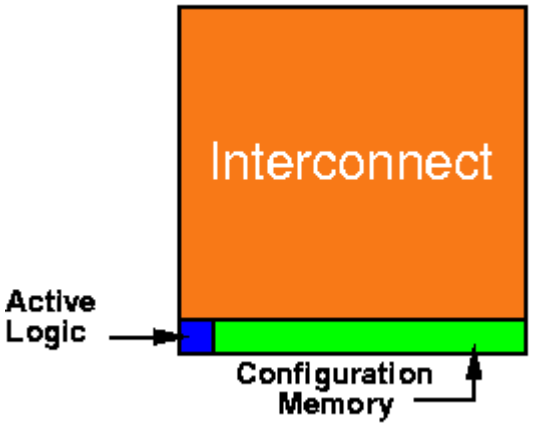
3- ARCHITECTURES ET CIRCUITS

FPGAs : Comparaison en Surface (vs. processeurs)



FPGAs : Comparaison en surface

Total Area ~ Active logic + Configuration memory + interconnect

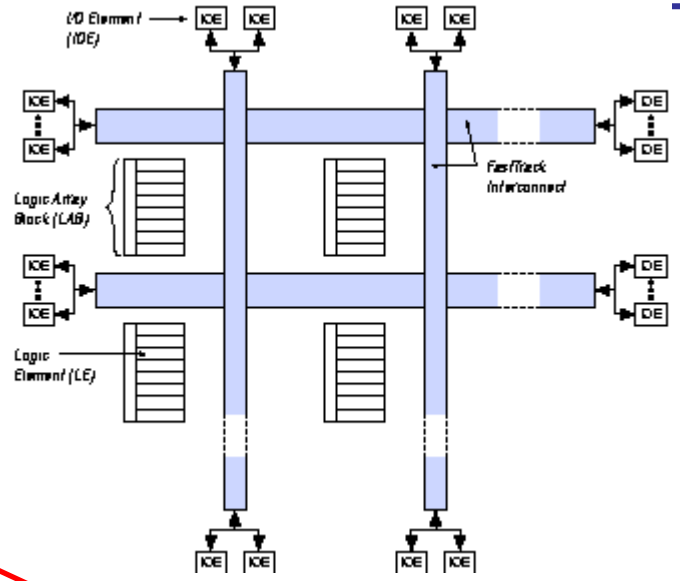


Coût de la technologie reconfigurable

Function	Area (λ^2)	
LUT MUX + ff	20K	(generous, closer to 10K)
Programming Memory	80K	(240K typical unencoded)
Interconnect	700K	(for $N_p = 2048$)

3- ARCHITECTURES ET CIRCUITS

FPGAs : Retard

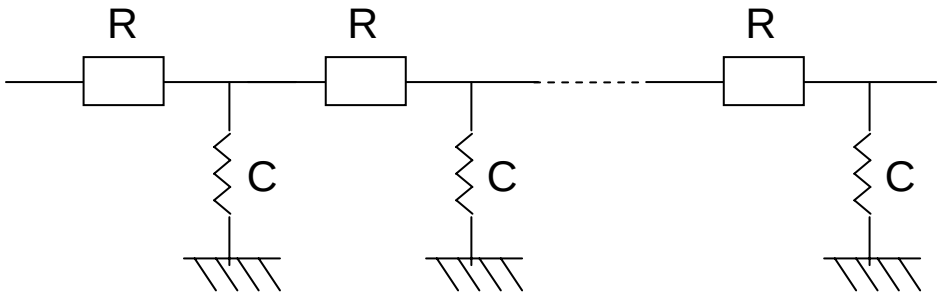


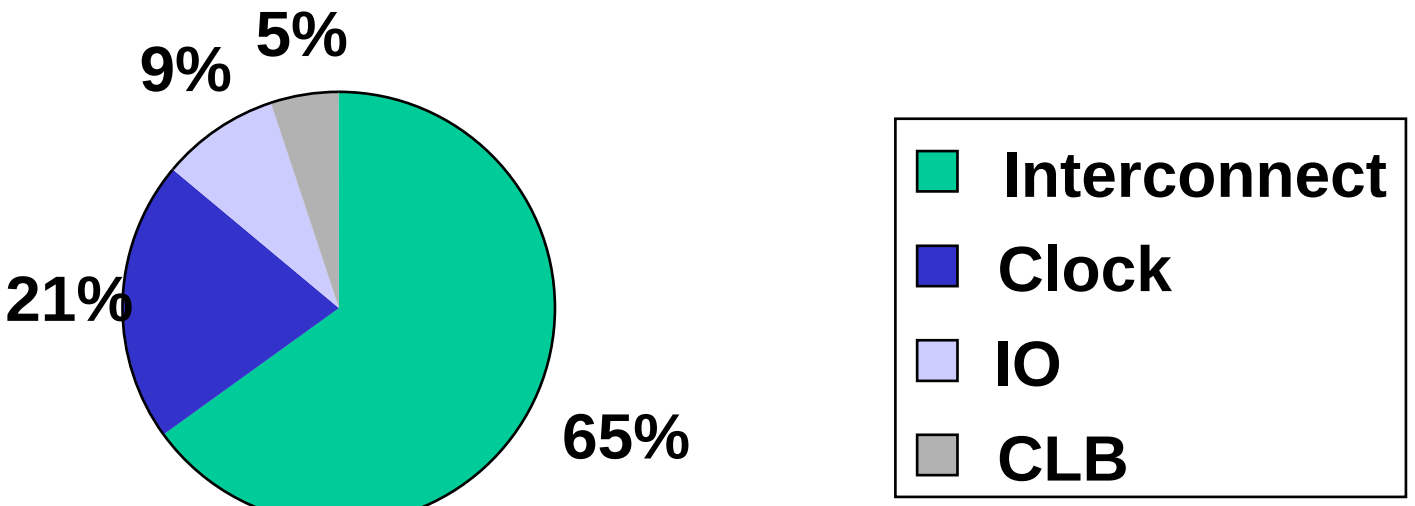
Interconnexion = 90% du retard

Design	Path	Total Delay	LUT Delay	Inter. %
Altera 10K130V-2	LUT-local-LUT	2.5 ns	2.1 ns	16%
	LUT-row-local-LUT	6.6 ns	2.1 ns	68%
	LUT-column-local-LUT	11.1 ns	2.1 ns	81%
	LUT-row-column-local-LUT	15.6 ns	2.1 ns	87%
	LUT-row-fanout-local-LUT (fanout)	28 ns	2.1 ns	90%

FPGAs : Consommation

Modélisation simple





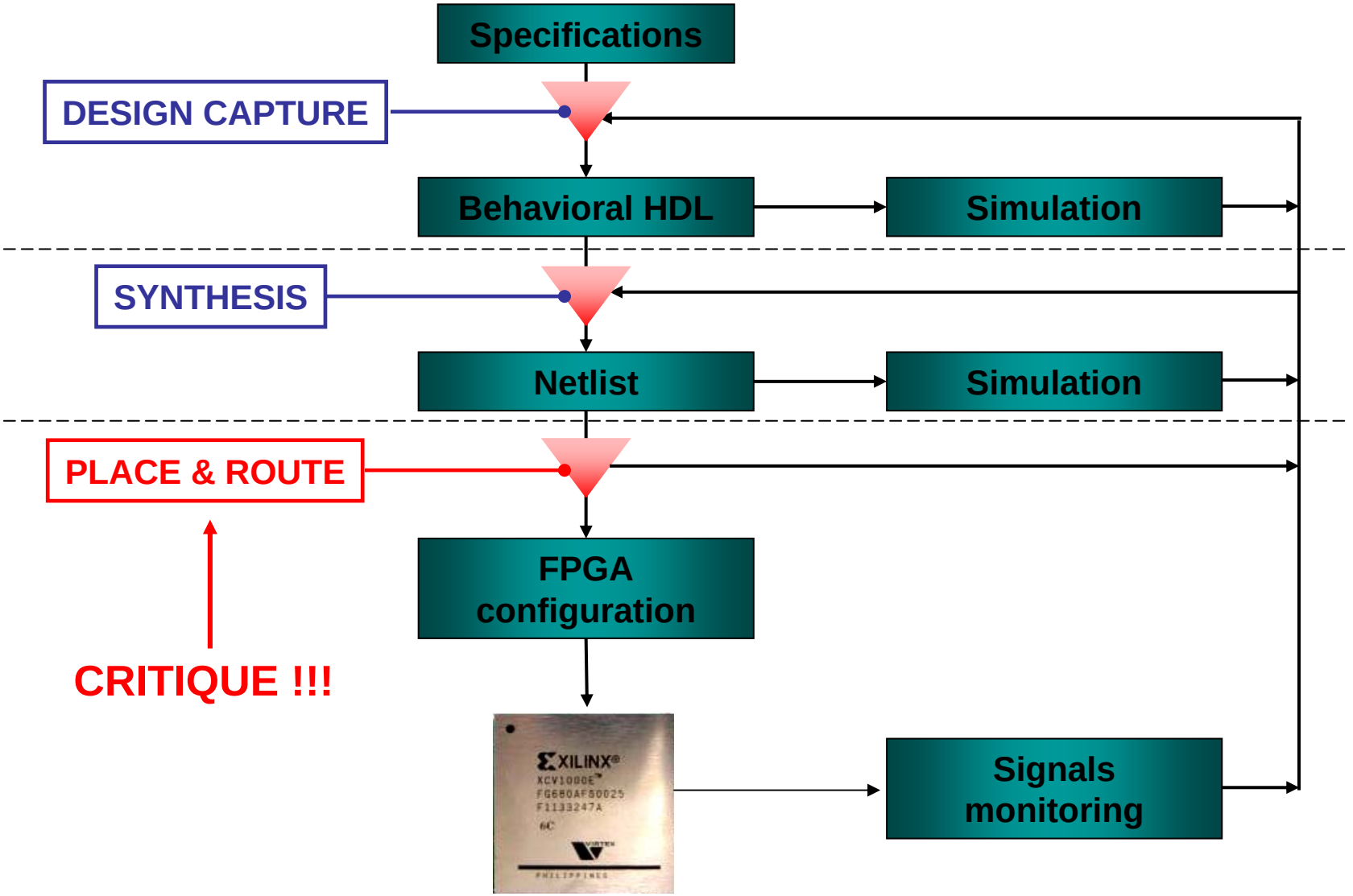
Sommaire

Partie 1 : Circuits reconfigurables « standards »

1- PRINCIPES et CLASSIFICATION
2- TECHNOLOGIES
3- ARCHITECTURES ET CIRCUITS
4- CONCEPTION

4- CONCEPTION

FPGAs design flow



4- CONCEPTION

Role of design tools

- Synthesis : Assignation HDL sur FPGA
- Place & Route : Assignation de la netlist sur les CLBs et routage des signaux
- Simulation :
  - Avant synthèse : simulation fonctionnelle
  - Après synthesis : Simulation de de la netlist, prise en compte des retards
  - Après Place & Route : idem, mais retard du routage pris en compte

accuracy ↓

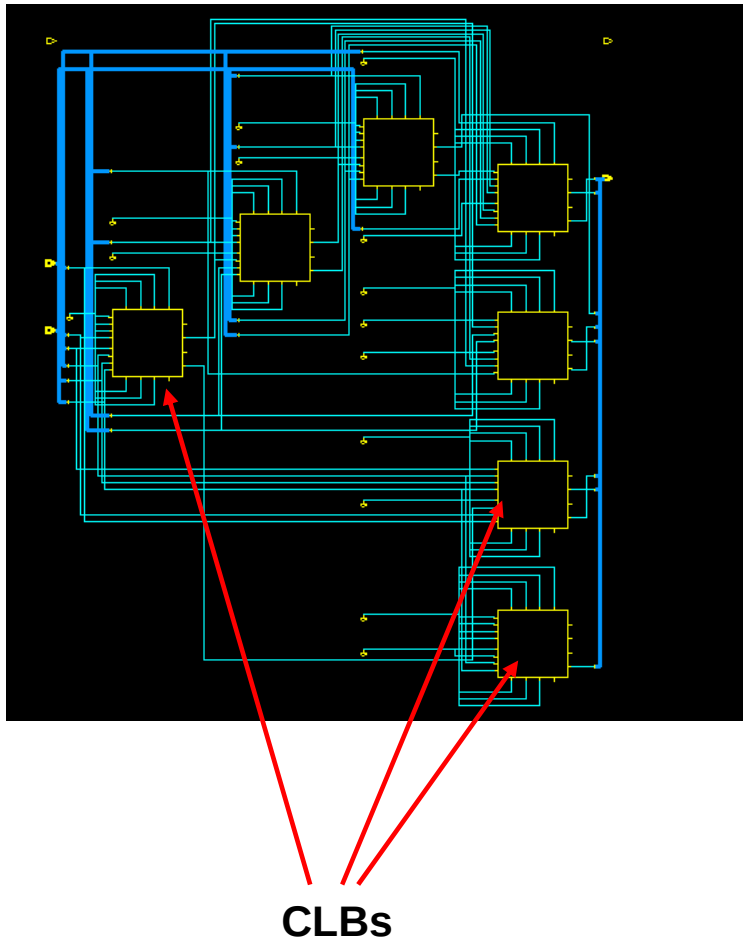
ETAPES GOURMANDES EN TEMPS CPU

4- CONCEPTION

VHDL description

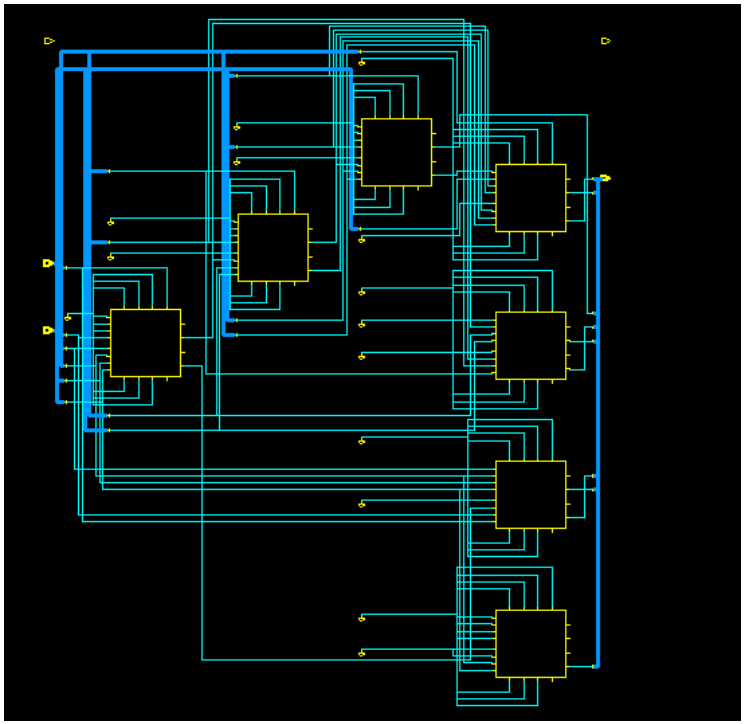
```
library IEEE;
 use IEEE.std_logic_1164.all;
 use IEEE.std_logic_arith.all;
 use IEEE.std_logic_signed.all;

entity add4 is
       port(   A, B    : in  std_logic_vector(7 downto 0);
               SUM     : out std_logic_vector(7 downto 0)
          );
end add4 ;

architecture archi_add4 of add4 is
begin
process(A, B)
begin
 sum <= A+B ;
end process;
end archi_add4;
```

Synthesized description (synopsys)



4- CONCEPTION

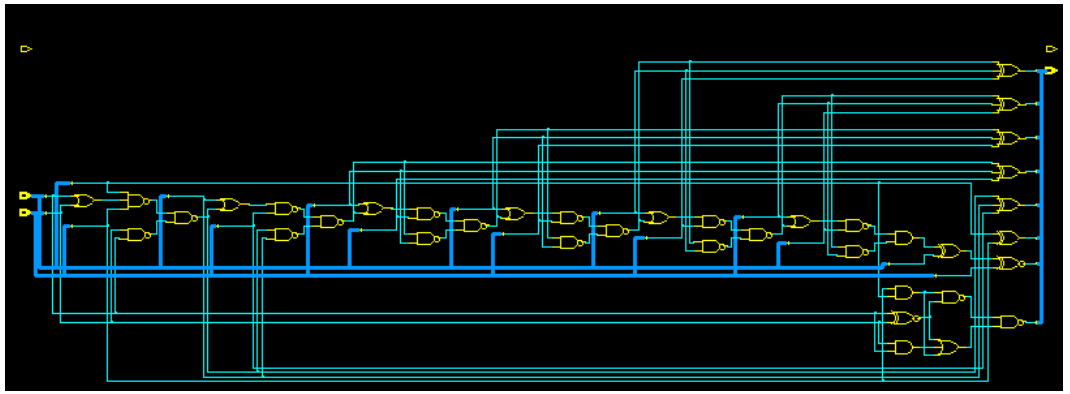
Synthesized description (synopsys)



```
module add4_DW01_add_8_0 ( A, B, CI, SUM, CO );
input  [7:0] A;
input  [7:0] B;
output [7:0] SUM;
input  CI;
output CO;
   wire n8, n9, n11, n12, n14;
   clb_4000 U4 ( .K(1'b0), .GSR(1'b0), .CIN(1'b0), .F1(B[0]), .F2(A[0]), .F3(
      1'b0), .F4(1'b0), .G1(1'b0), .G2(1'b0), .G3(1'b0), .G4(1'b0), .C1(1'b0
      ), .C2(1'b0), .C3(1'b0), .C4(1'b0), .X(SUM[0]) );
   clb_4000 U6 ( .K(1'b0), .GSR(1'b0), .CIN(1'b0), .F1(A[2]), .F2(B[2]), .F3(
      n8), .F4(1'b0), .G1(B[0]), .G2(B[1]), .G3(A[0]), .G4(A[1]), .C1(1'b0),  …
```

Synthesized description

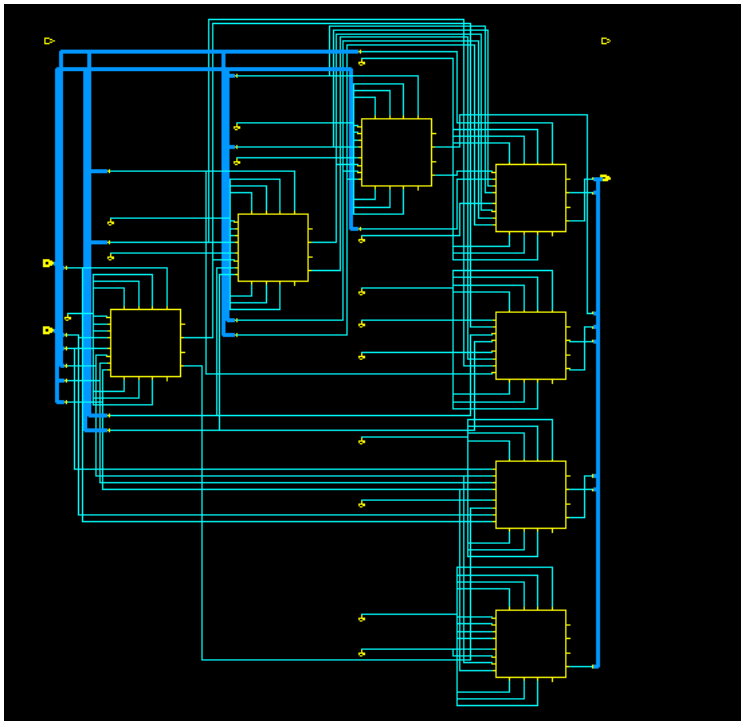
(Gate equivalent)



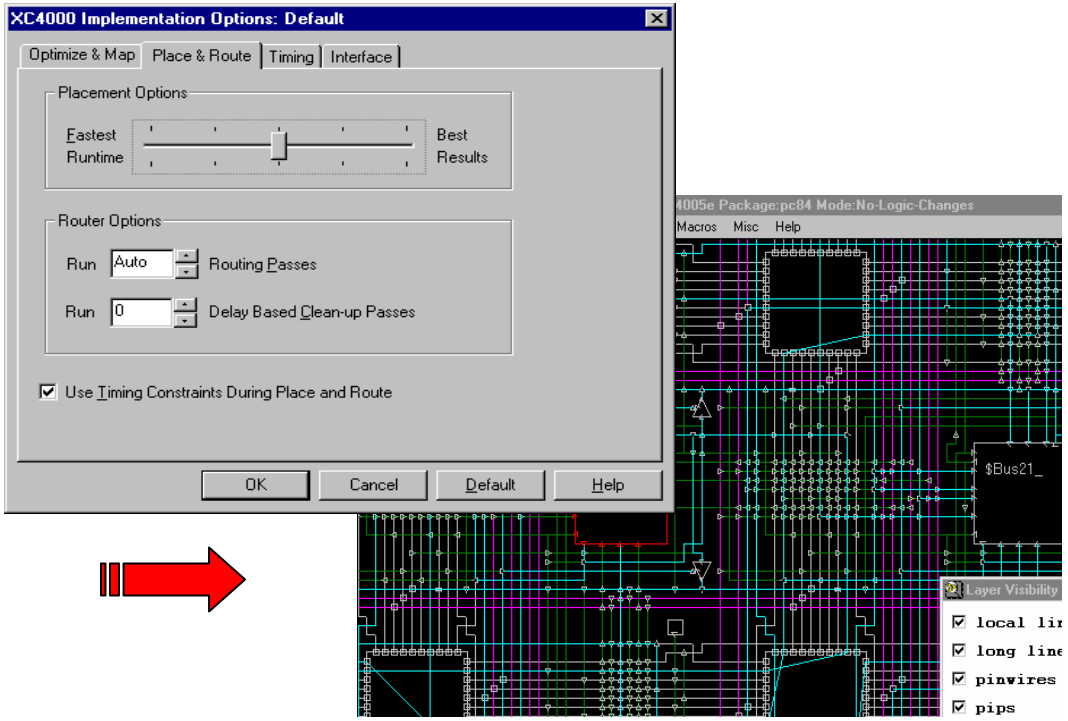
```
module add4_DW01_add_8_0 ( A, B, CI, SUM, CO );
input  [7:0] A;
input  [7:0] B;
output [7:0] SUM;
input  CI;
output CO;
   wire n8, n9, n11, n12, n14, n20, n21, n22, n23, n24, n25, n26, n27, n28,
      n29, n30, n31, n32, n33, n34, n35, n36, n37, n38, n39, n40, n41, n42,
      n43, n44;
   XOR2 U25 ( .O(SUM[0]), .I1(B[0]), .I0(A[0]) );
   OR3 U26 ( .O(n24), .I2(n21), .I1(n20), .I0(n22) );
   …
```

4- CONCEPTION

Synthesized description (synopsys)



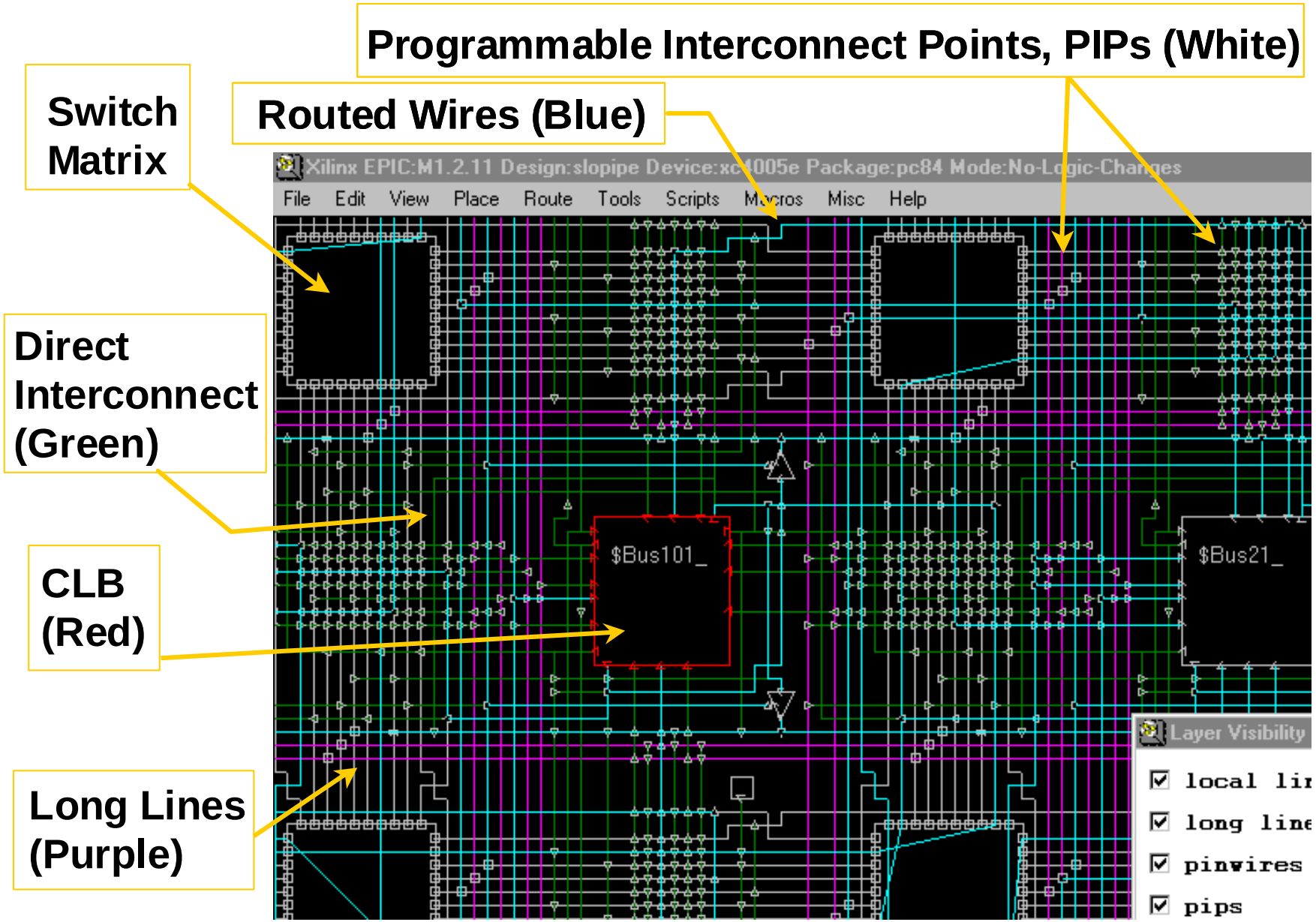
P&R'ed description (synopsys)



CLBs voisins placés aussi proche que possible
Connexions au travers des switch matrix

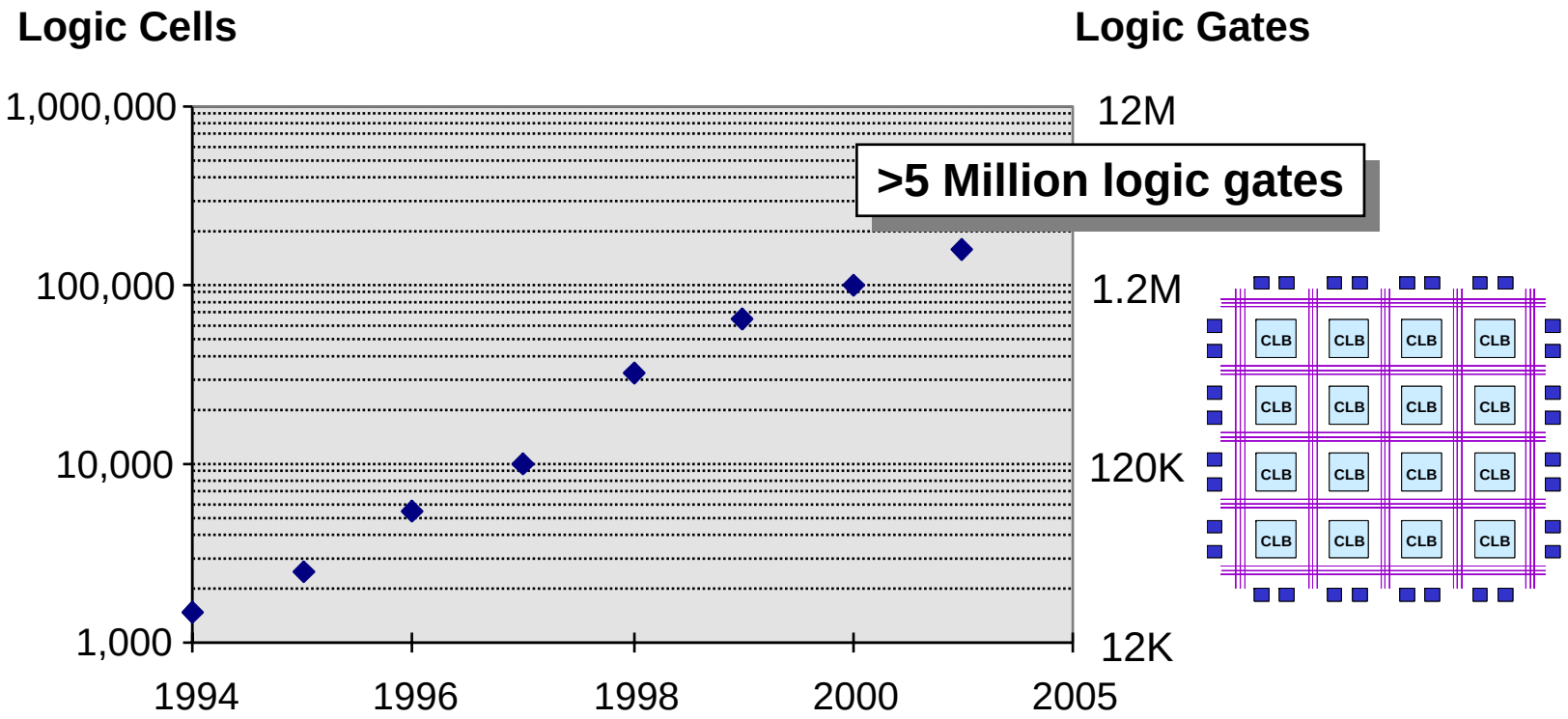
PHASE CRITIQUE

4- CONCEPTION

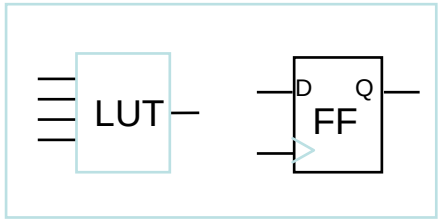


Partie 1 : conclusion

FPGAs : Logic Cells & gates count



- XX = XX million de portes logiques en 2005
- 1 Logic cell = 4-input LUT + FF
- Convention : 1 porte = 4 transistors
- Equivalent porte logique n'est pas une bonne métrique



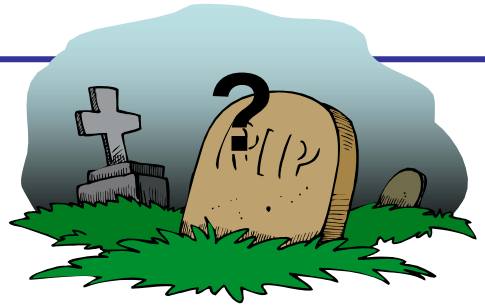
Partie 1 : conclusion

ASICs are dead…

My Position [Jonathan Rose]

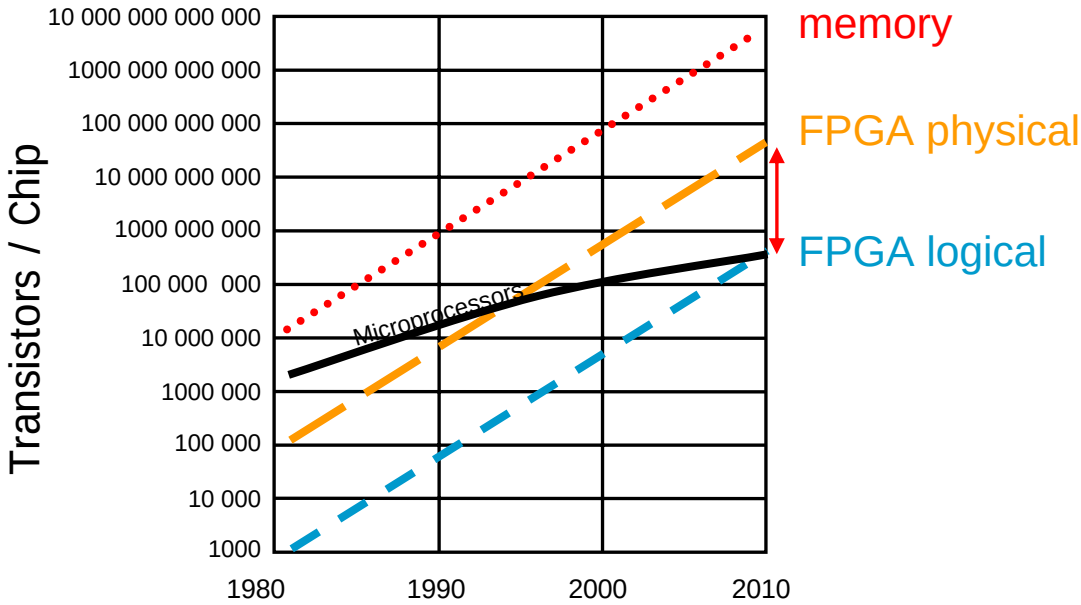
They Just Don’t Know It Yet!

Partie 1 : conclusion



But …

- **Coût de la reconfiguration**
- **Consommation**
- **Routage complexe**
- **~ 50 Millions de bits de Configuration !!**



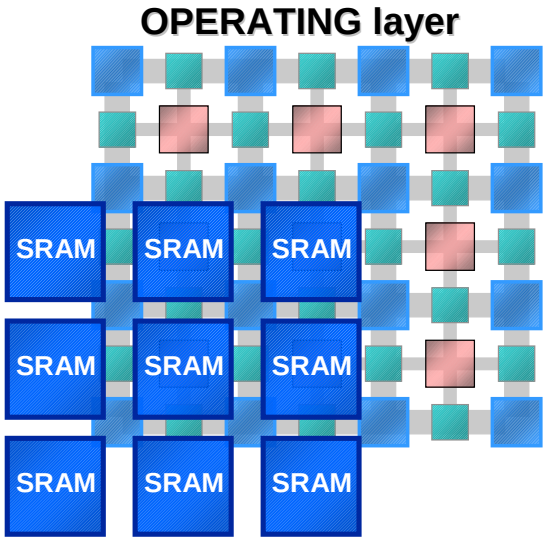
Partie 1 : conclusion

Evolutions des FPGAs

FPGAs :

RECONFIGURATION STATIQUE

- Chargement séquentiel des configurations
- une architecture par application



CONFIGURATION layer :
- **LUT configuration**
- **CLBs / IOBs /Switch matrixes bits**

FPGAs, évolutions:

RECONFIGURATION DYNAMIQUE (RTR: Run-Time Reconfiguration)

- Plusieurs configurations (multiple memories/LUT)
- une seule configuration, les fonctions peuvent changer en cours de fonctionnement

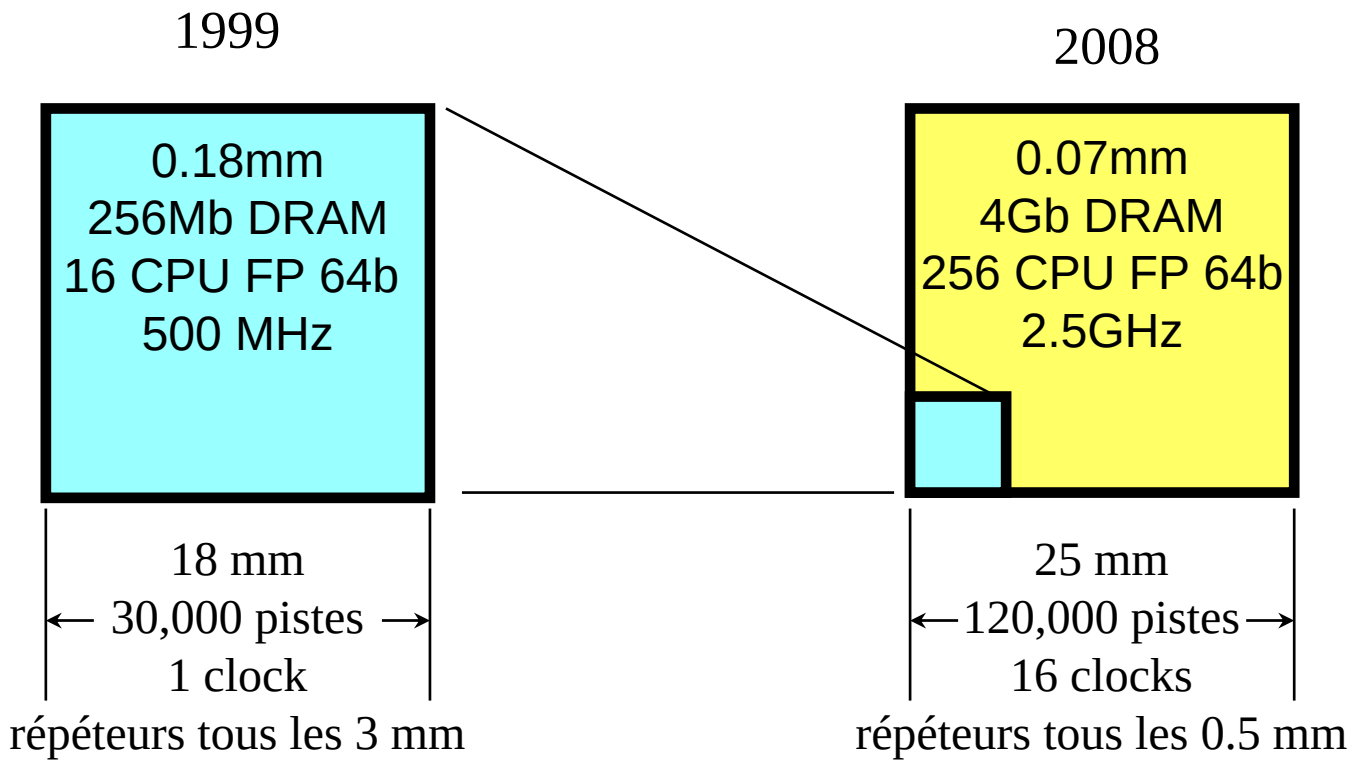
Sommaire

Partie 2 : Les architectures reconfigurables dynamiquement

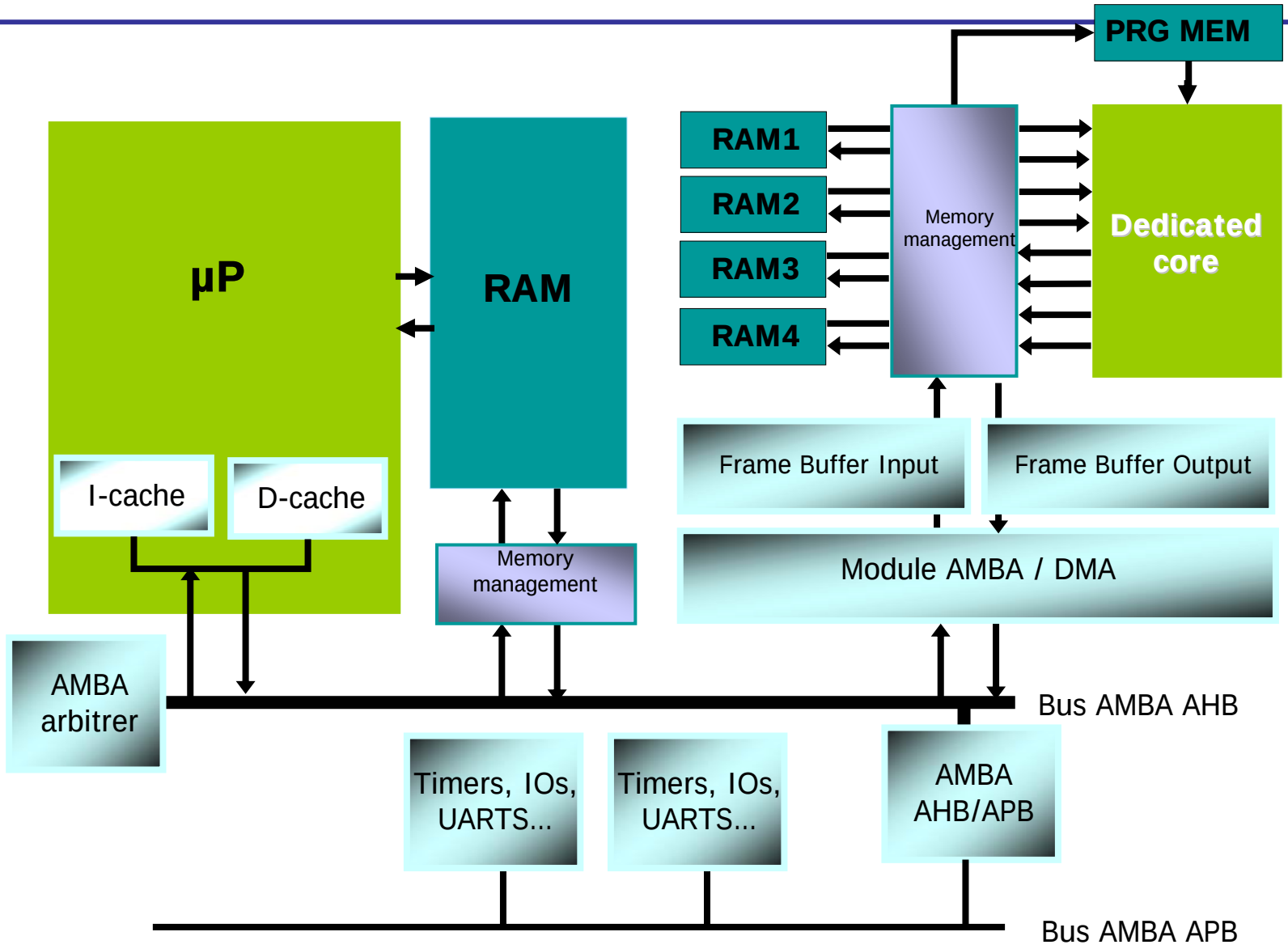
1- INTRODUCTION & CONTEXTE

Technologies microélectroniques en 2005 :

- Technologie de 90nm (voir 65nm)
- Intégration courante de plusieurs dizaines de millions de transistors… **1 milliard en 2004**
- Fréquence de fonctionnement supérieures à 2GHz

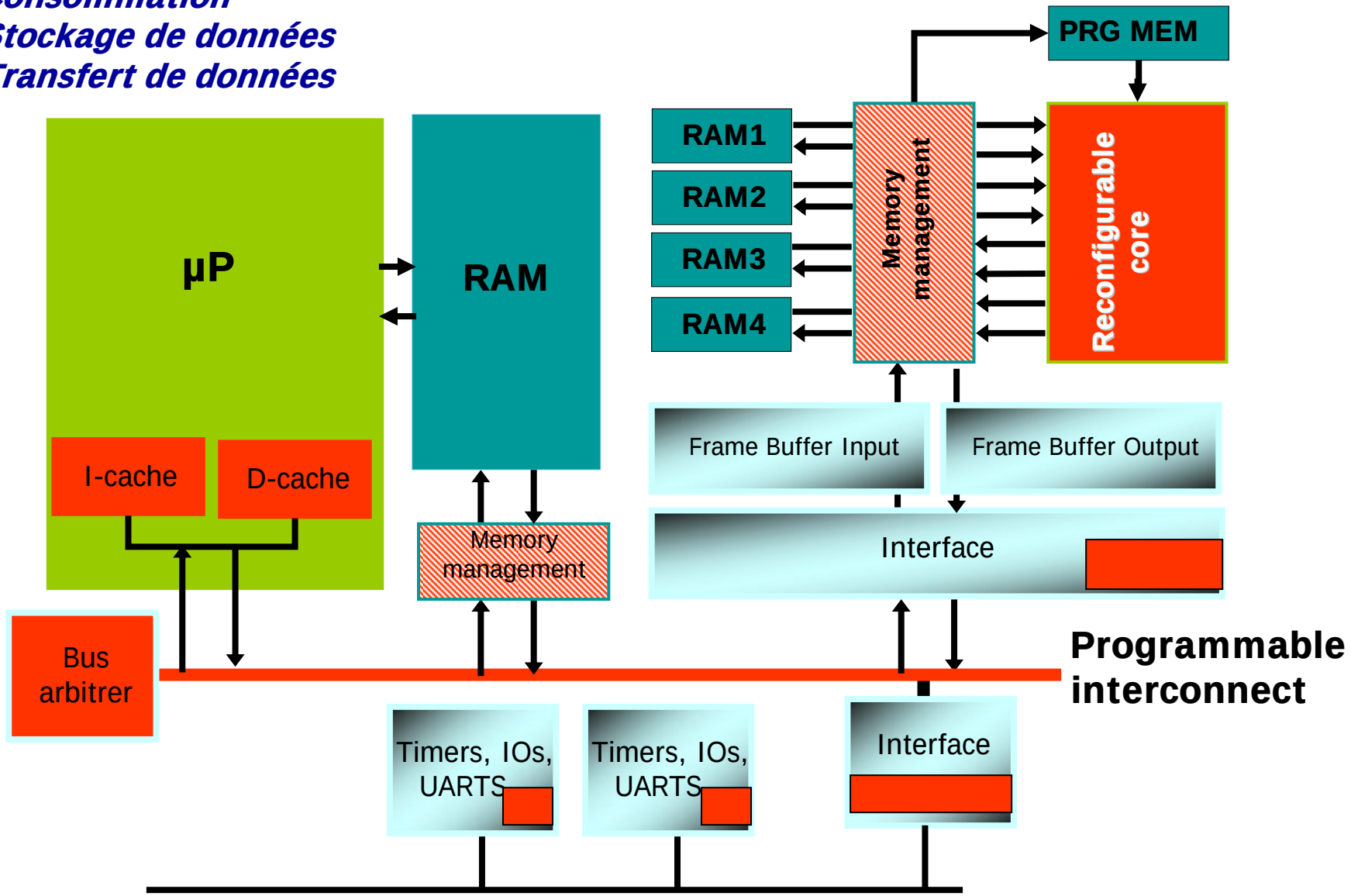


1- INTRODUCTION & CONTEXTE



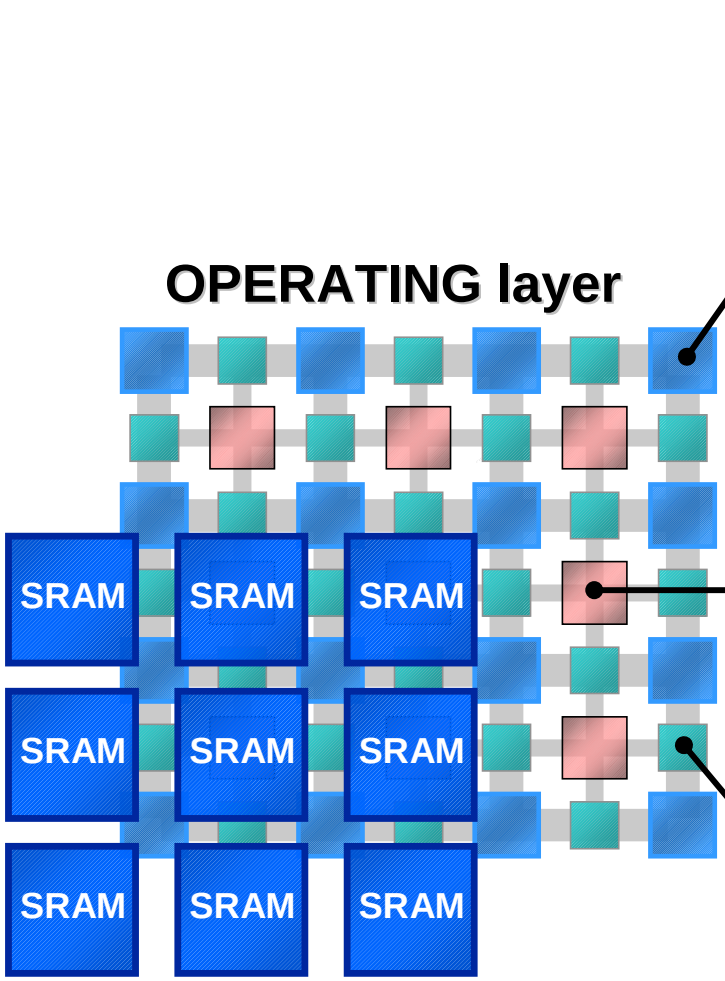
1- INTRODUCTION & CONTEXTE

- ***Consommation***
- ***Stockage de données***
- ***Transfert de données***

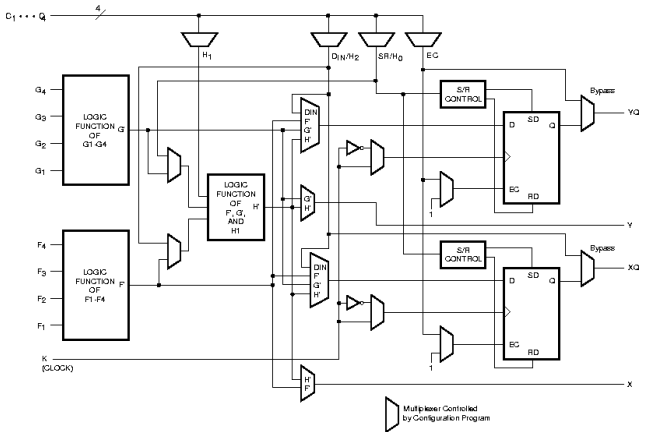


1- INTRODUCTION & CONTEXTE

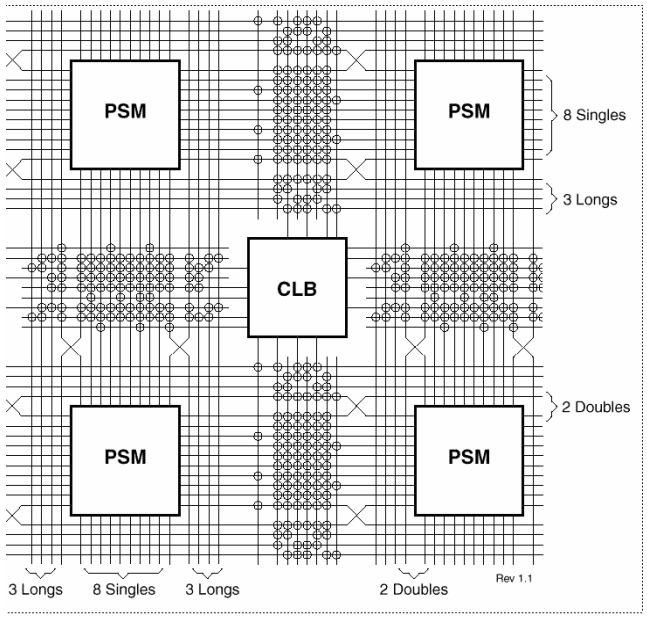
CLB, LC : bit-level processing unit (Look-Up-Tables)



CLB, LC…



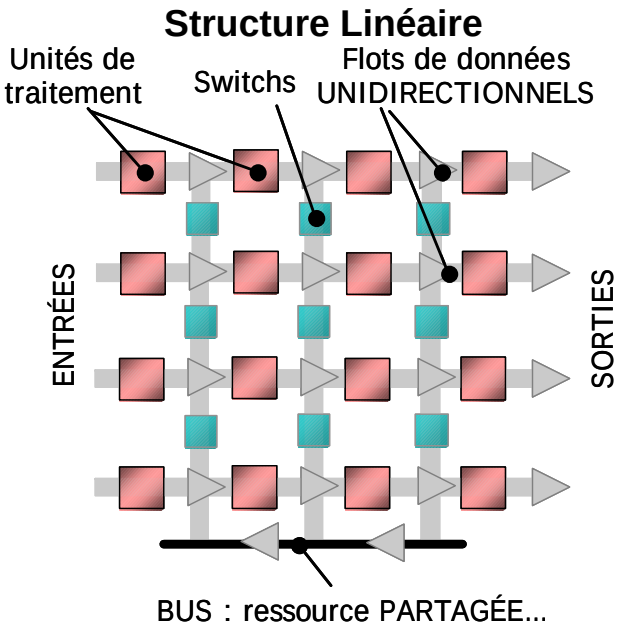
Global interconnect



Local interconnect

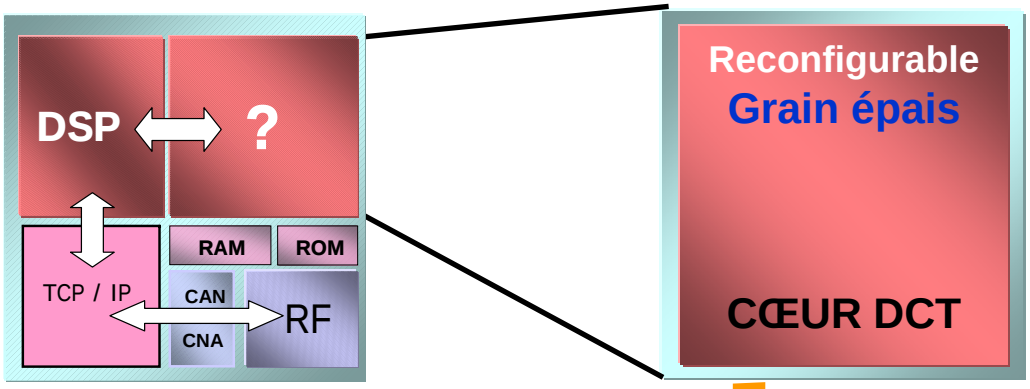
1- INTRODUCTION & CONTEXTE

- Reconfiguration
  - Interconnexions
  - Logiques
- Statique (en général)
- Optimisation de l'architecture pour des calculs niveau bit - contrôle
- Temps de reconfiguration, quantité d'information
- Points faibles : Interconnexion



- Le routage devient critique
  - Temps processeur de placement/routage important
  - Difficile de dépasser les 80-90 % de la capacité
  - Croissance non linéaire avec la taille du composant

1- INTRODUCTION & CONTEXTE



Approche Reconfigurable

☺ Flexible

Specific approach

☺ Efficient solution

☹ Restricted field of applications

Grain fin (FPGA)

☹ Peu efficace

Grain épais

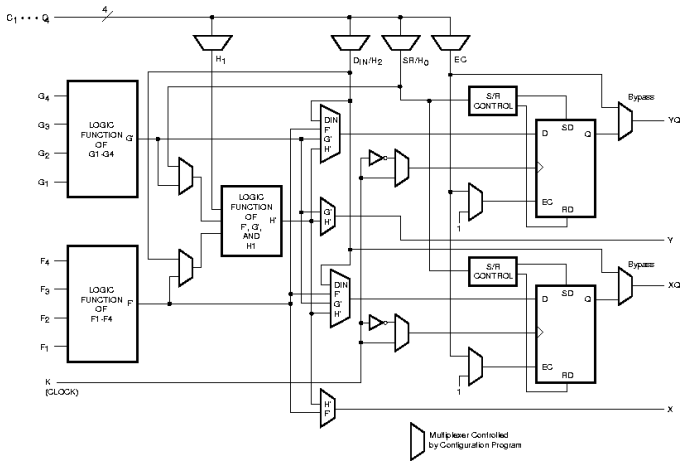
☺ Applications orientées flots de données

1- INTRODUCTION & CONTEXTE

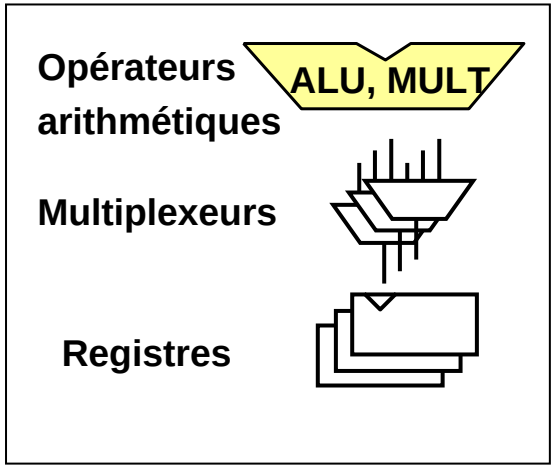
- **Accélération des applications orientées flots de données**
- **Extrapolation aisée sur de nouvelles technologies**
- **Flexibilité d'utilisation maximale**
- **Personnalisation aisée**

→ GRAIN EPAIS
- Faible surcoût

Grain fin

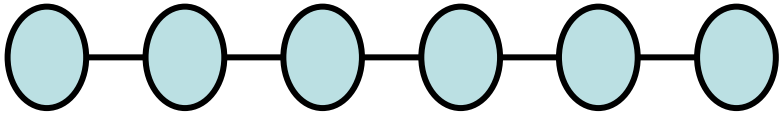


Grain épais

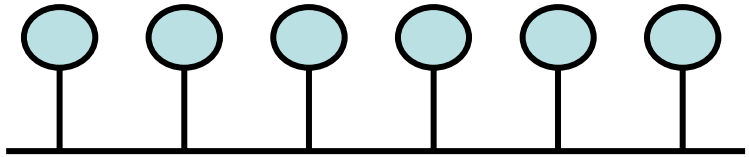


1- INTRODUCTION & CONTEXTE

1D array

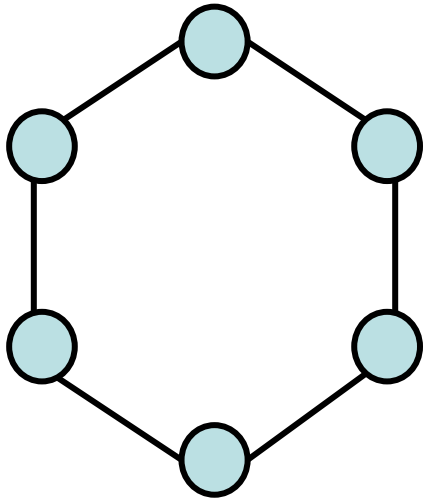


Bus

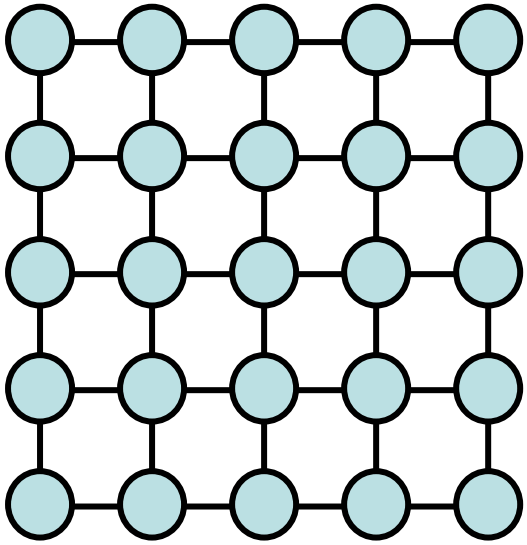


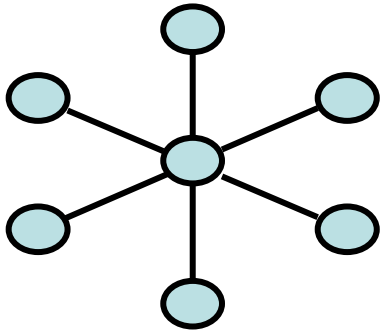
Star

Ring

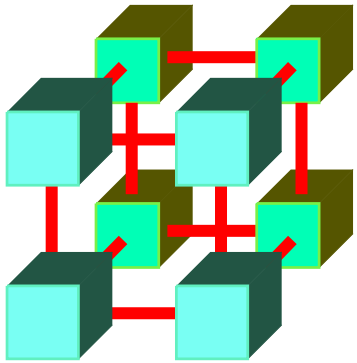


2D array

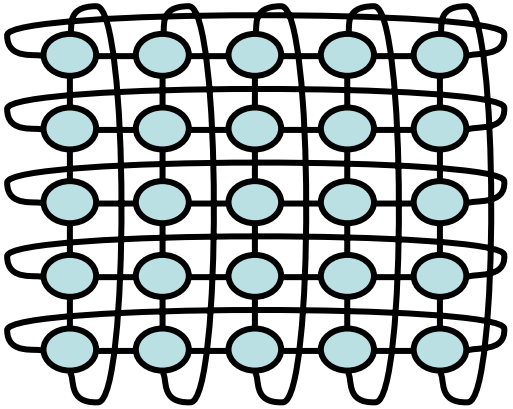




3D array

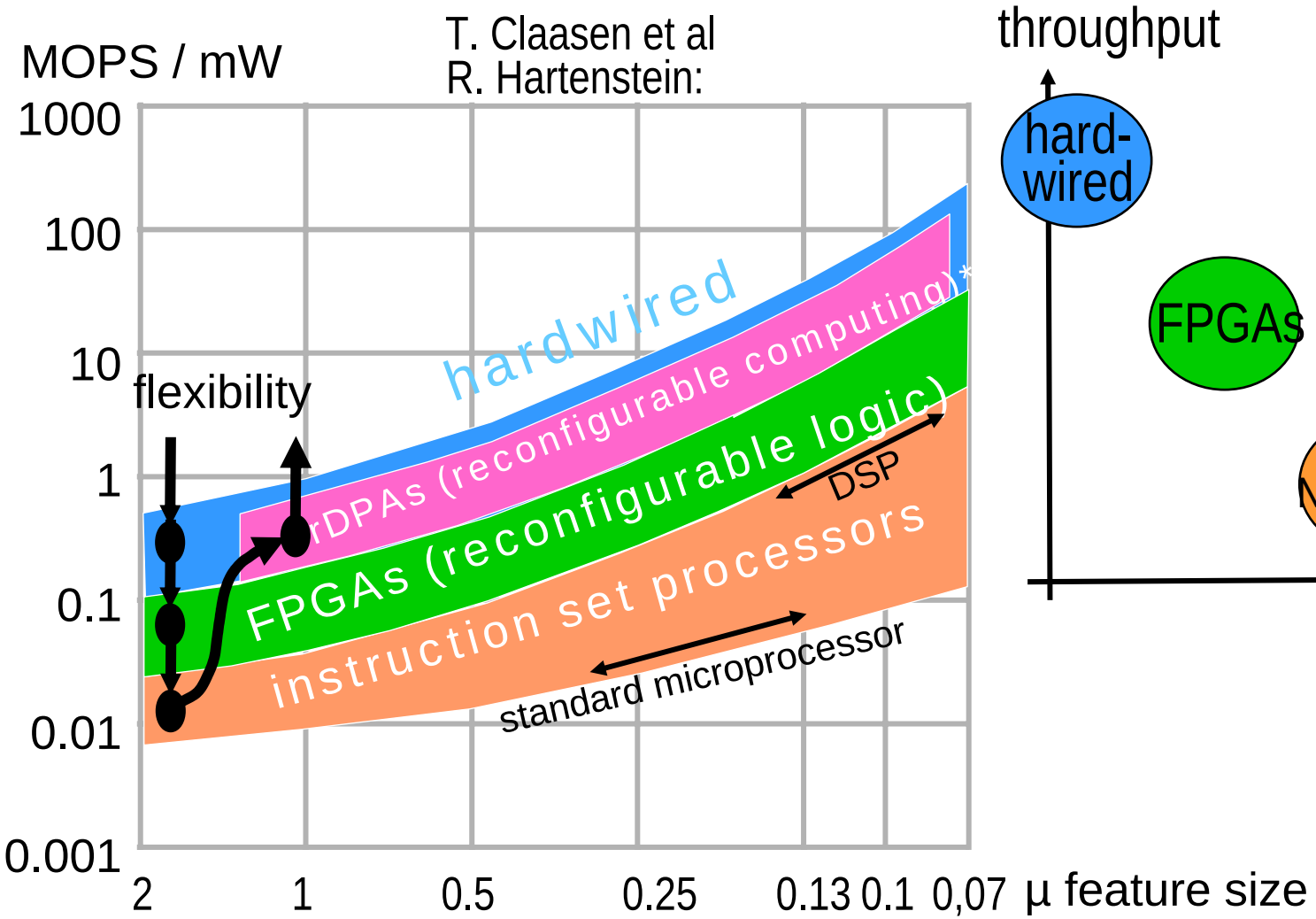


Torus



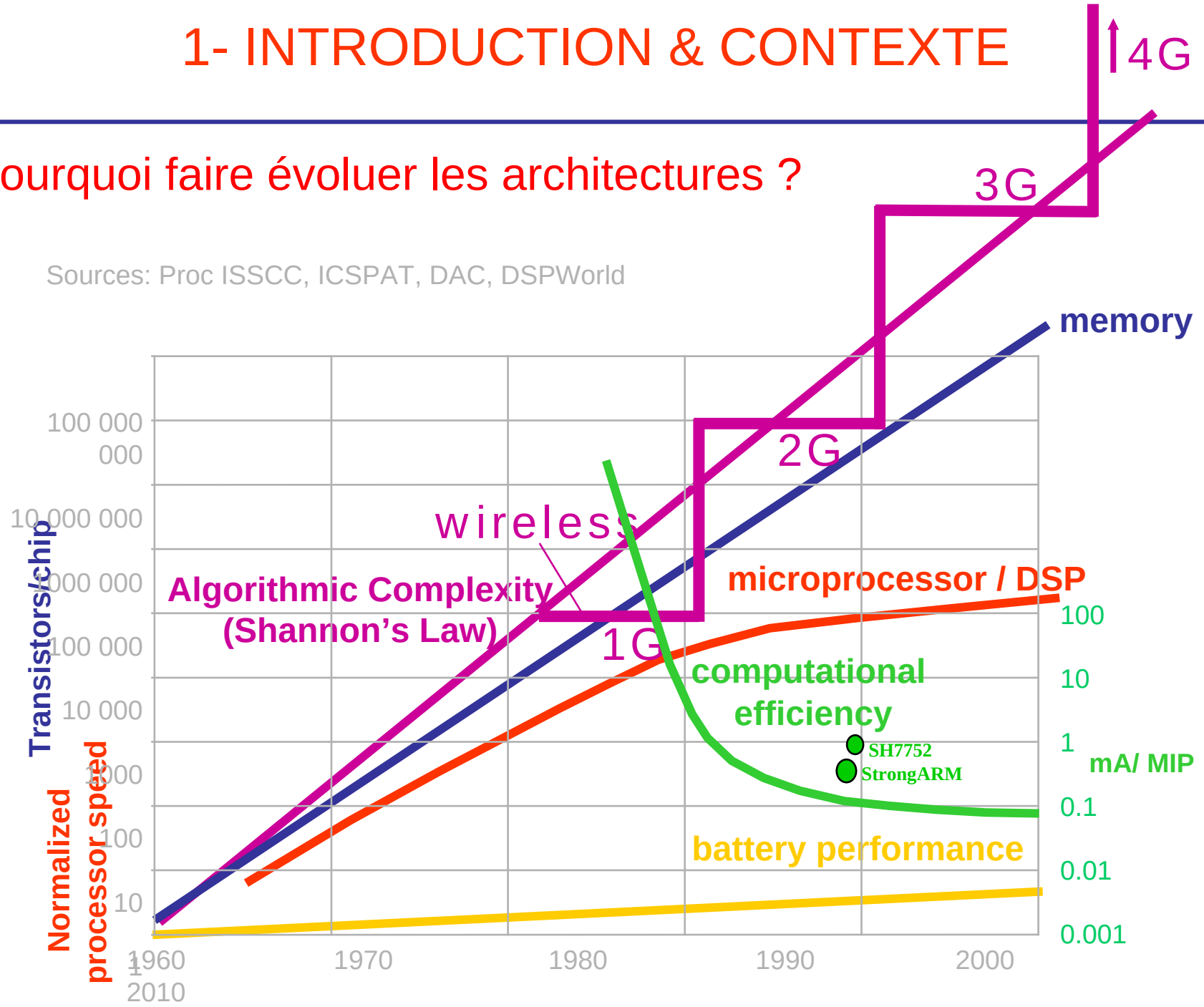
1- INTRODUCTION & CONTEXTE

Energy Efficiency vs. Flexibility



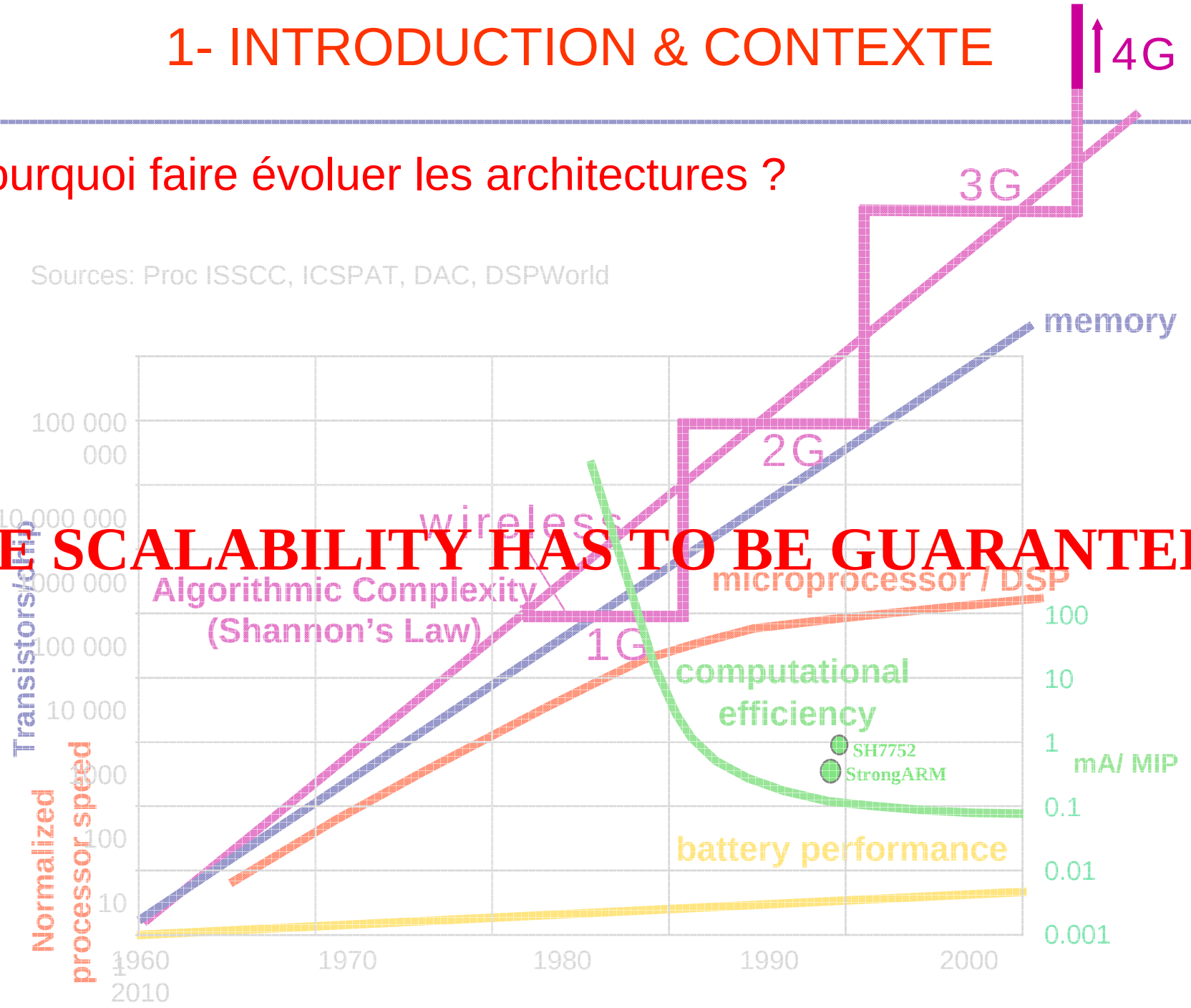
1- INTRODUCTION & CONTEXTE

Pourquoi faire évoluer les architectures ?



1- INTRODUCTION & CONTEXTE

Pourquoi faire évoluer les architectures ?



THE SCALABILITY HAS TO BE GUARANTEED

Sommaire

Partie 2 : Les architectures reconfigurables dynamiquement

1- INTRODUCTION & CONTEXTE
2- CLASSIFICATION D'ARCHITECTURES
3- QUELQUES EXEMPLES
4- EXEMPLE DE GESTION DYNAMIQUE
5- CONCLUSIONS & PERSPECTIVES

2- CLASSIFICATION D'ARCHITECTURES

RECONFIGURATION STATIQUE

- **Chargement séquentiel de plusieurs architectures**
- **Une architecture par application**

RECONFIGURATION PSEUDO-DYNAMIQUE

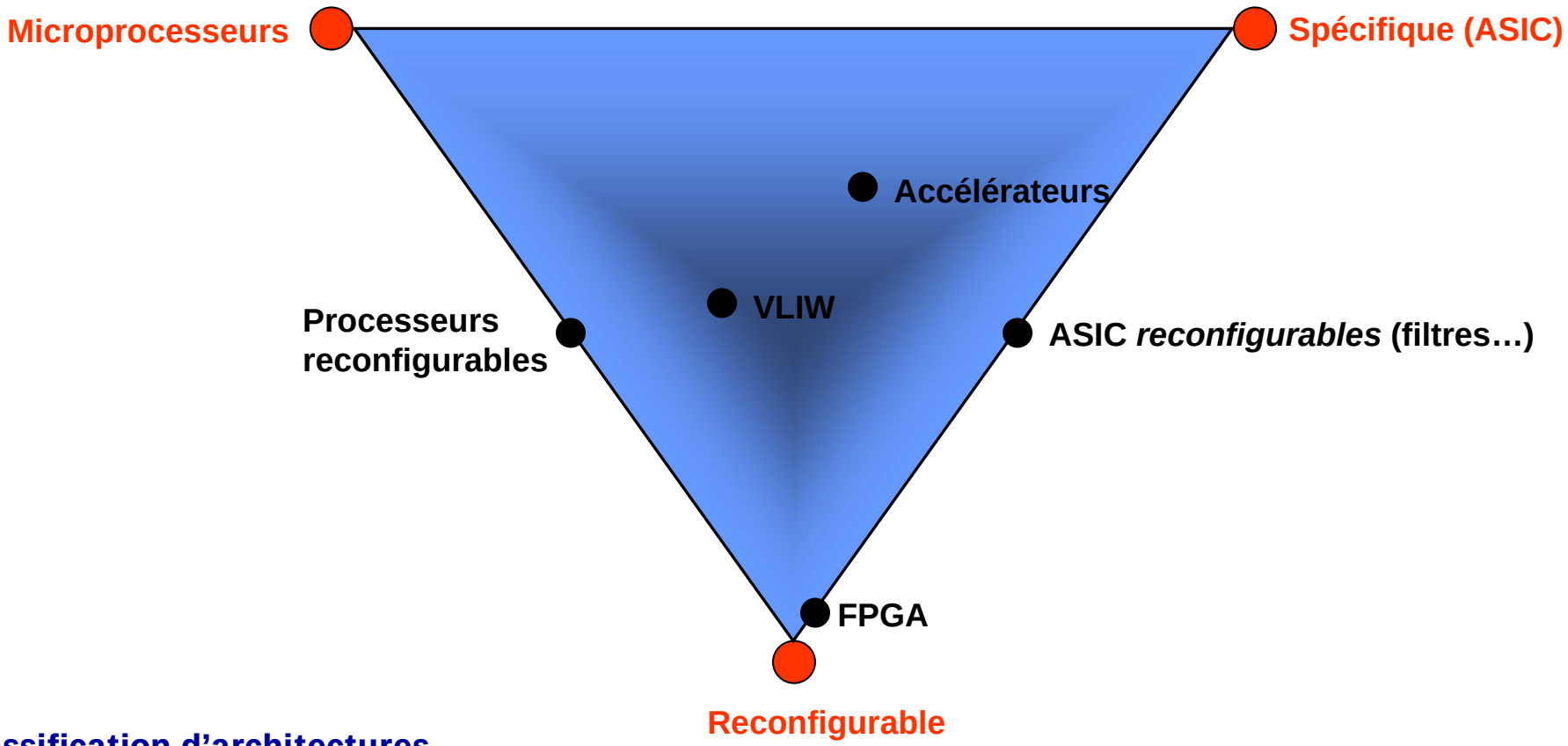
- **Chargement séquentiel de plusieurs architectures**
- **Plusieurs architecture par application**

RECONFIGURATION DYNAMIQUE

- **Chargement continu**
- **L'architecture évolue en cours de traitement**

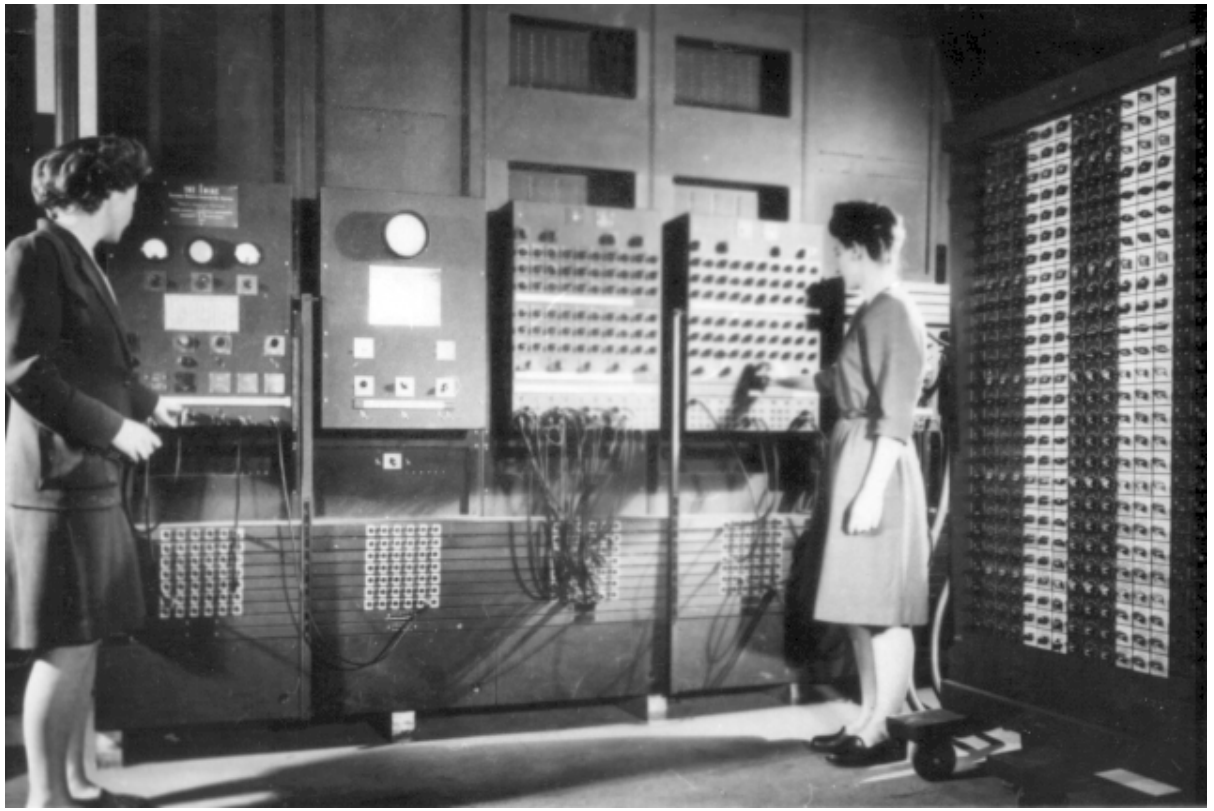
2- CLASSIFICATION D'ARCHITECTURES

Traitement numérique du signal : trois familles d'architectures candidates



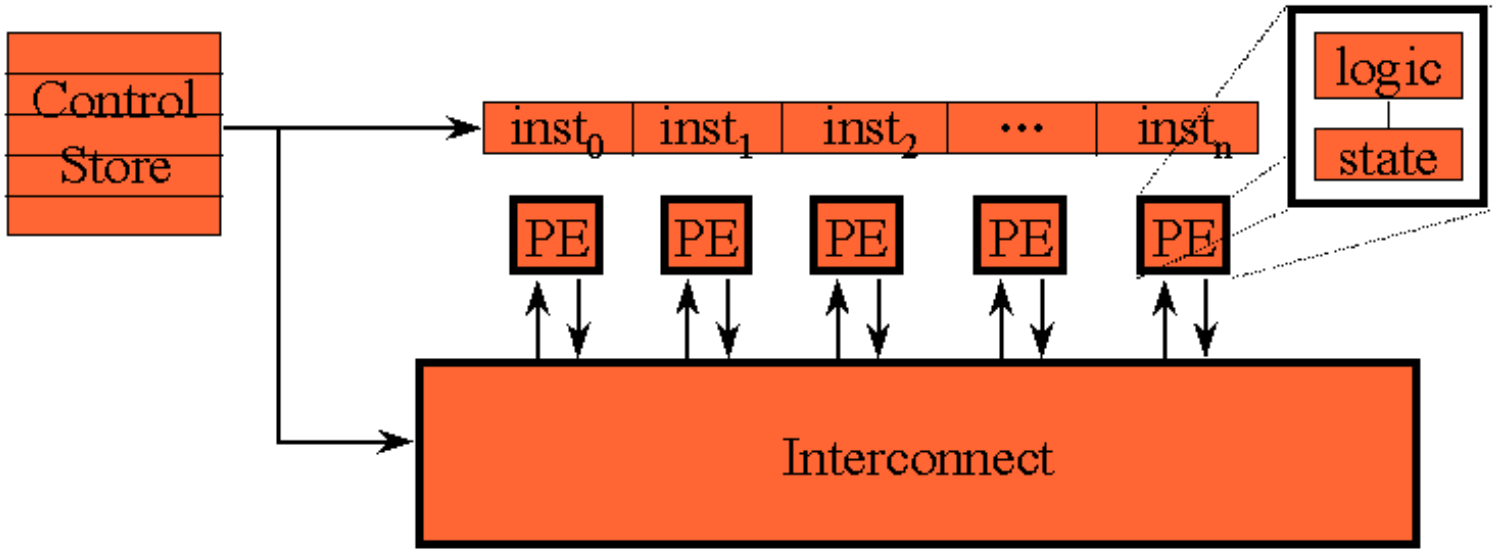
2- CLASSIFICATION D'ARCHITECTURES

Processeur = Machine Reconfigurable

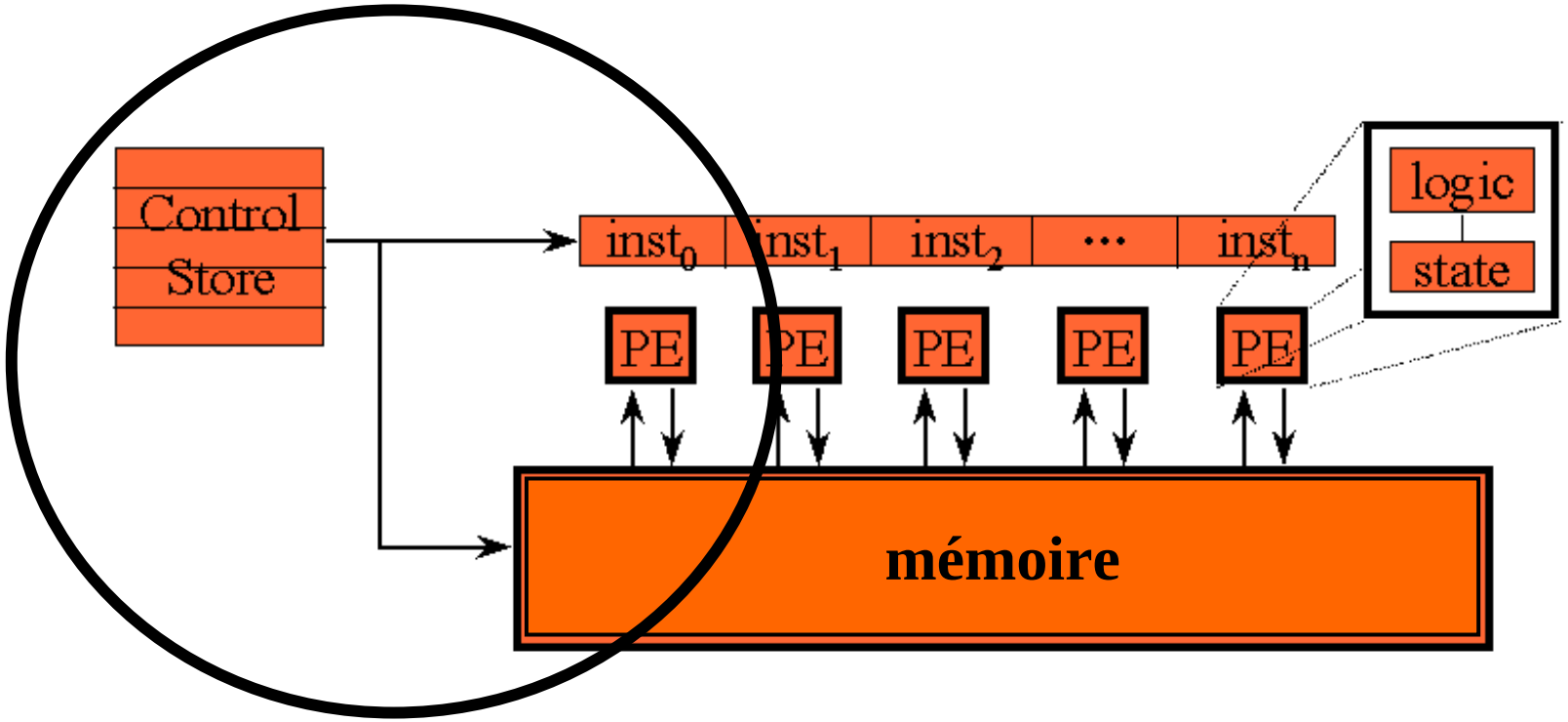


ENIAC

2- CLASSIFICATION D'ARCHITECTURES



2- CLASSIFICATION D'ARCHITECTURES



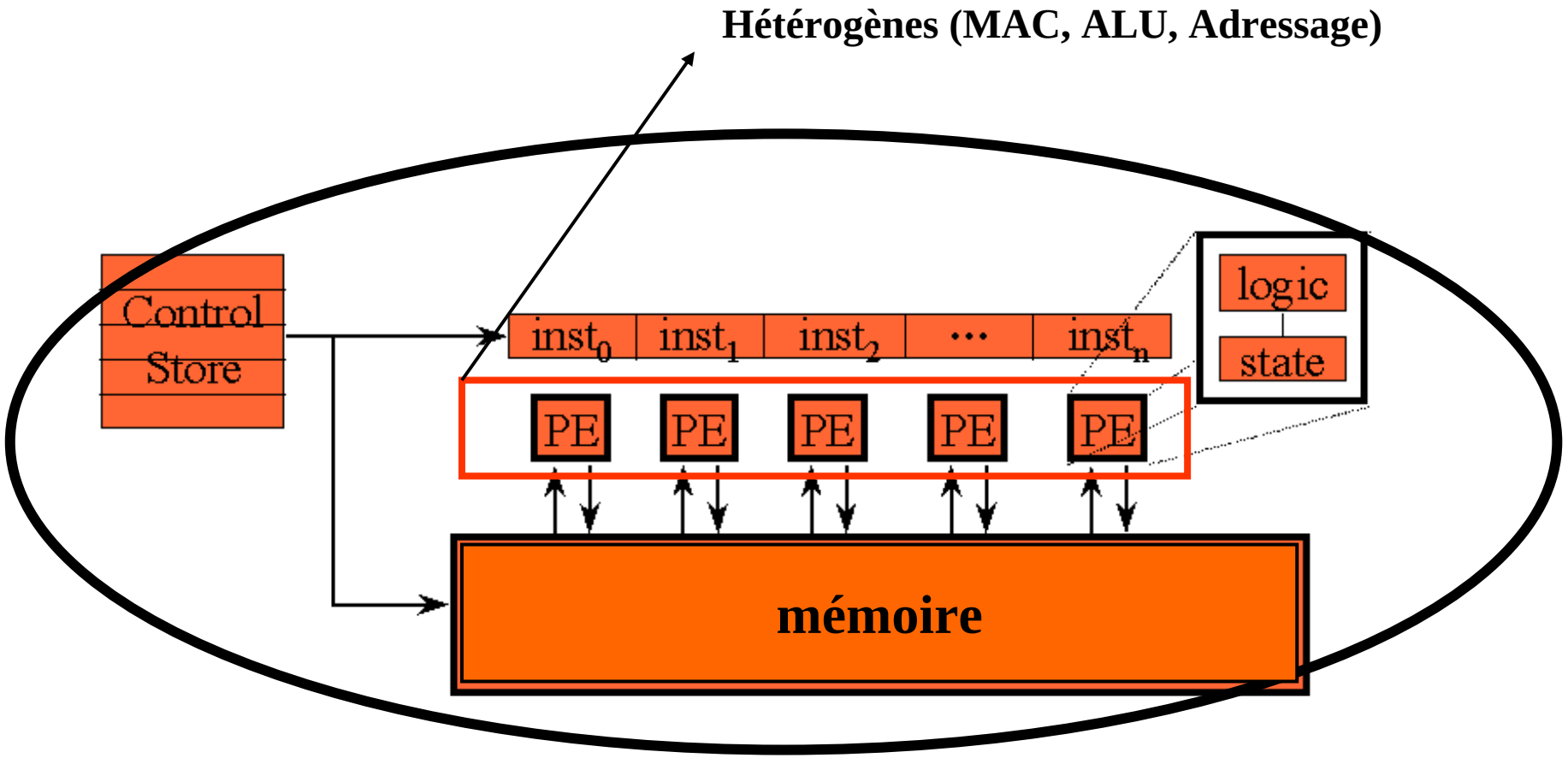
•**CISC**, *Complex Instruction Set Computer* :

- Circuits imposants
- Performances faibles
- Compilation difficile

•**RISC**, *Reduced Instruction Set Computer* :

- Performances plus élevées
- Compilation aisée
- Taille de code importante

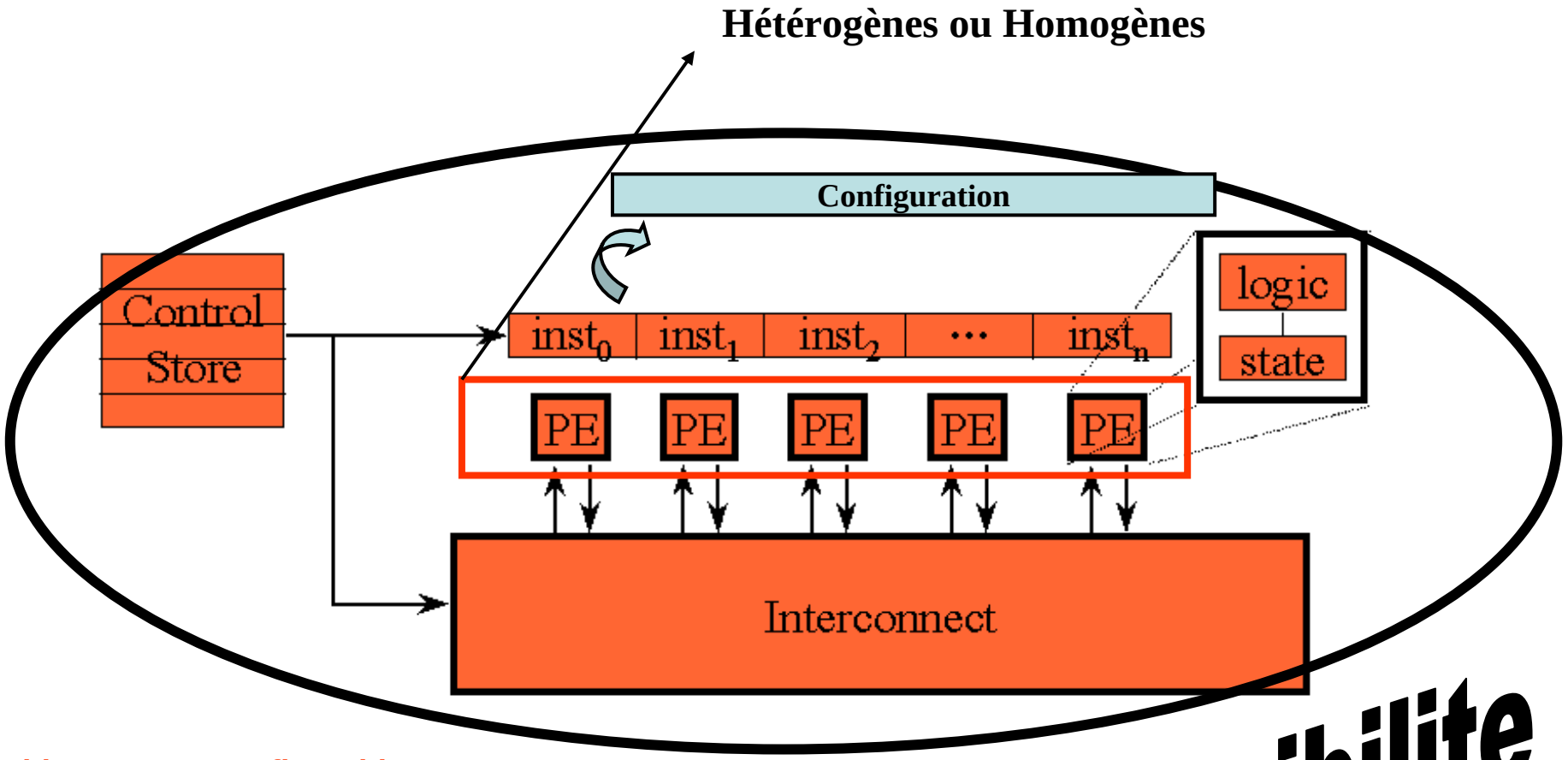
2- CLASSIFICATION D'ARCHITECTURES



•**VLIW**, *Very Long Instruction Word :*

- Performances élevées
- Présence de compilateur (pas forcément optimal – rapport de 20 entre C et assembleur)
- Parallélisme limité vs ARD

2- CLASSIFICATION D'ARCHITECTURES



Flexibilite

•**Architectures reconfigurables** *:*

- Paradigme de machine : basée sur de la RAM
- Exécution des opérations : parallèles
- Description de l'application : langages matériels

Classification d'architectures

2- CLASSIFICATION D'ARCHITECTURES

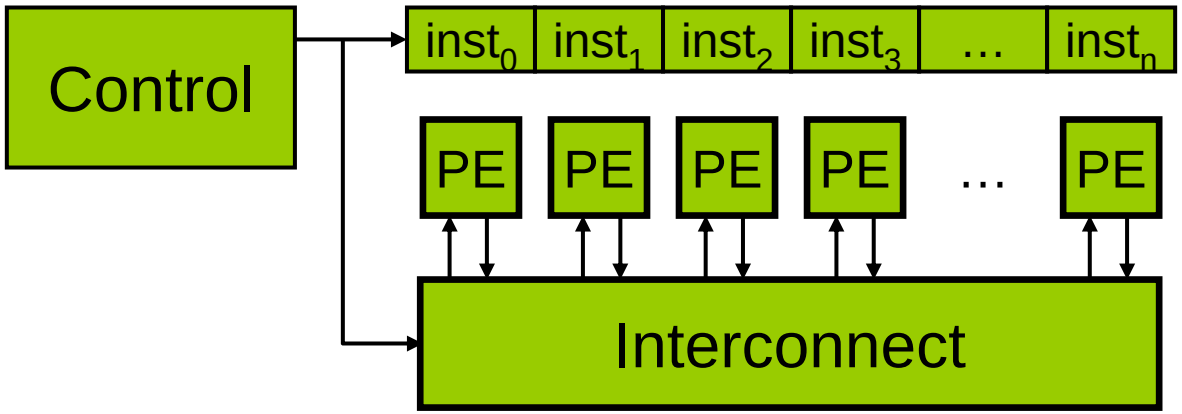
Charnière Von Neumann – Architectures reconfigurables

VLIW

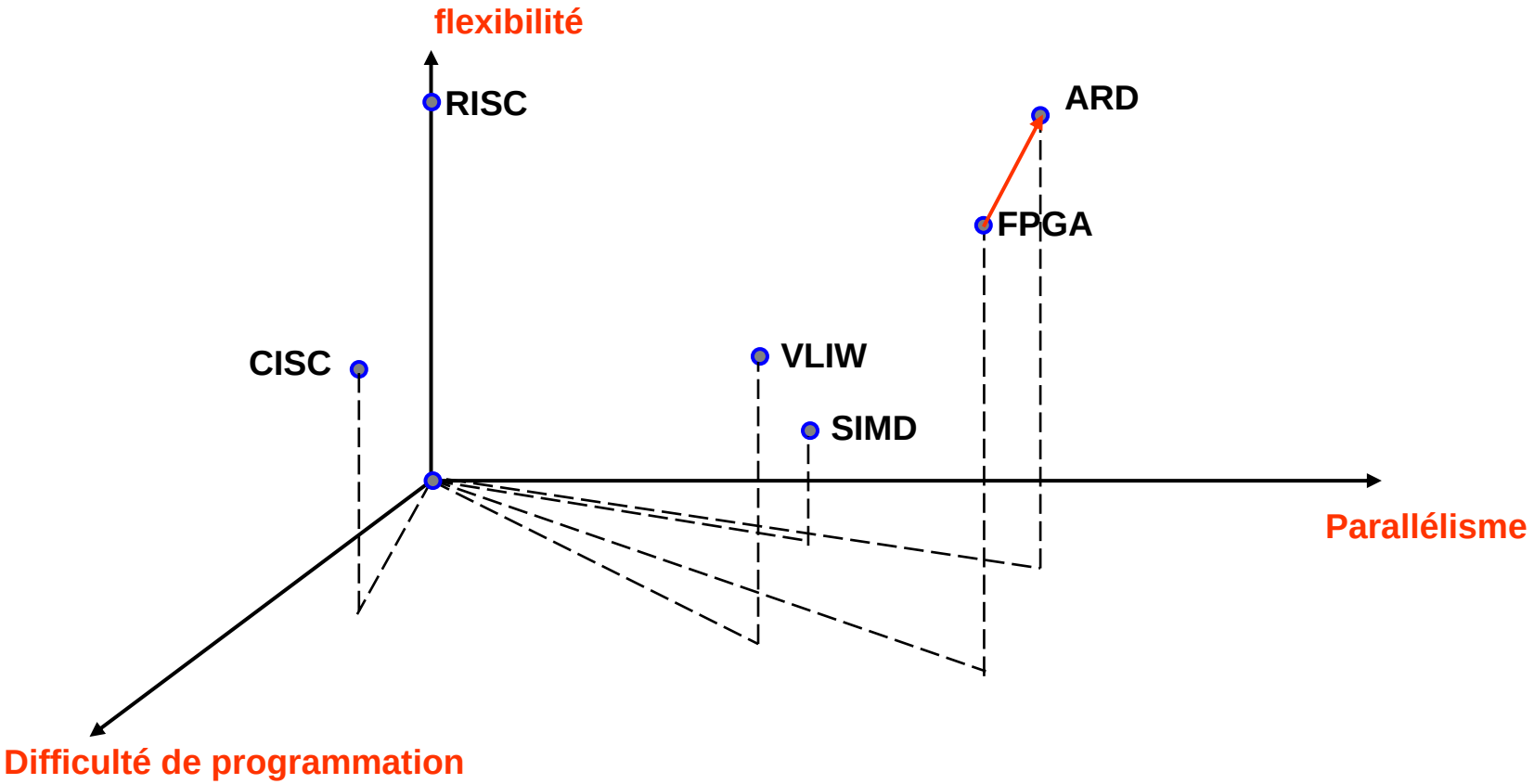
- Reconfiguration de tous les PE chaque cycle
- Topolgie 1D : Reg op Reg → Reg
- PE principalement hétérogènes (MAC, ALU, MULT)
- Présence d'un compilateur (pas optimal)
- Support des opération conditionnelles (JE, JZ..)
- Communications avec une BR multi-ports

AR grain épais

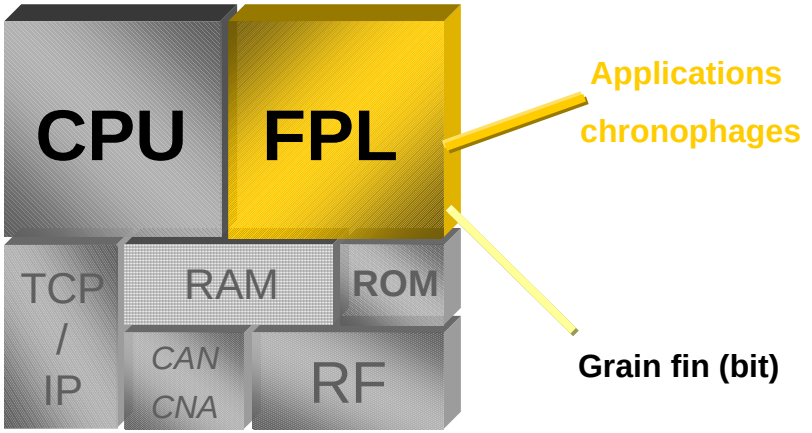
- Reconfiguration d'une partie des PE (reconf. dynamique)
- Topologie 2D : Datapath pipelinés
- PE souvent homogènes (Op + Reg)
- Programmation souvent bas niveau (assembleur, VHDL)
- Pas d'opération conditionnelles (conf. Conditionnelle !)
- Communications avec plusieurs banques mémoires

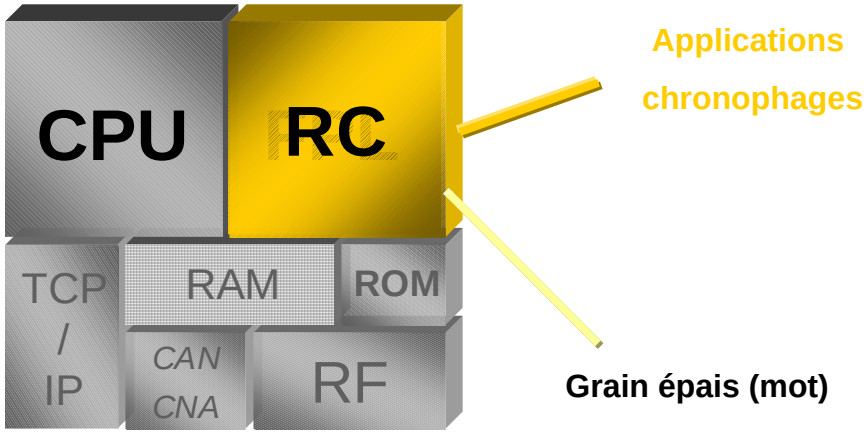


2- CLASSIFICATION D'ARCHITECTURES



2- CLASSIFICATION D'ARCHITECTURES





	Approche Programmée		Approche Spécifique	Approche Reconfigurable	
	CPU	DSP	ASIC	FPL	RC
Puissance de traitement	+	++	++++	++ +?	+++
Faible Consommation	-	+	++++	?	++
Flexibilité	++++	++	----	+++	++

2- CLASSIFICATION D'ARCHITECTURES

- Choix et Grain des éléments de calcul
- Choix et Grain du réseau d'interconnexion
- Temps et taux de (Re)configuration

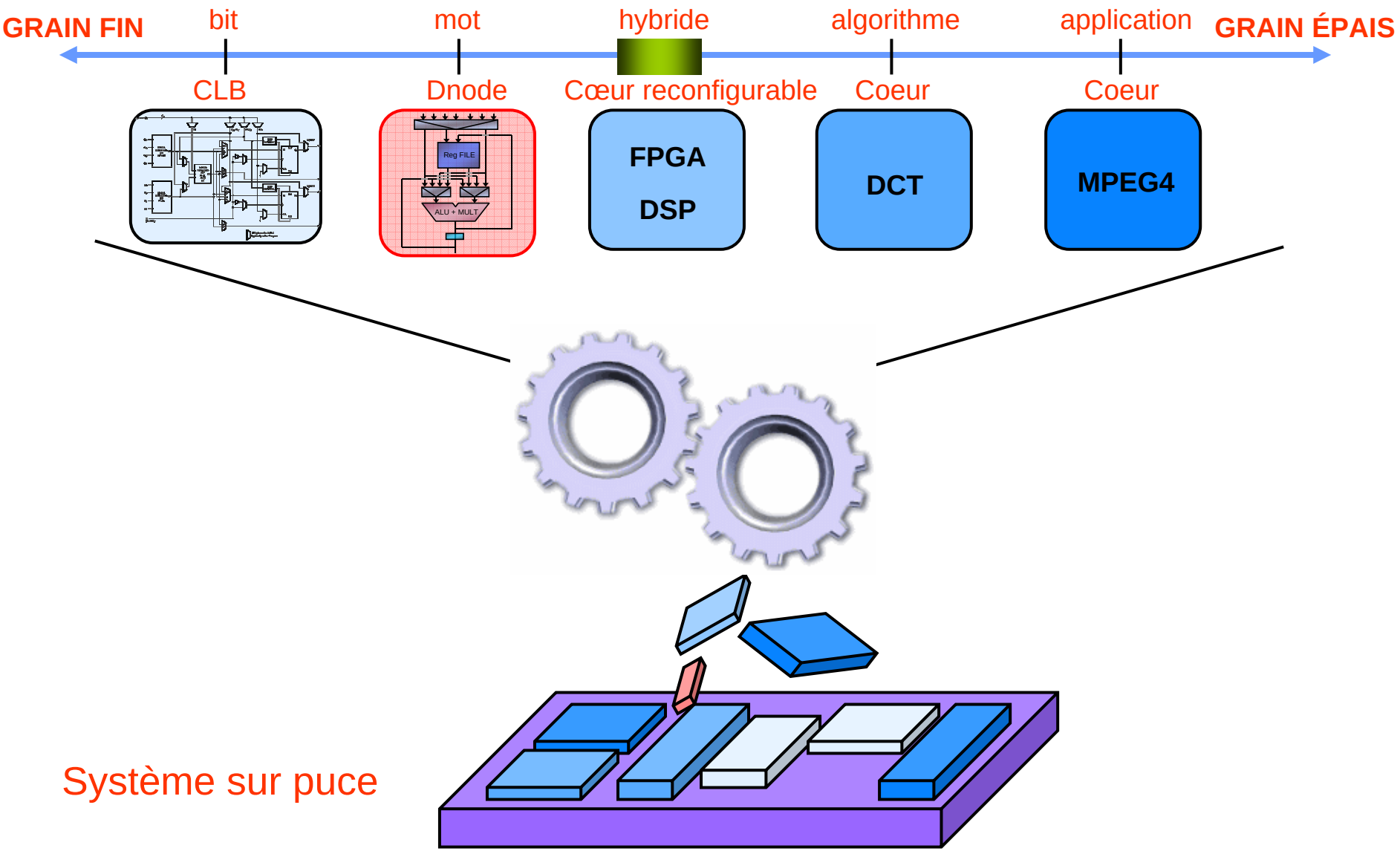
  - Pour
    - *Consommation*
    - *Stockage de données*
    - *Interconnexion flexible*
    - *Ré-utilisation*
    - *Flexibilité*

2- CLASSIFICATION D'ARCHITECTURES

- Reconfiguration au niveau système
  - *Lx, C62 (décomposition en cluster)*
- Reconfiguration au niveau fonctionnel
  - *Pleiades, RaPiD, DART, SystolicRing*
- Reconfiguration au niveau opérateur
  - *Chameleon, Piperench, Morphosys*
- Reconfiguration au niveau porte
  - *Napa, GARP, FPGA*

Source : O. Sentiyes, S. Pillement – R2D2

2- CLASSIFICATION D'ARCHITECTURES

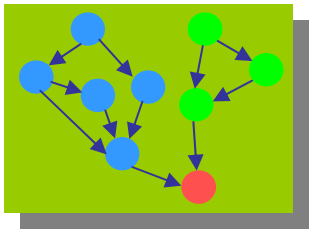


Système sur puce

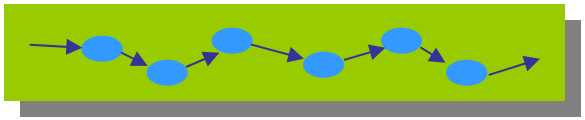
2- CLASSIFICATION D'ARCHITECTURES

Les différents niveaux parallélismes

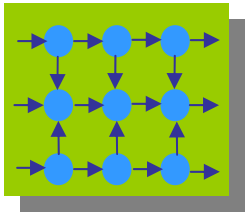
- **Instruction Level Parallelism**



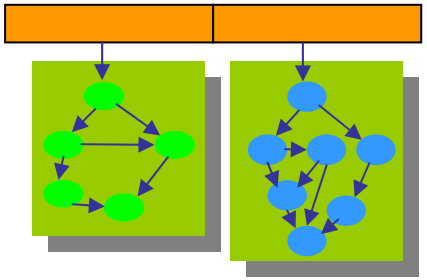
- **Pipeline**



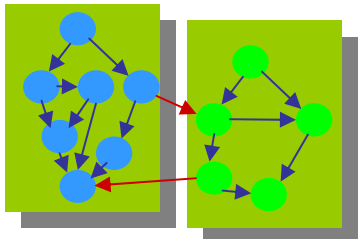
- **Systolic Operation**



- **Multiple Parallel Instances**



- **Multi-threading and Multitasking**



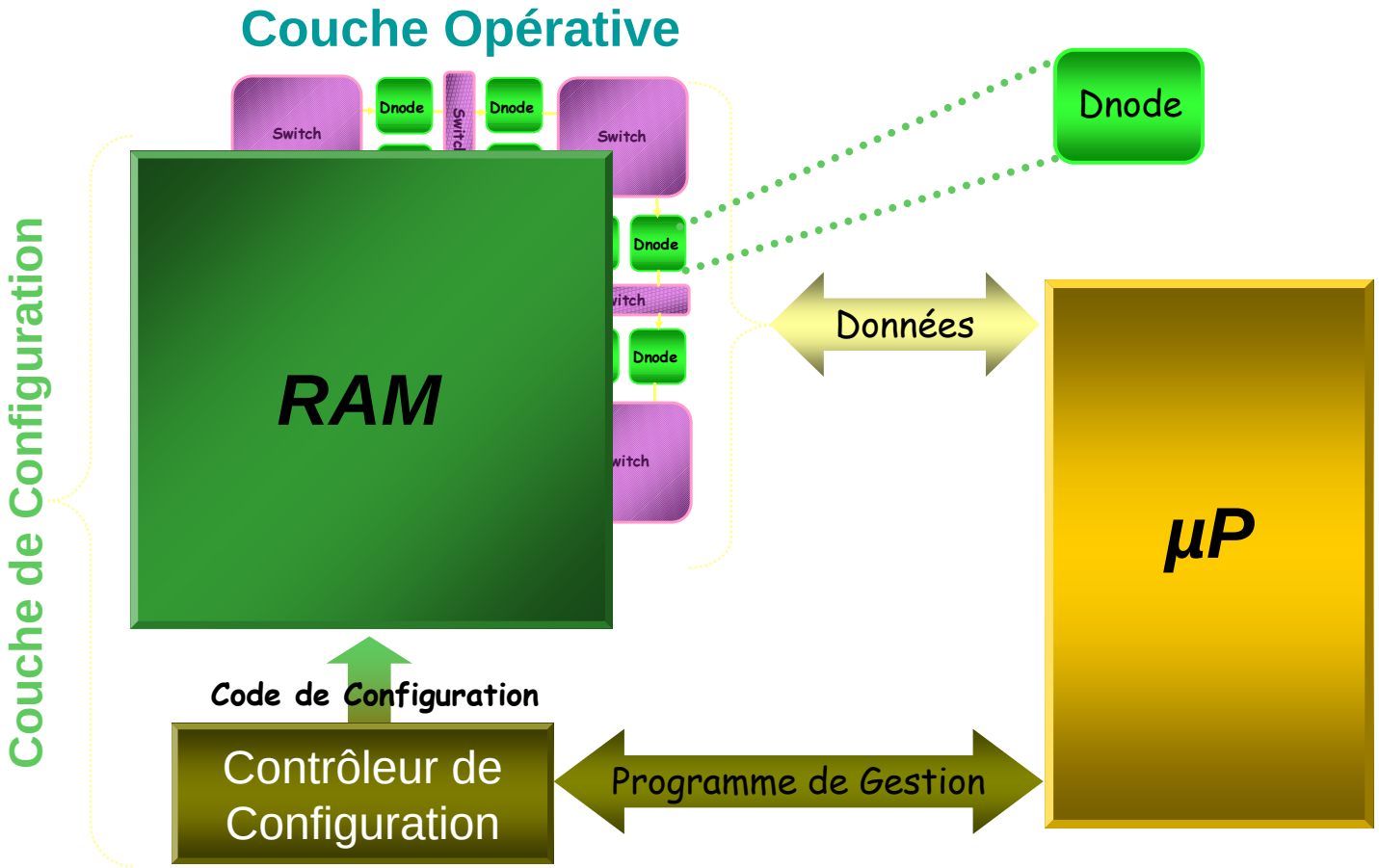
Sommaire

Partie 2 : Les architectures reconfigurables dynamiquement

1- INTRODUCTION & CONTEXTE
2- CLASSIFICATION D'ARCHITECTURES
3- QUELQUES EXEMPLES
4- EXEMPLE DE GESTION DYNAMIQUE
5- CONCLUSIONS & PERSPECTIVES

3- QUELQUES EXEMPLES

Systolic Ring (LIRMM)



Composant à grain épais

Reconfigurable dynamiquement

3- QUELQUES EXEMPLES

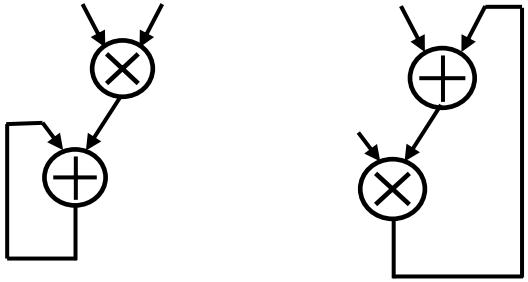
DNODE : Data Node

Constitution

- Chemin de données optimisé
- Banque de registres 4x16 bits
- ALU et multiplieur 16x16 câblés

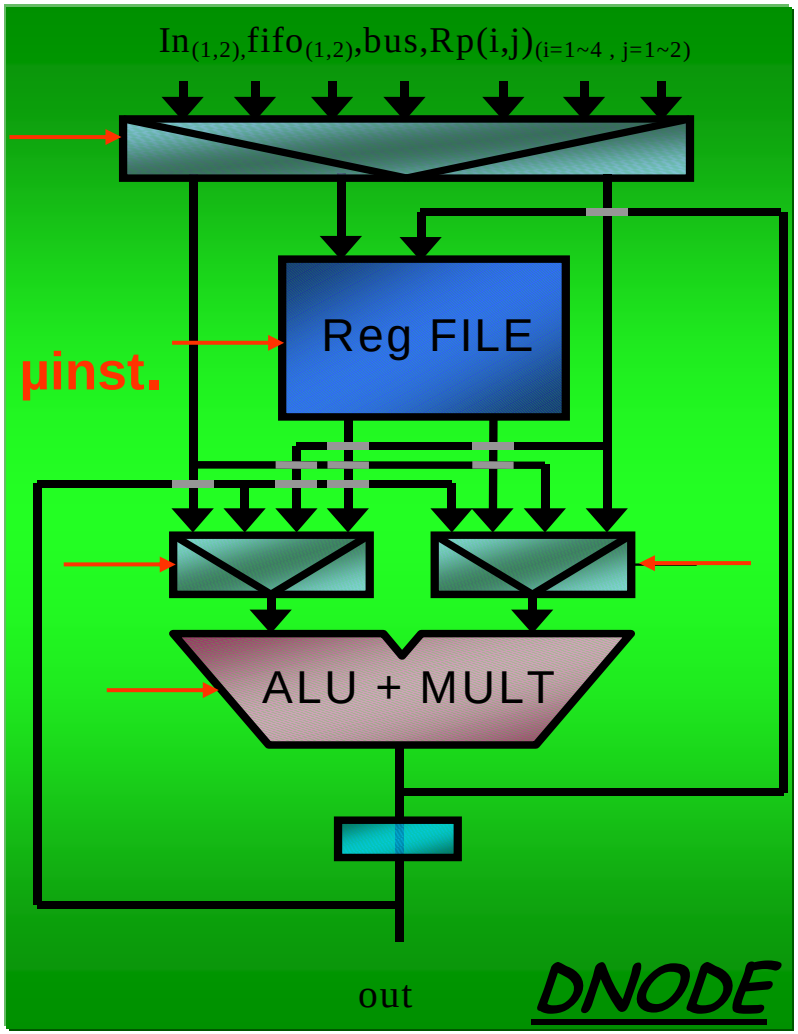
Spécificités

- Tout le jeu d'instructions en un unique cycle
- Multiplieur et additionneur fusionnables (unité MAC)



Format d'opérations

- op(input,input) → output,reg
- op(input,reg) → output,reg
- op(reg,reg) → ouput,reg

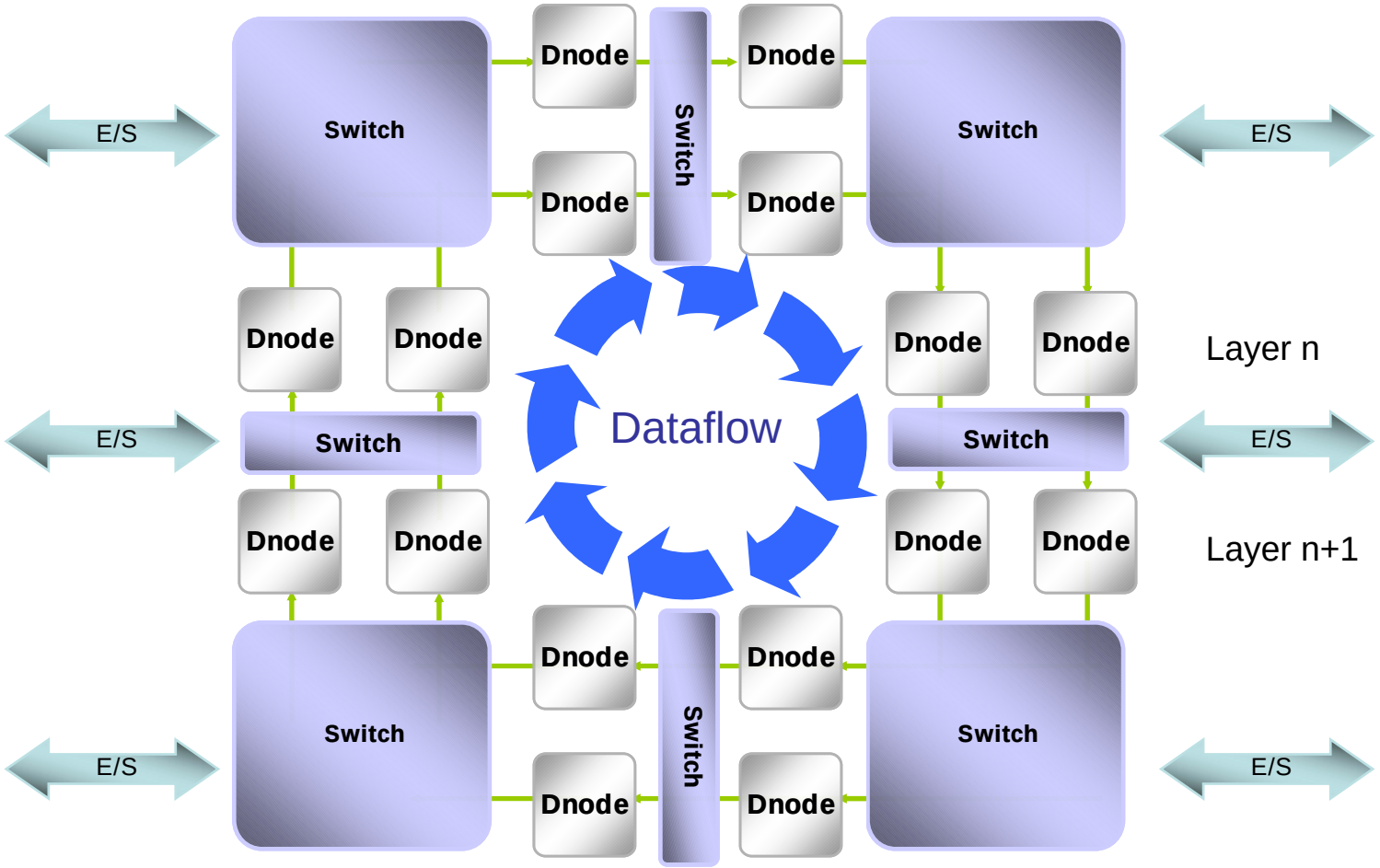


3- QUELQUES EXEMPLES

Systolic Ring architecture

Forward dataflow

- Peak power : 3200 MIPS@200MHz (16 Dnodes version)

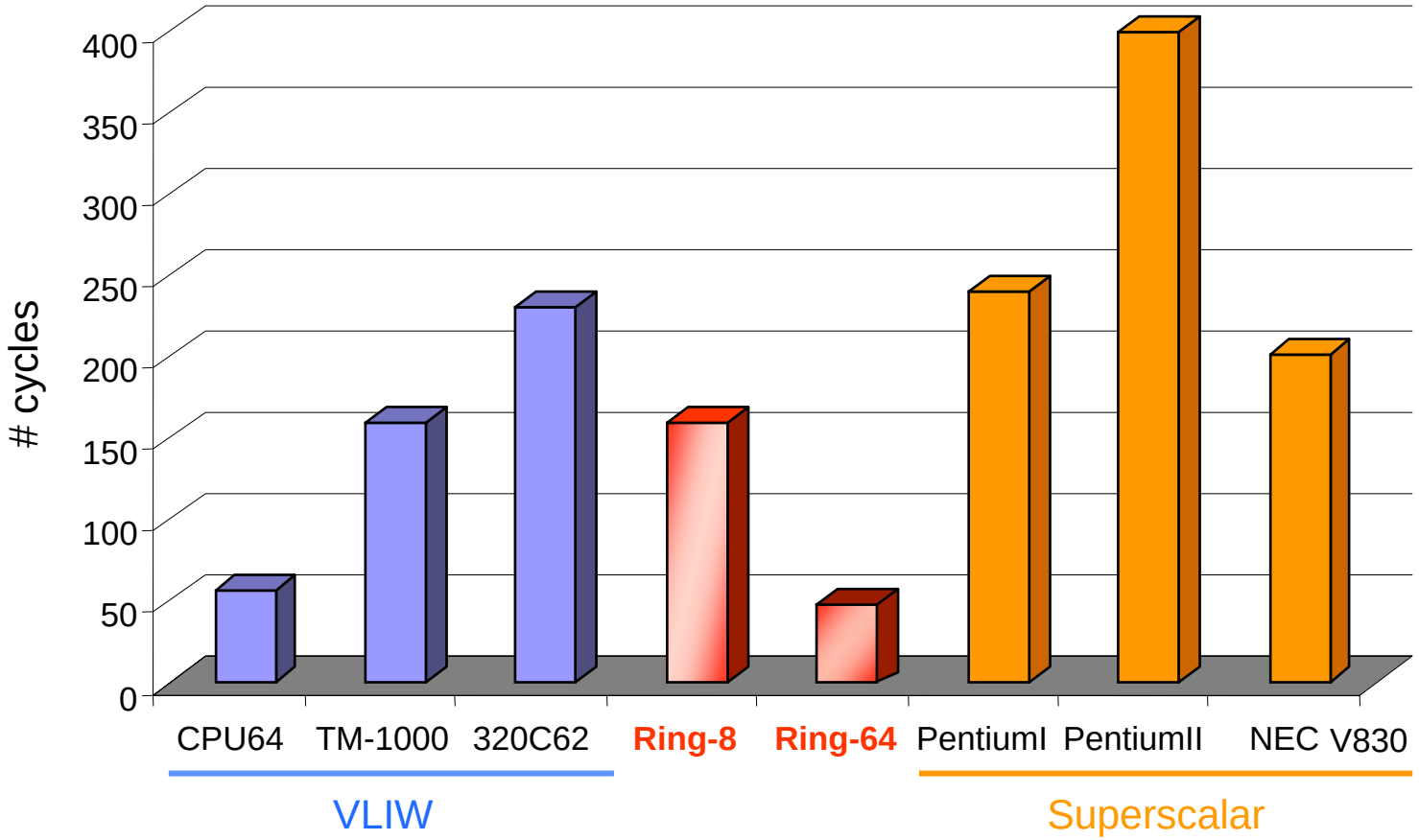


3- QUELQUES EXEMPLES

Comparaisons (cycles)

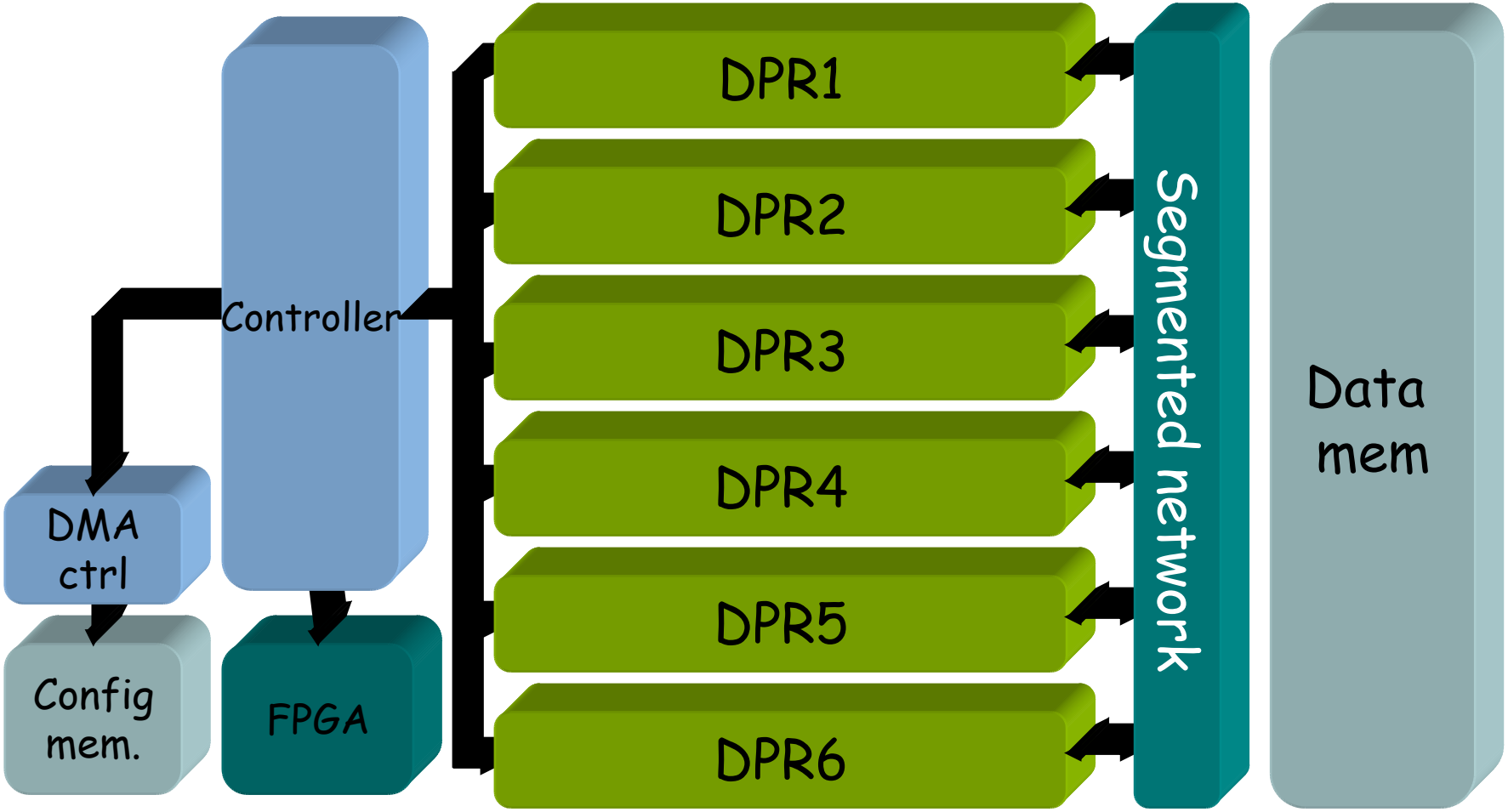
VLIW : CPU64, TM1000, TI 320C60

Superscalar : Pentium I, Pentium II, NEC V830



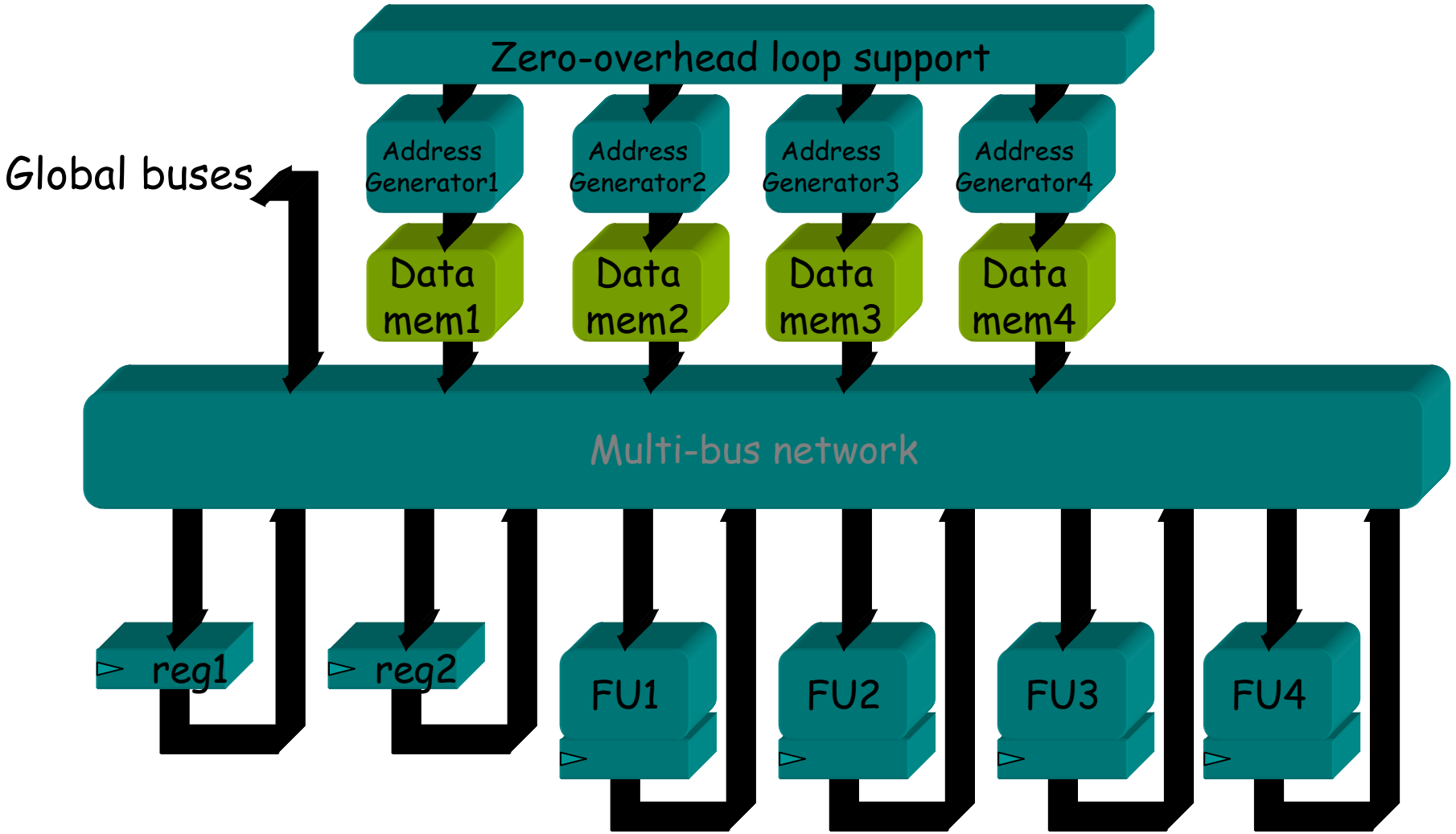
3- QUELQUES EXEMPLES

DART architecture (R2D2)



3- QUELQUES EXEMPLES

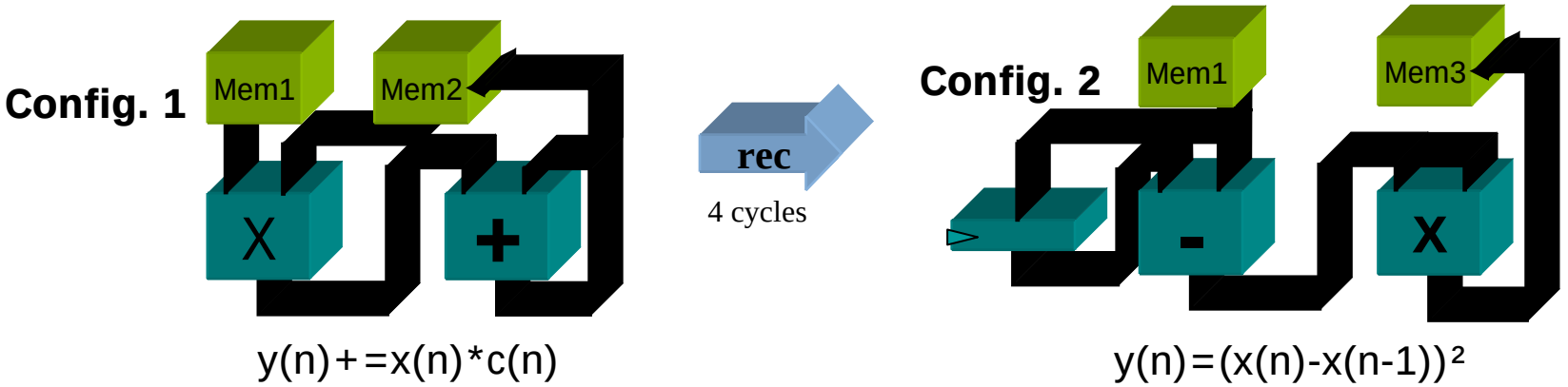
DART architecture (IRISA)



3- QUELQUES EXEMPLES

DART architecture (IRISA)

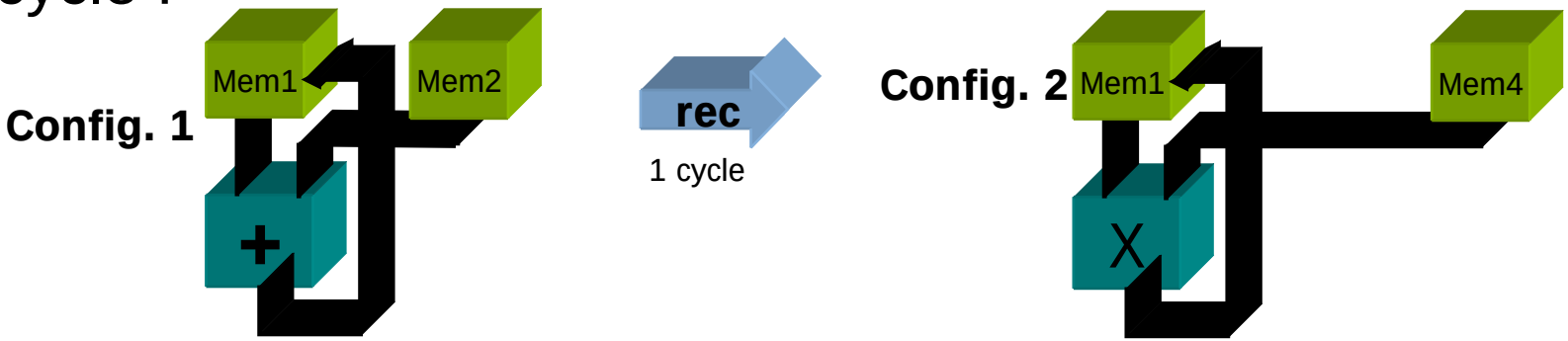
- HW reconfiguration pour optimiser le datapath:



y(n)+=x(n)*c(n)

y(n)=(x(n)-x(n-1))²

- SW reconfiguration reconfigure le datapath à chaque cycle :



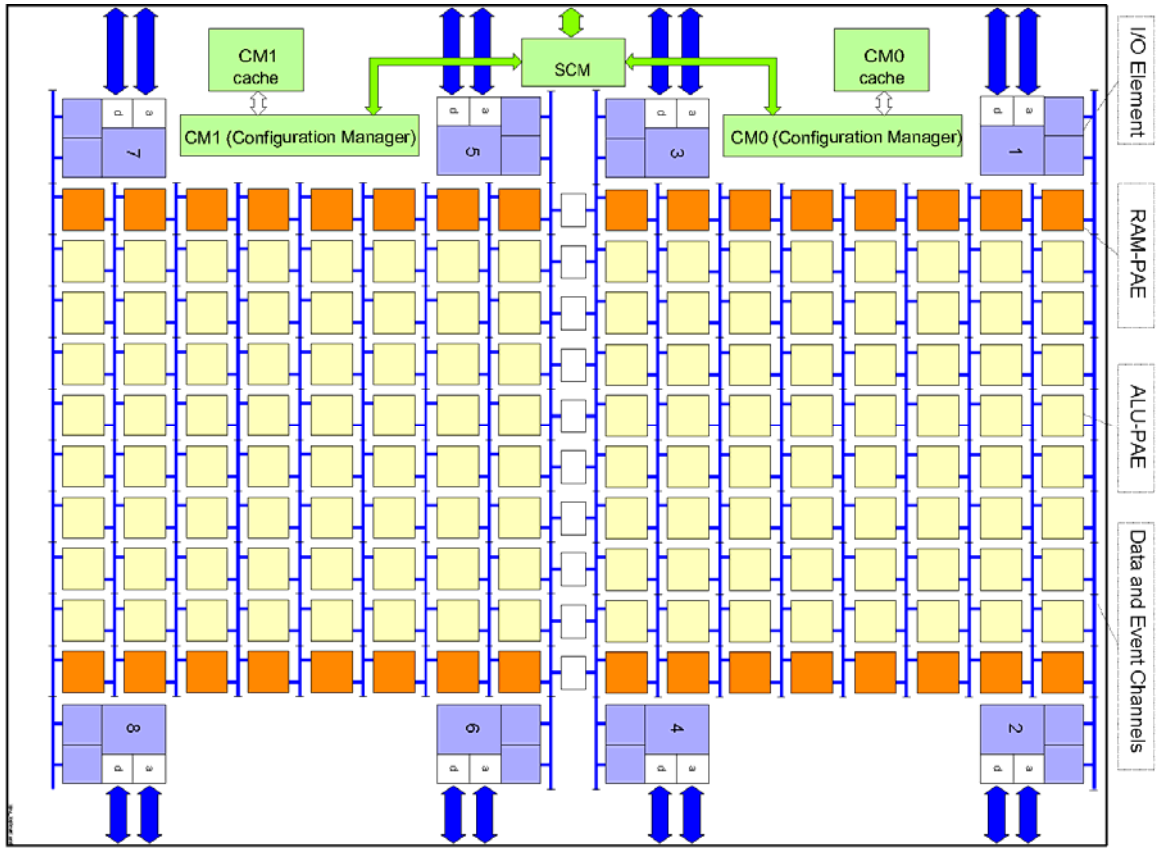
S=A+B

S=C*D

3- QUELQUES EXEMPLES

Source slides : http://pactcorp.com

PACT XPP (eXtrem Processor Platform)

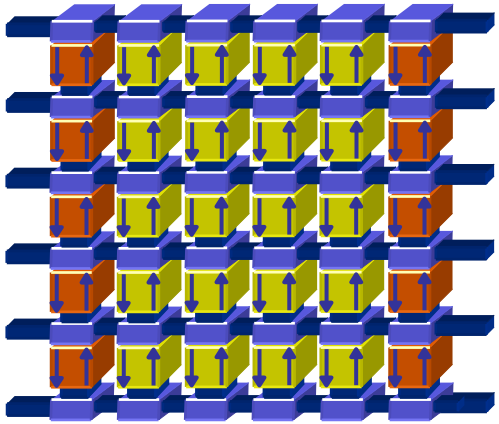


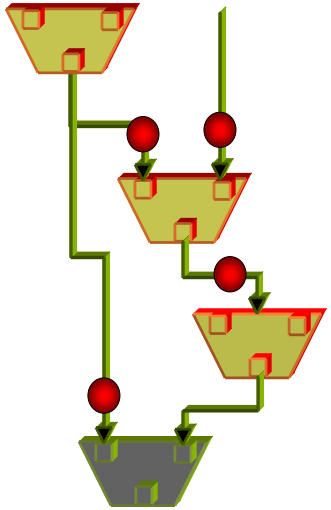
128 CFB@100 @100 MHz … 12.8 GMACs/s (32 bits fixed point)

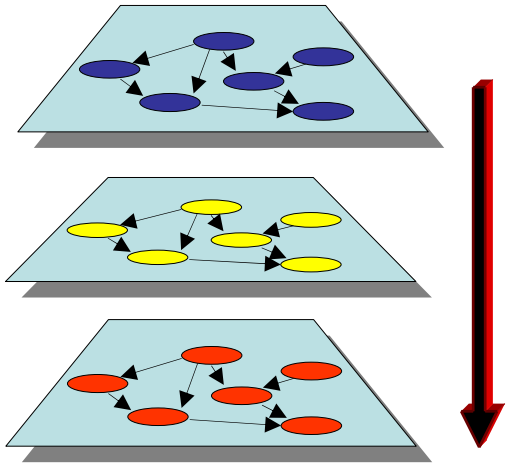
3- QUELQUES EXEMPLES

PACT XPP (eXtrem Processor Platform)

- **multiple parallel processing elements**
- **configuration flow replaces instruction flow**





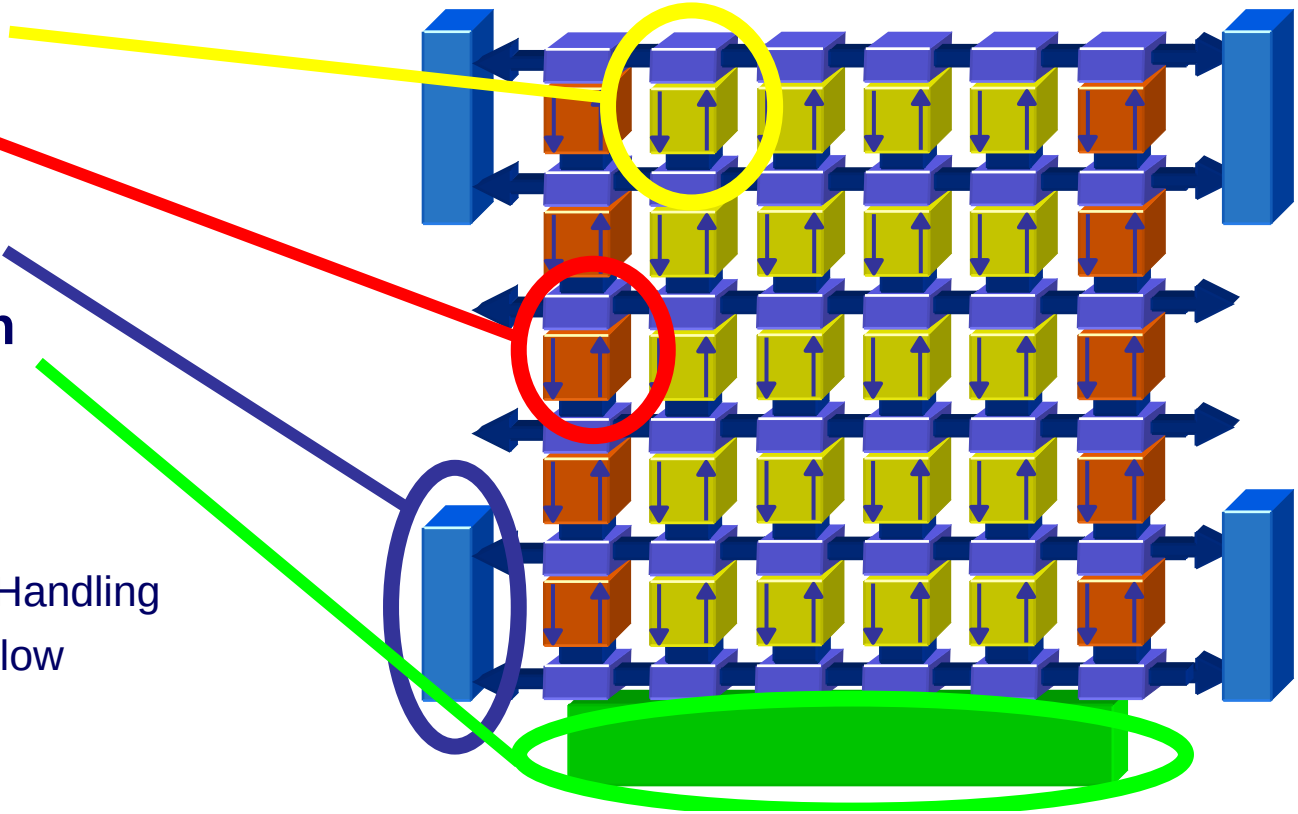


3- QUELQUES EXEMPLES

PACT XPP (eXtrem Processor Platform)

XPUs are eXtreme Processing Units built from a small number of standardized Processing Array Elements (PAEs)

- **ALU - PAE**
- **RAM - PAE**
- **I/O-Element**
- **Configuration Manager**

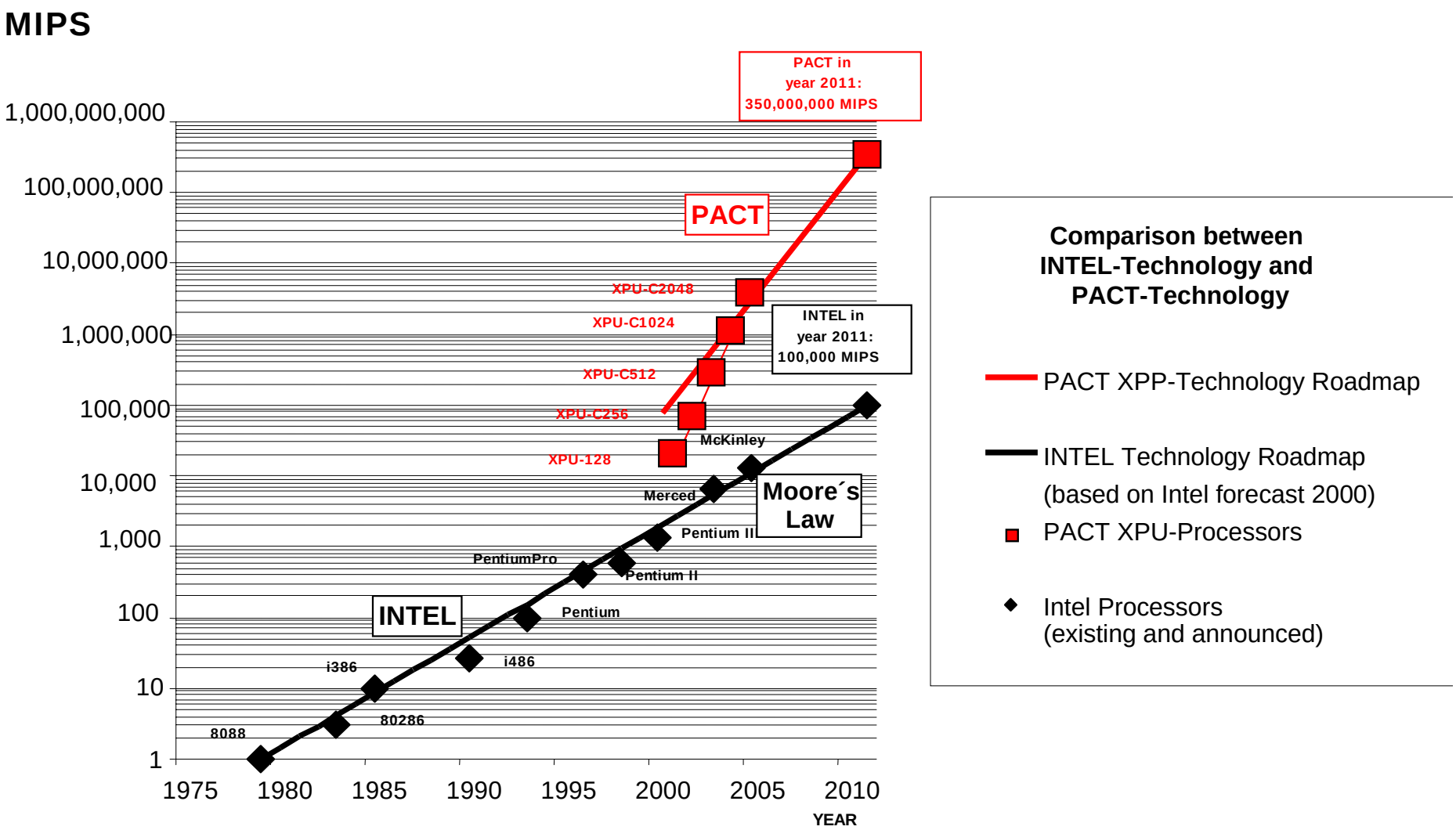


All PAEs include:

+ Reconfiguration Handling
+ Automatic DataFlow Synchronization
+ Event Network

3- QUELQUES EXEMPLES

PACT XPP (eXtrem Processor Platform)



3- QUELQUES EXEMPLES

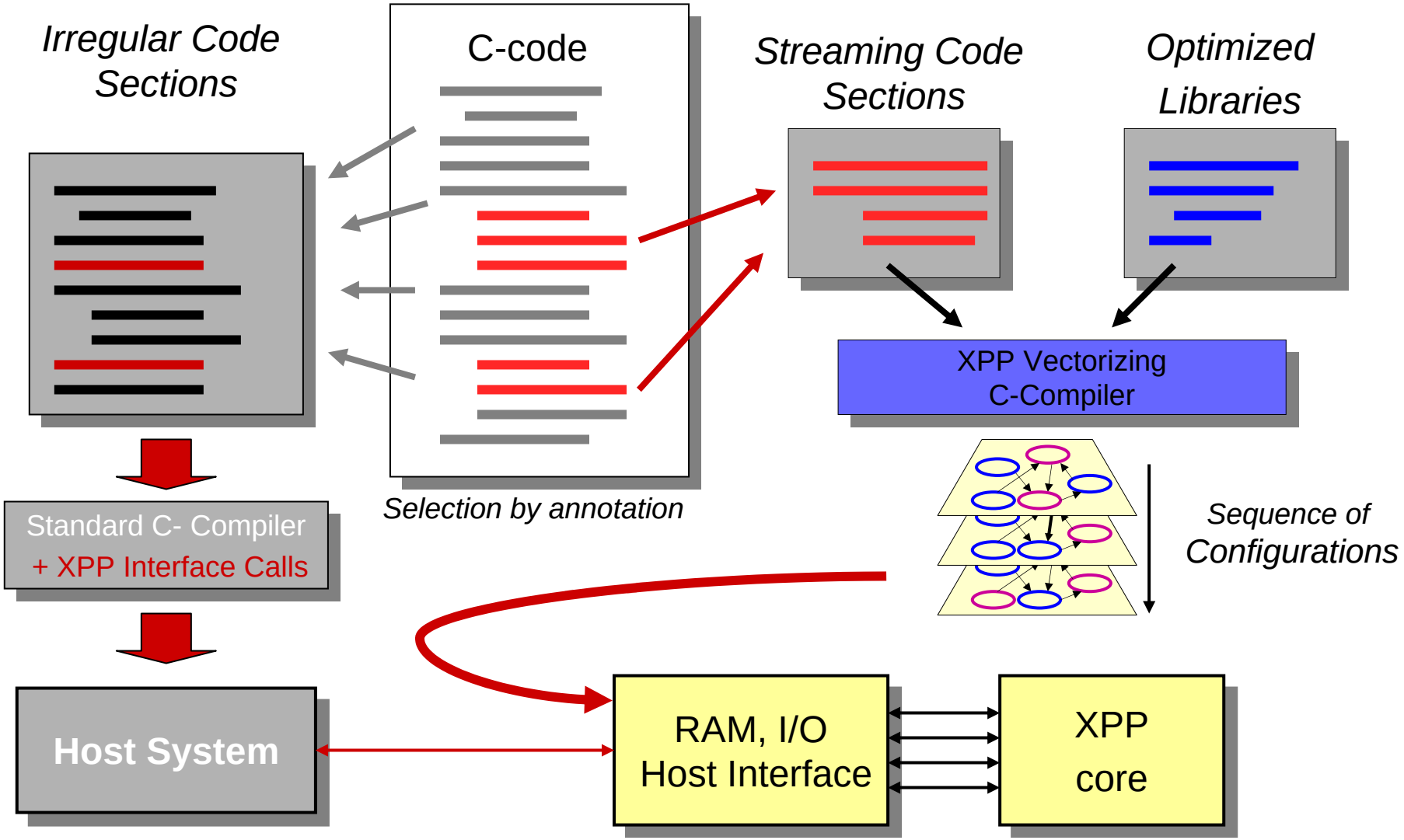
PACT XPP (eXtrem Processor Platform) : compilation

Strategies

- XPP coprocessor or streaming code sections are selected by annotations in original source code (currently)
- Irregular code sections are compiled with standard compilers for the Host system + XPP system calls
- The XPP C-Compiler performs a data dependency analysis and vectorizes the code
- The sequence of configurations is loaded during runtime on Host demand
- Communication between Host and XPP by means of library functions

3- QUELQUES EXEMPLES

PACT XPP (eXtrem Processor Platform) : compilation

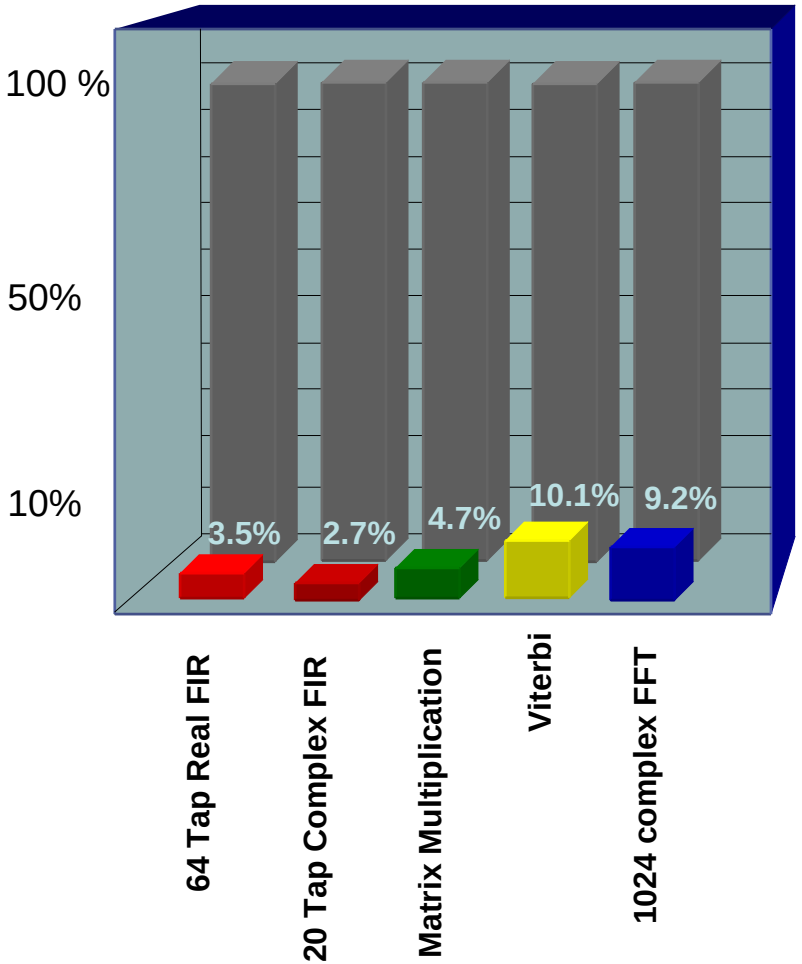


3- QUELQUES EXEMPLES

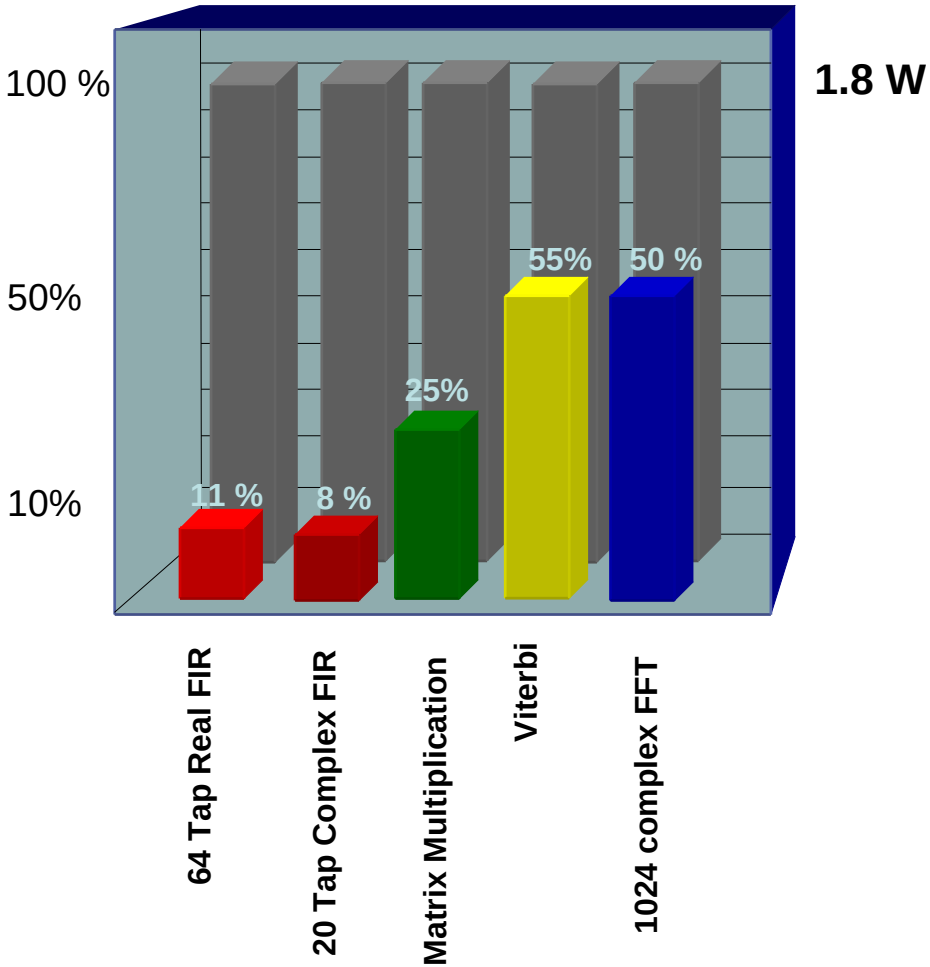
PACT XPP (eXtrem Processor Platform)

Comparison: Texas Instruments C6203 vs. XPU128

Clock Cycles



Normalized Power



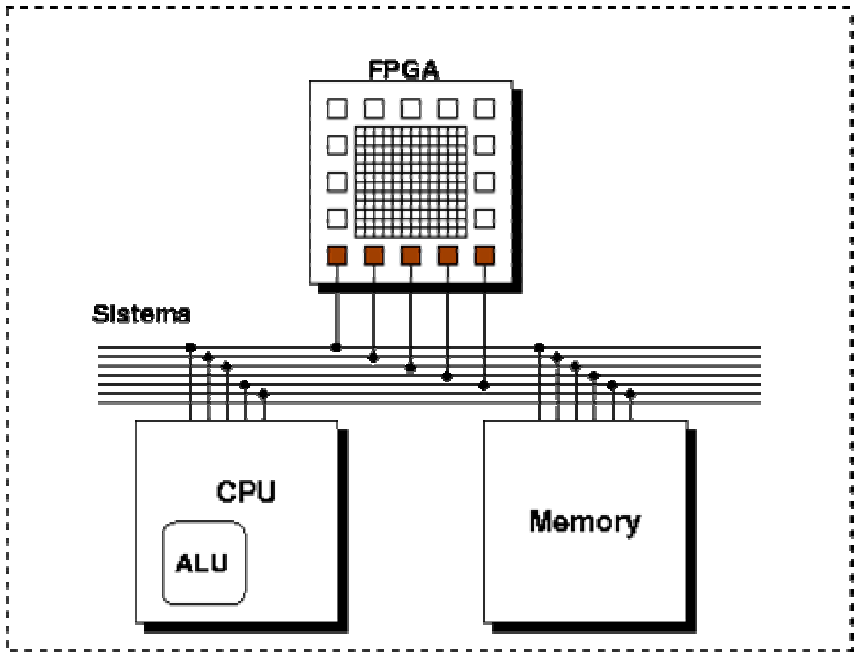
3- QUELQUES EXEMPLES

Parallelisme au niveau instruction: XiRISC

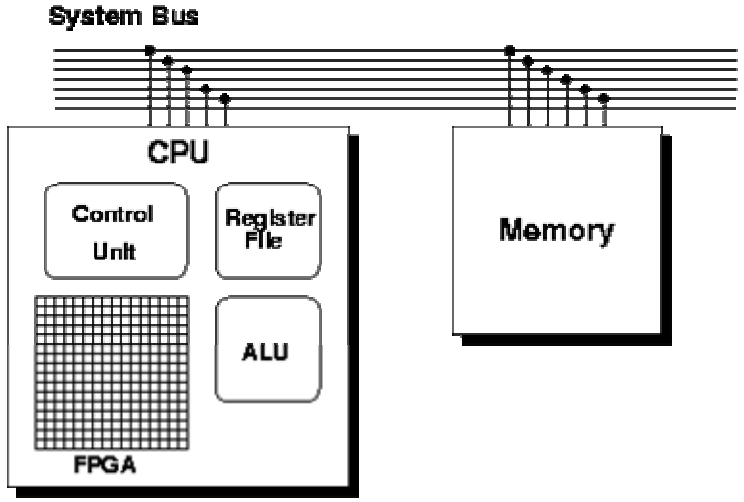
L'accélérateur reconfigurable est integraté dans le pipeline du CPU

http://xirisc.deis.unibo.it

1: *Coprocessor model*



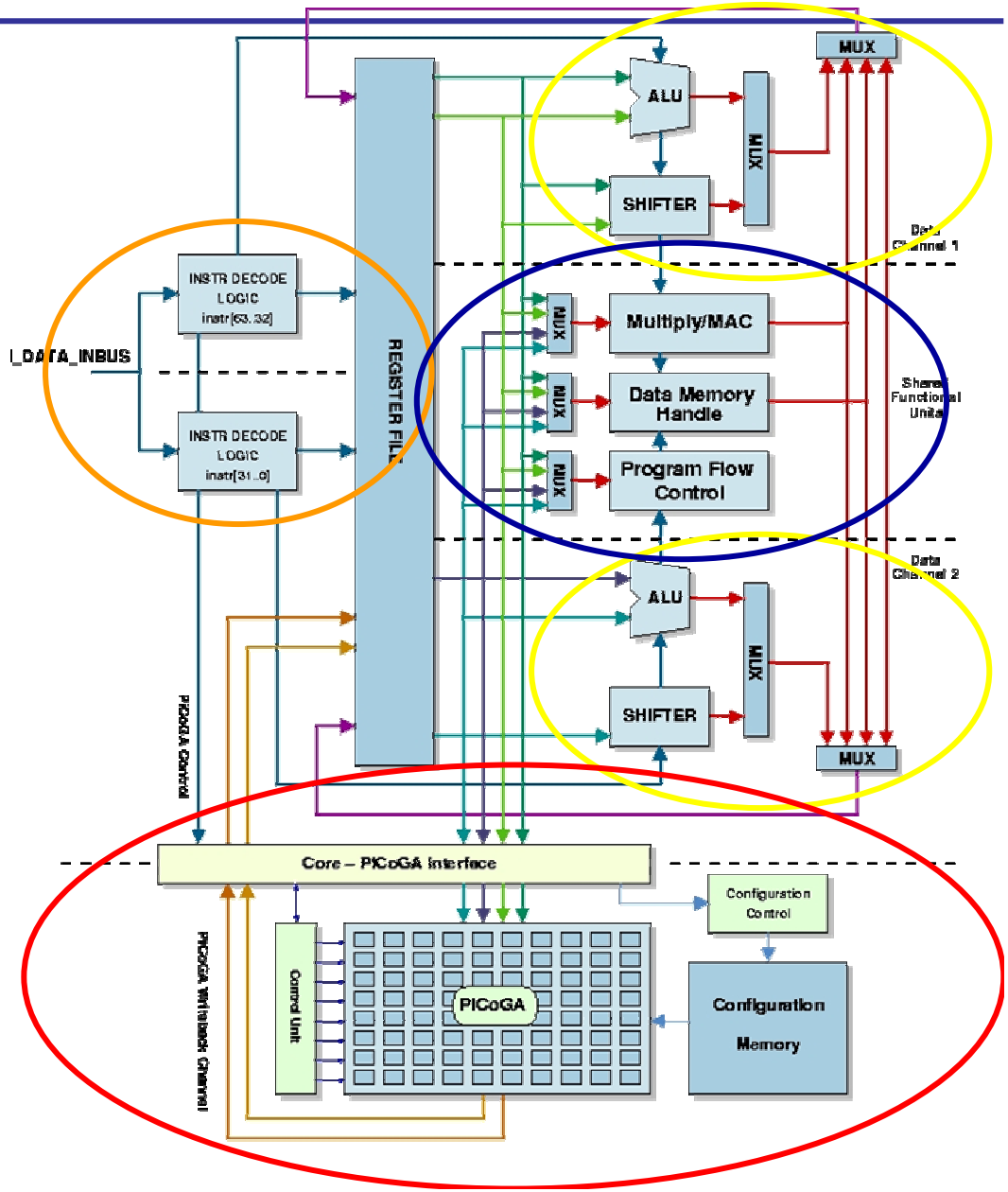
2: *Function unit model*



3- QUELQUES EXEMPLES

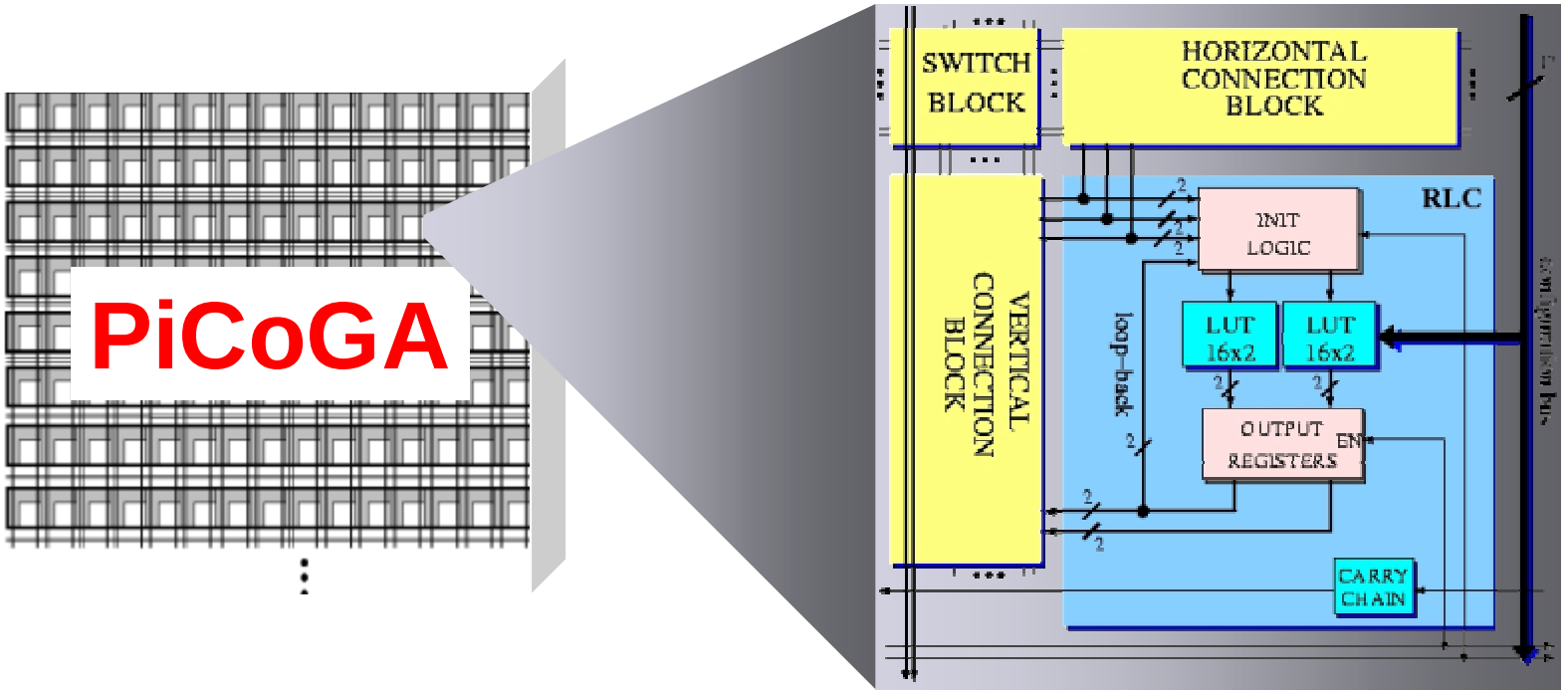
XiRisc

- decodage d'instruction dupliqué
- Unités Alu and Shifter dupliqués
- Fonctions partagées DSP operations,Memory handler
- **A pipelined configurable Gate Array**



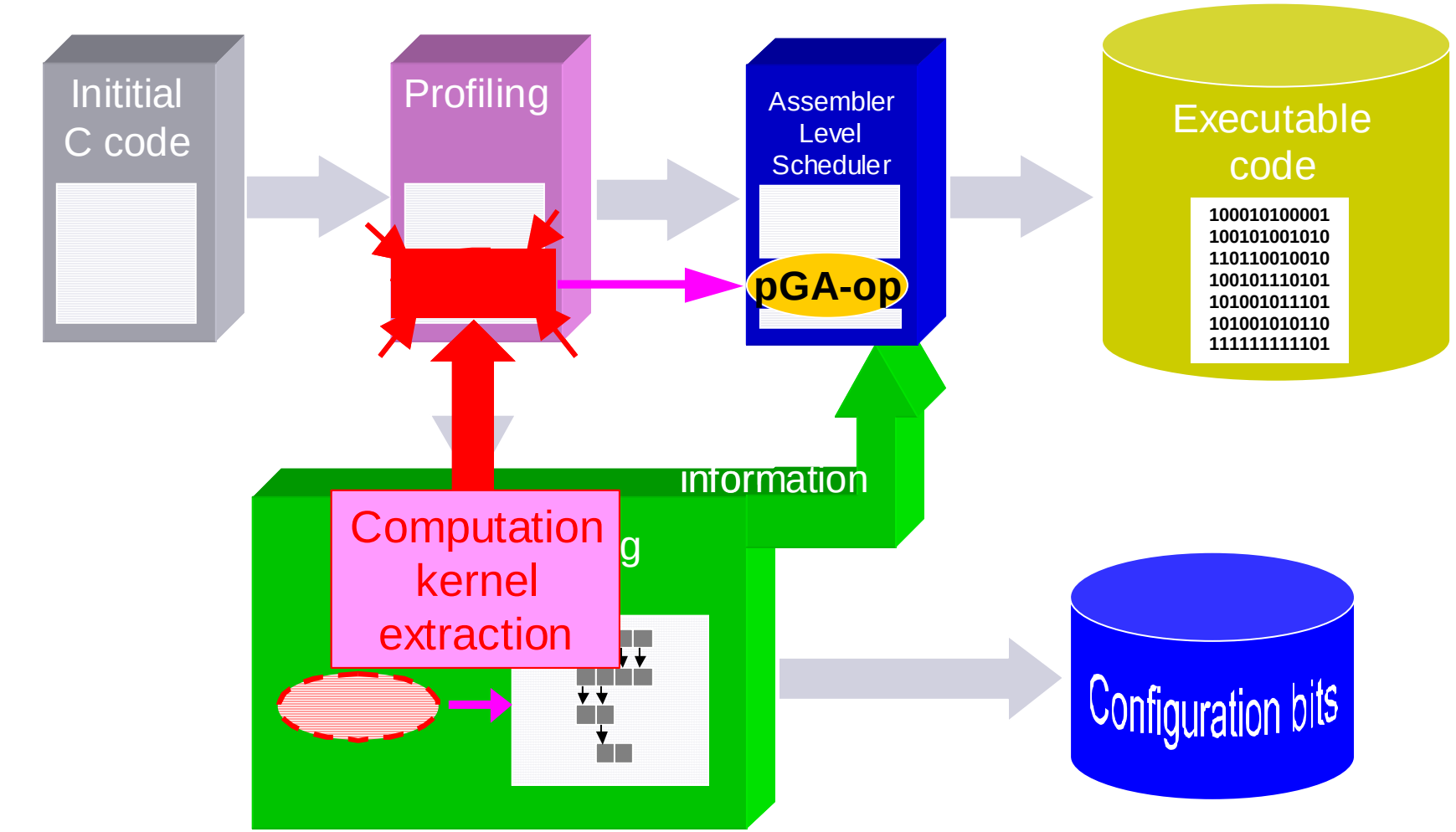
3- QUELQUES EXEMPLES

XiRisc



- 2D-LUT-based architecture
- Exécution indépendante ou concourrente avec le processeur
- 4x32-bit données en entrée, 2x32-bit en sortie

3- QUELQUES EXEMPLES



3- QUELQUES EXEMPLES

Utilisation de pragma dans le Code C

```
#pragma pGA shift_add 0x12  5  c a b
        c = ( a << 2 ) + b
#pragma end
```

```
/*************************************/
/*  Shift_add mapped on PiCoGA */
/*************************************/

#if defined(PiCoGA)
...
asm("pGA-op 0x12 ...")
...
```

```
/*************************************/
/* Emulation function _shift_add */
/*************************************/

#else
void _shift_add(){
...
c = ( a << 2 ) + b
...
}
#endif
```

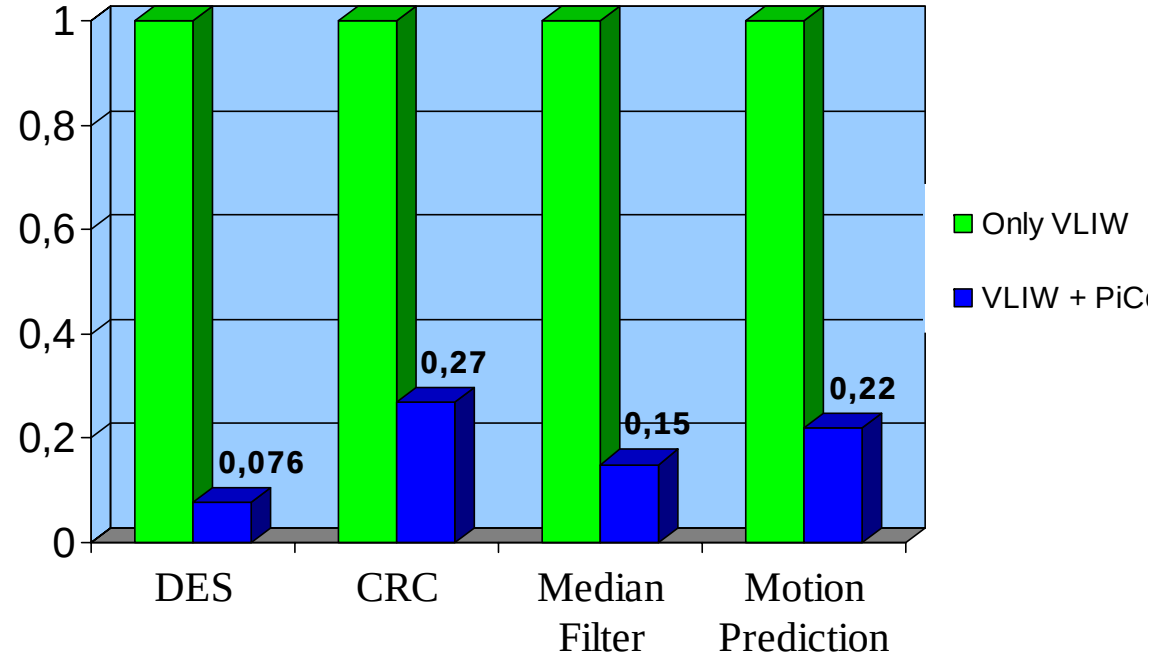
3- QUELQUES EXEMPLES

Exemple d'implémentation

DES	CRC	Median Filter	Motion Estimation	Motion Prediction	Turbo Codes
13.5x	**4.3x**	**7.7x**	**12.4x**	**4.5x**	**12x**

Réduction des accès mémoire

75% de la conso dans un VLIW est due à l'accès aux instructions et données



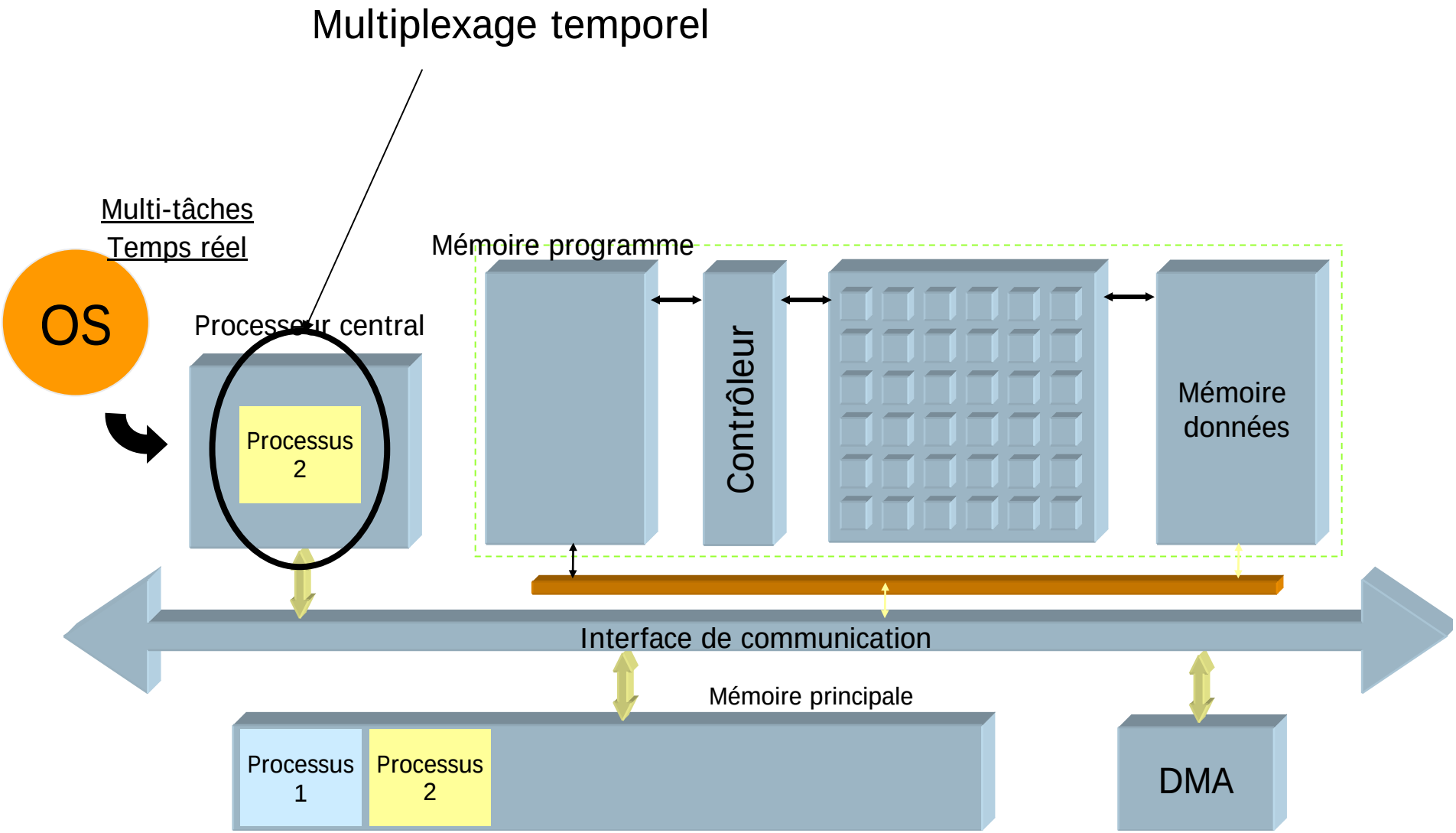
Sommaire

Partie 2 : Les architectures reconfigurables dynamiquement

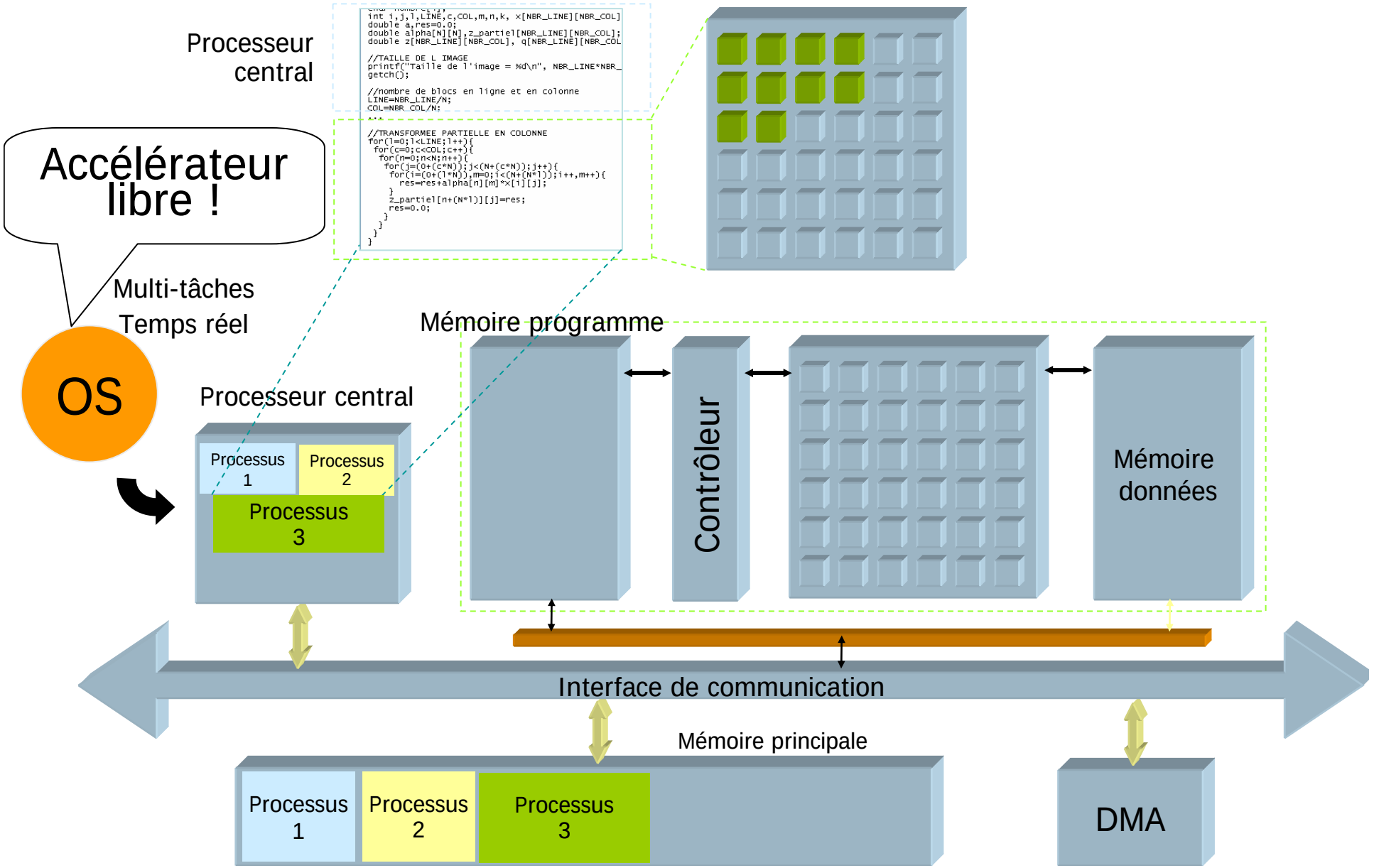
1- INTRODUCTION & CONTEXTE
2- CLASSIFICATION D'ARCHITECTURES
3- QUELQUES EXEMPLES
4- EXEMPLE DE GESTION DYNAMIQUE
5- CONCLUSIONS & PERSPECTIVES

4- EXEMPLE DE GESTION DYNAMIQUE

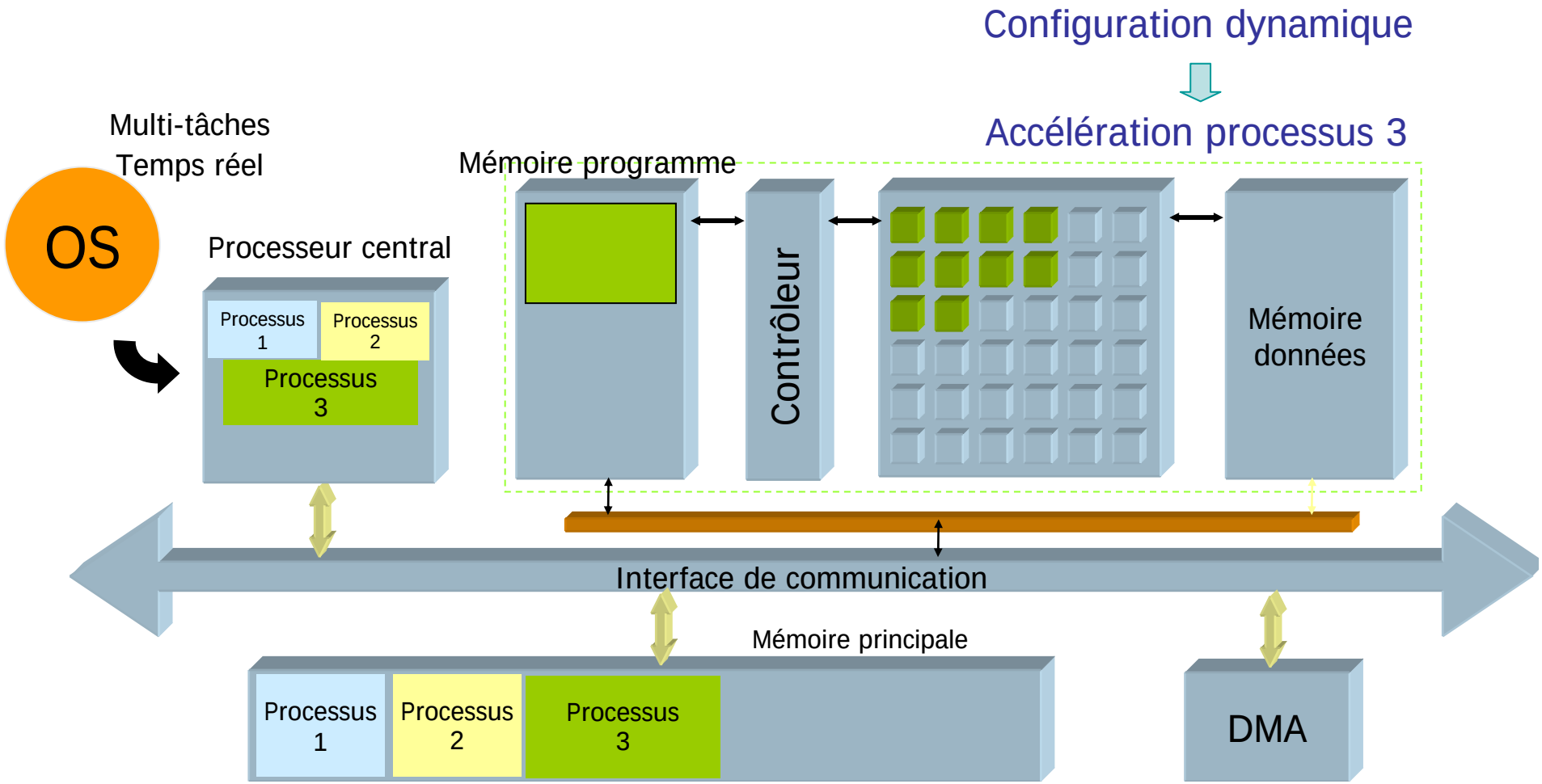
- ‘Zoom out’, réalisation au niveau système



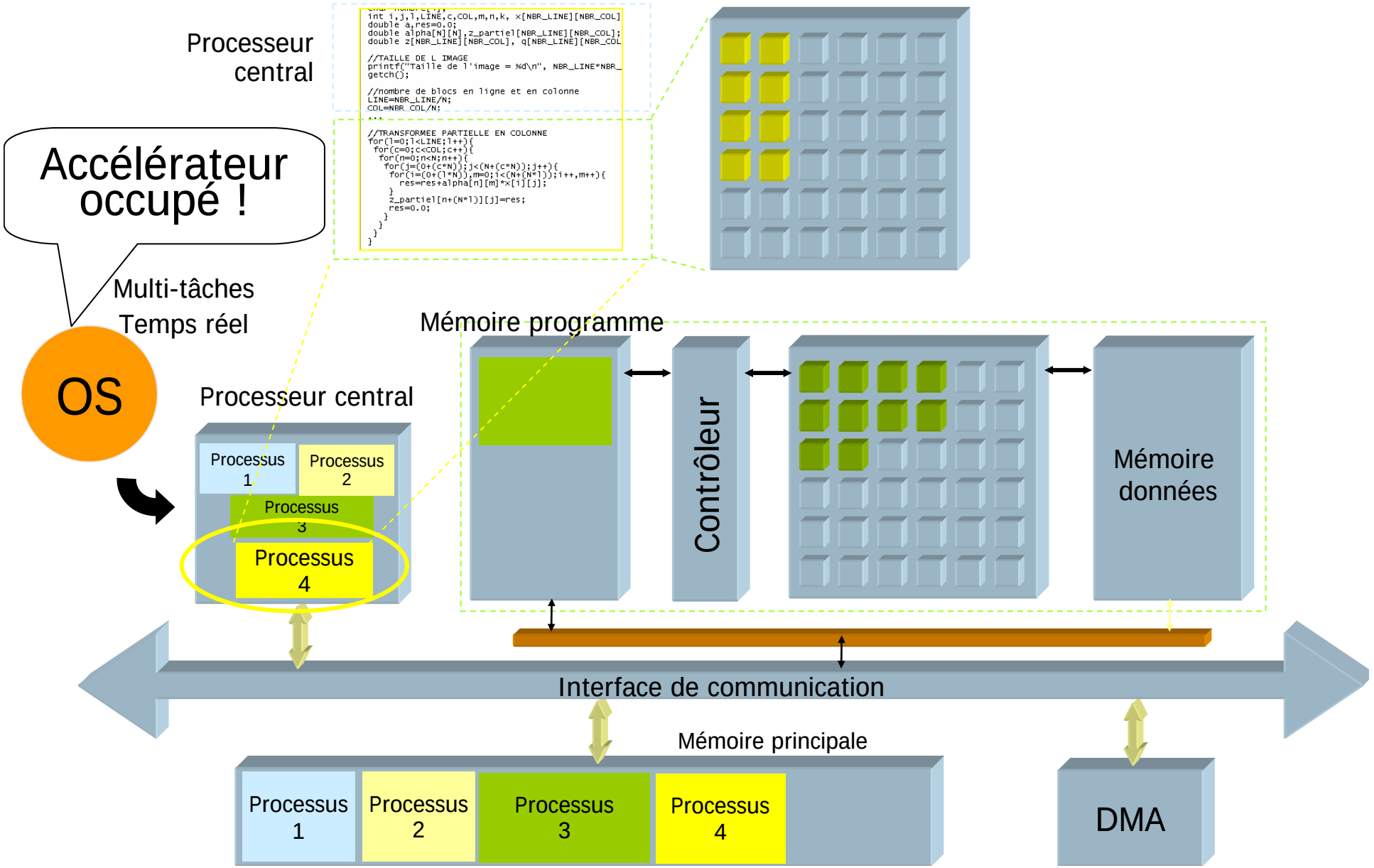
4- EXEMPLE DE GESTION DYNAMIQUE



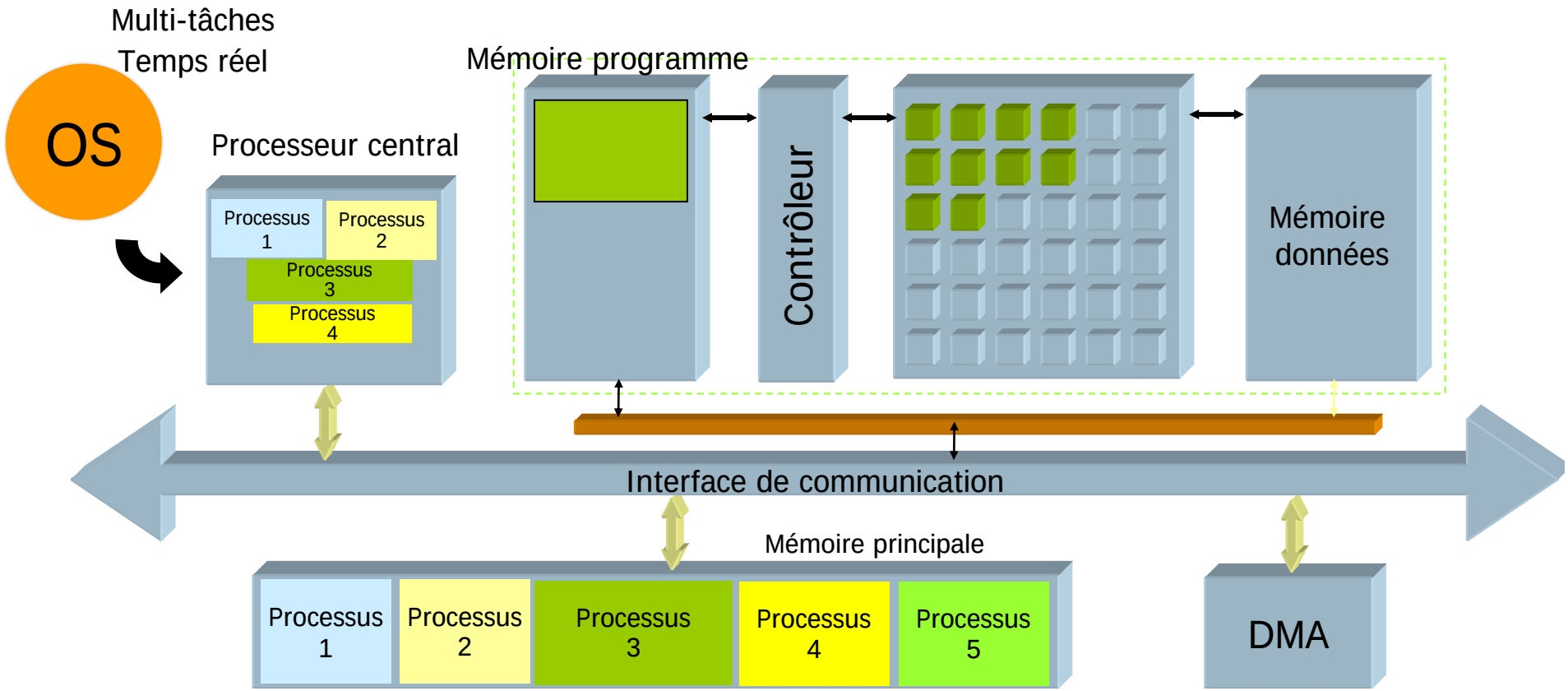
4- EXEMPLE DE GESTION DYNAMIQUE



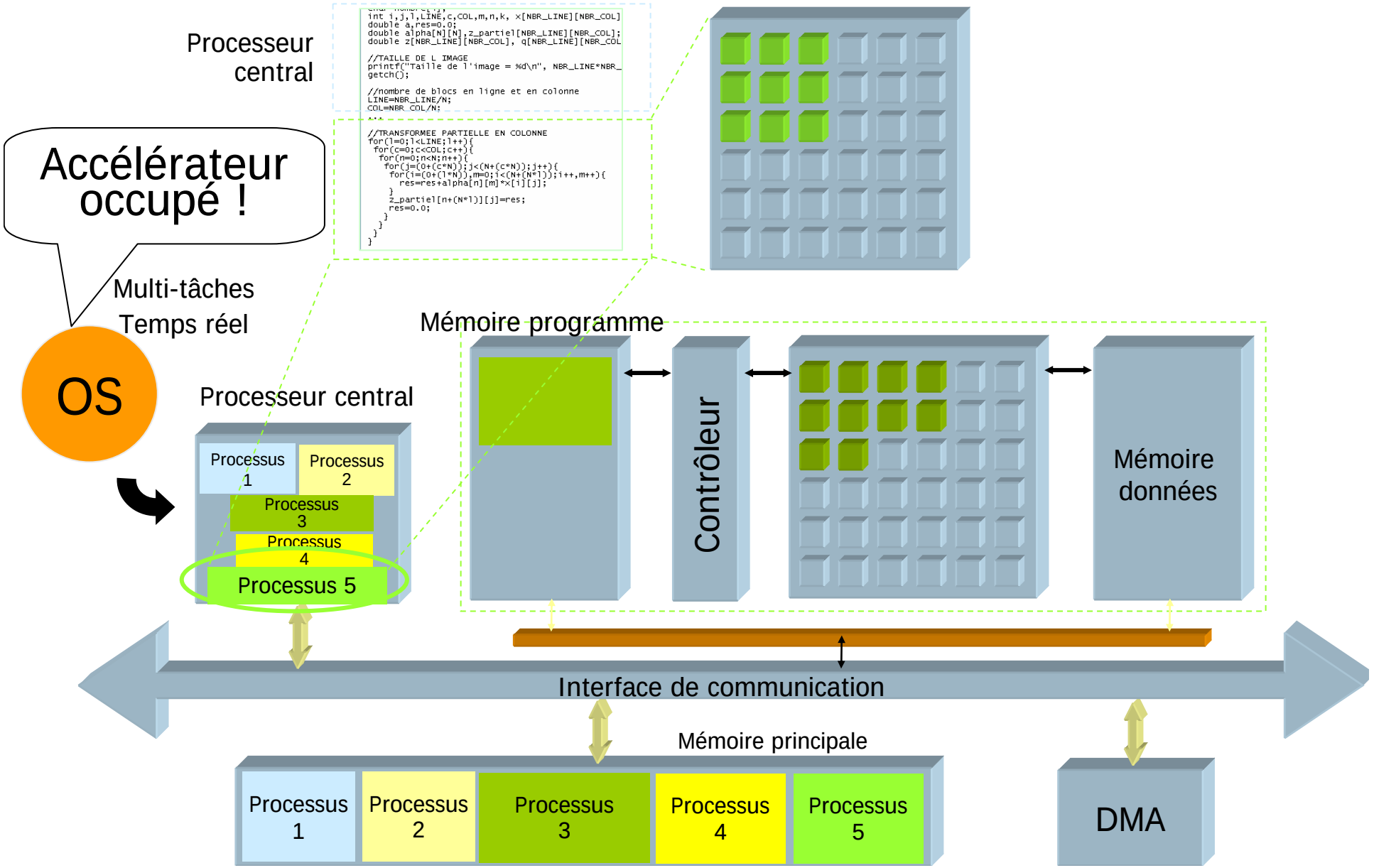
4- EXEMPLE DE GESTION DYNAMIQUE



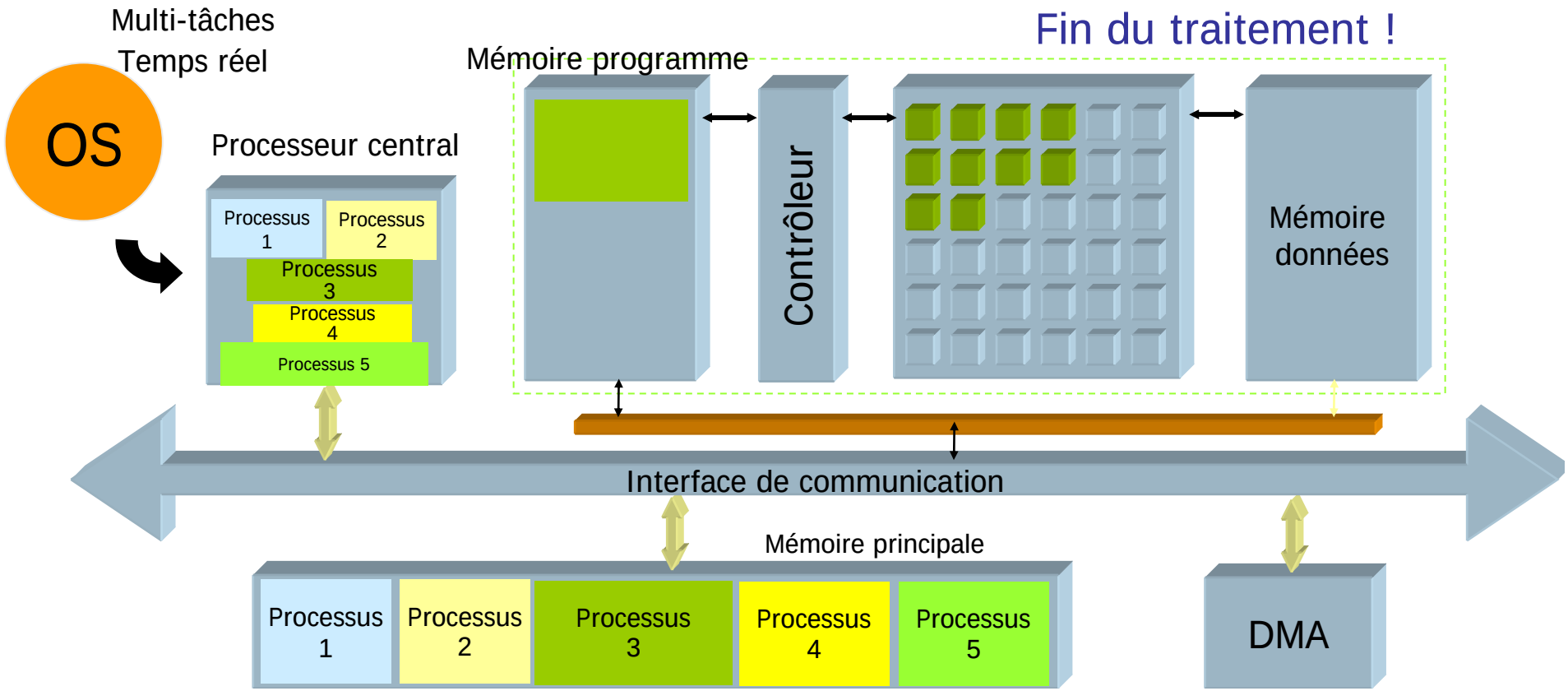
4- EXEMPLE DE GESTION DYNAMIQUE



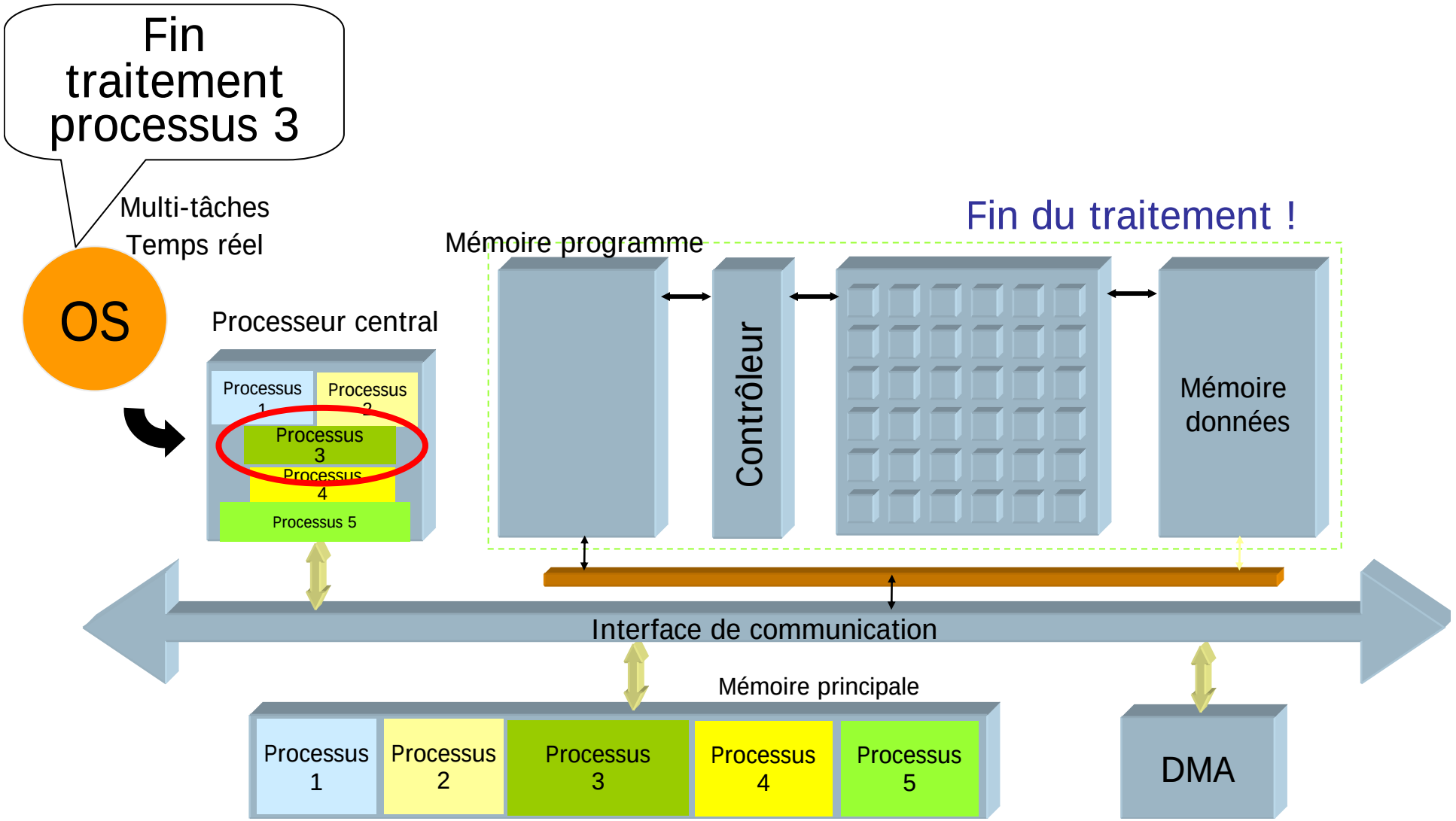
4- EXEMPLE DE GESTION DYNAMIQUE



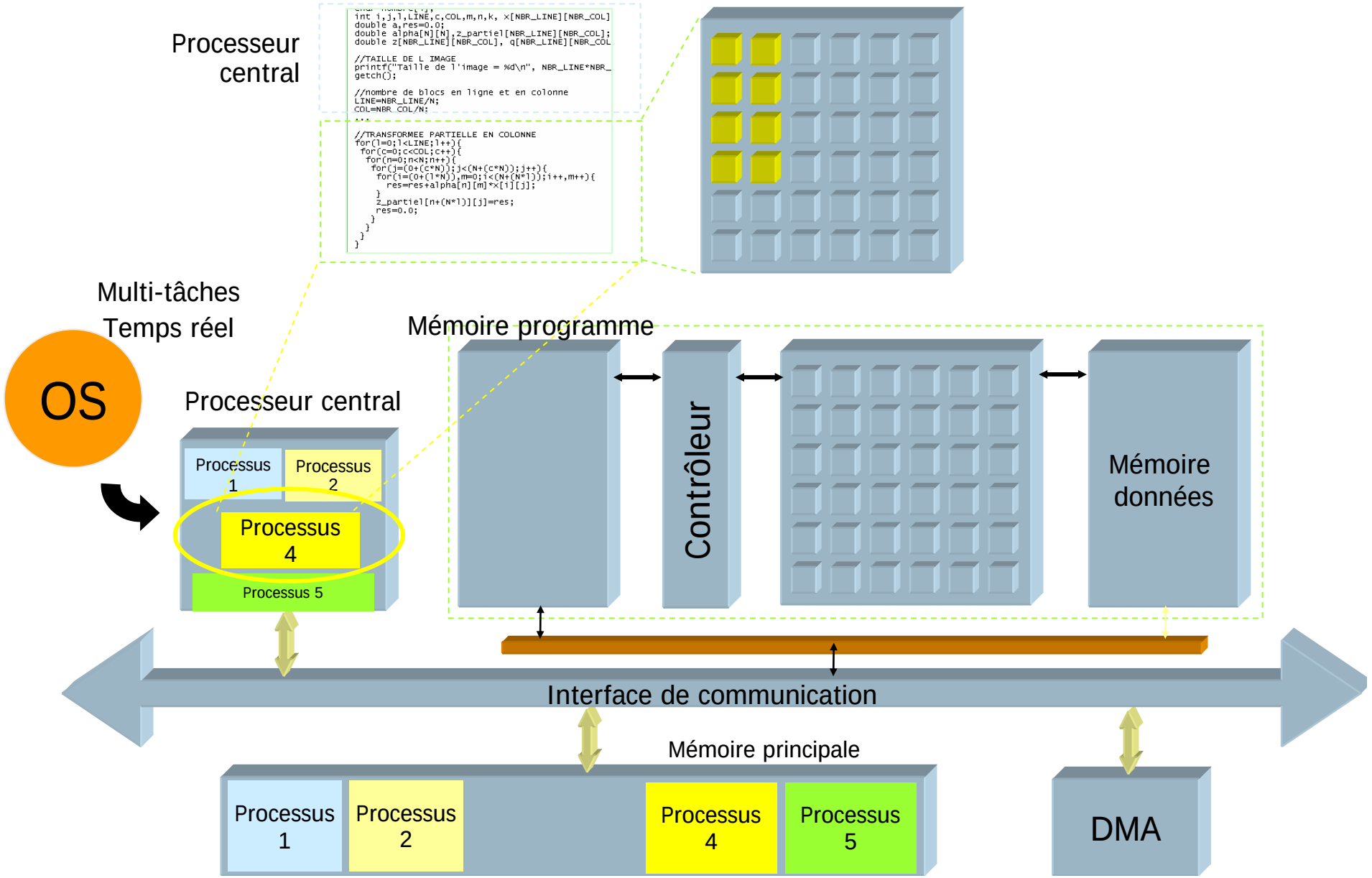
4- EXEMPLE DE GESTION DYNAMIQUE



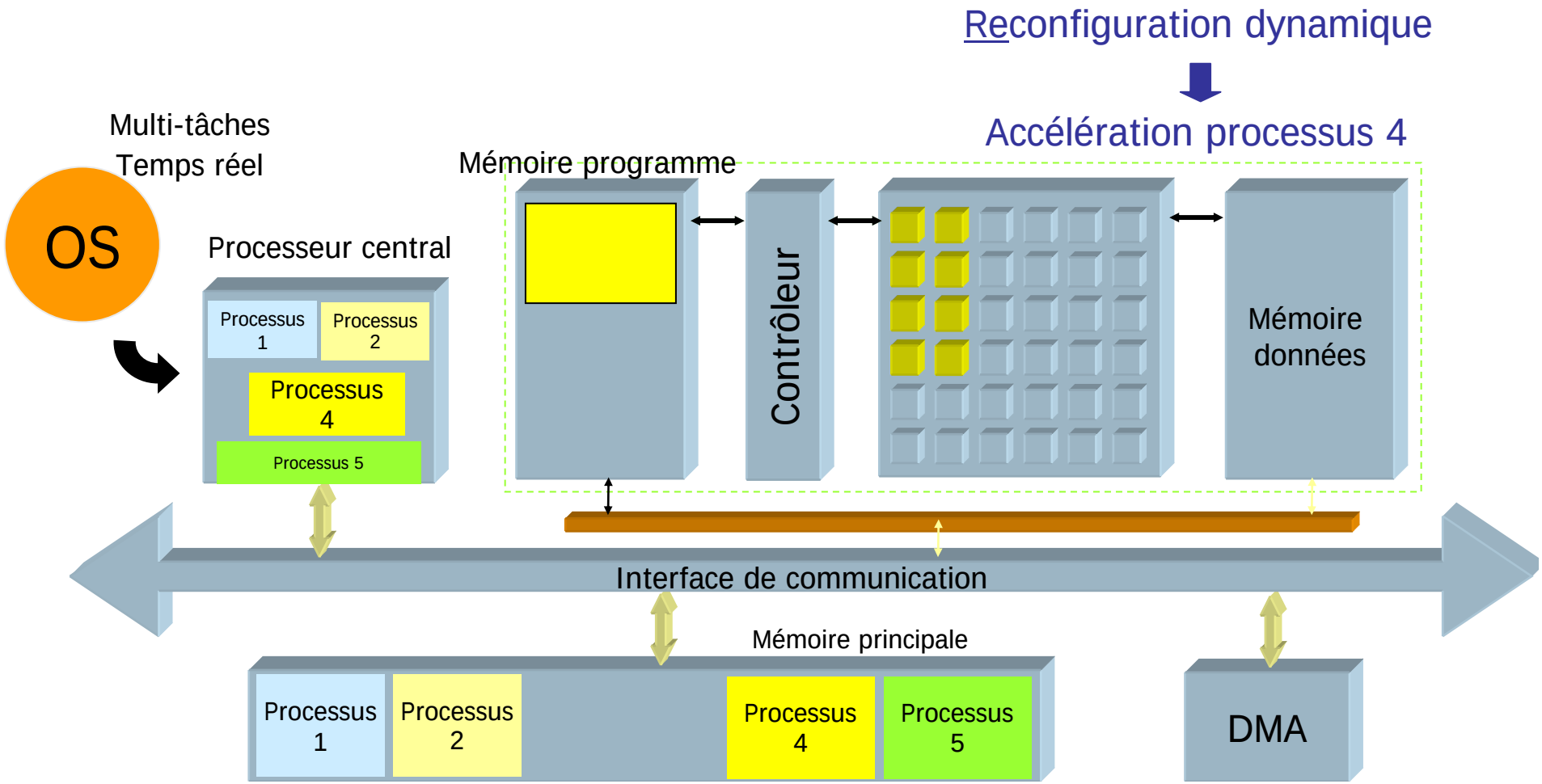
4- EXEMPLE DE GESTION DYNAMIQUE



4- EXEMPLE DE GESTION DYNAMIQUE

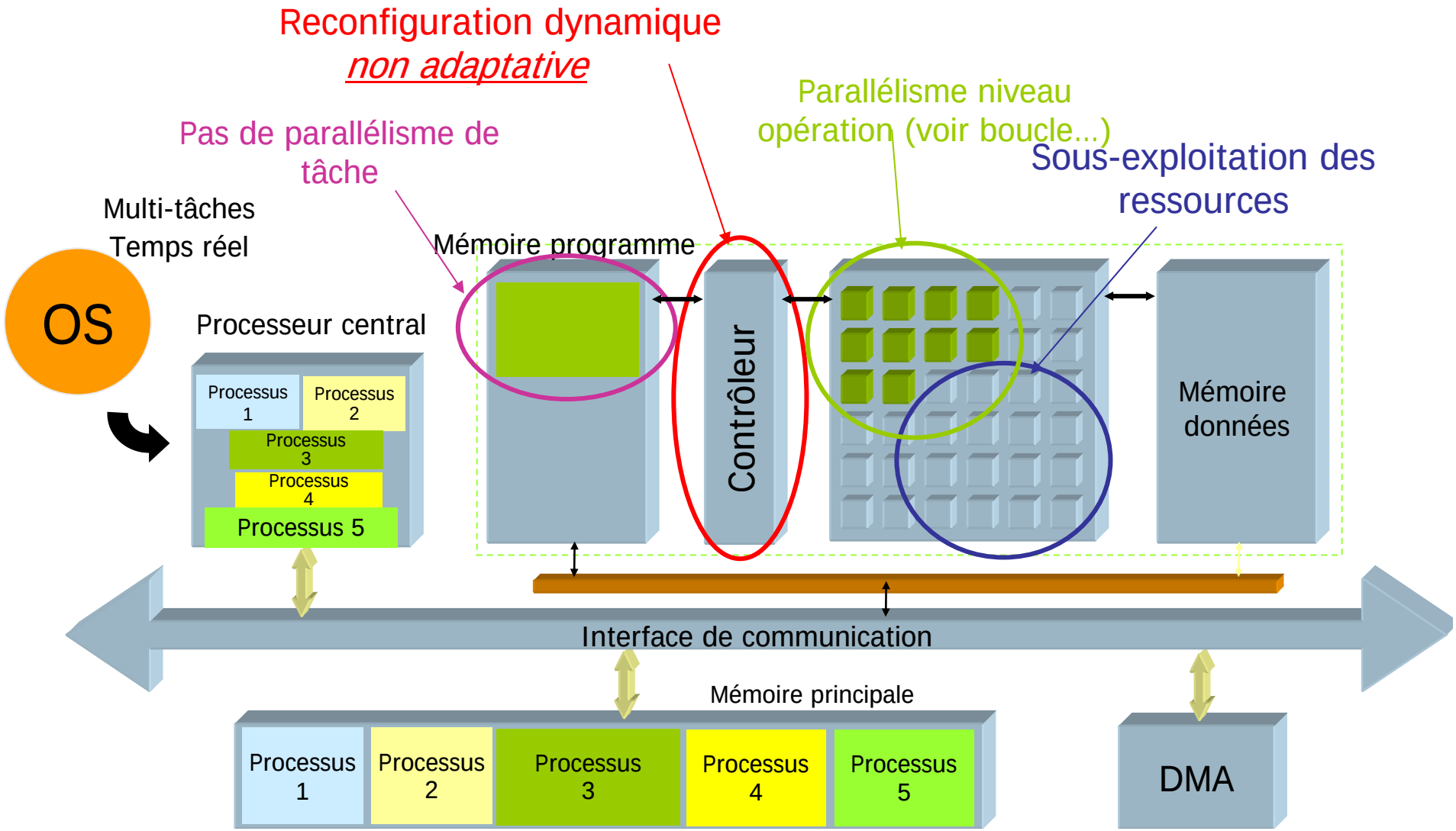


4- EXEMPLE DE GESTION DYNAMIQUE



4- EXEMPLE DE GESTION DYNAMIQUE

Bilan



4- EXEMPLE DE GESTION DYNAMIQUE

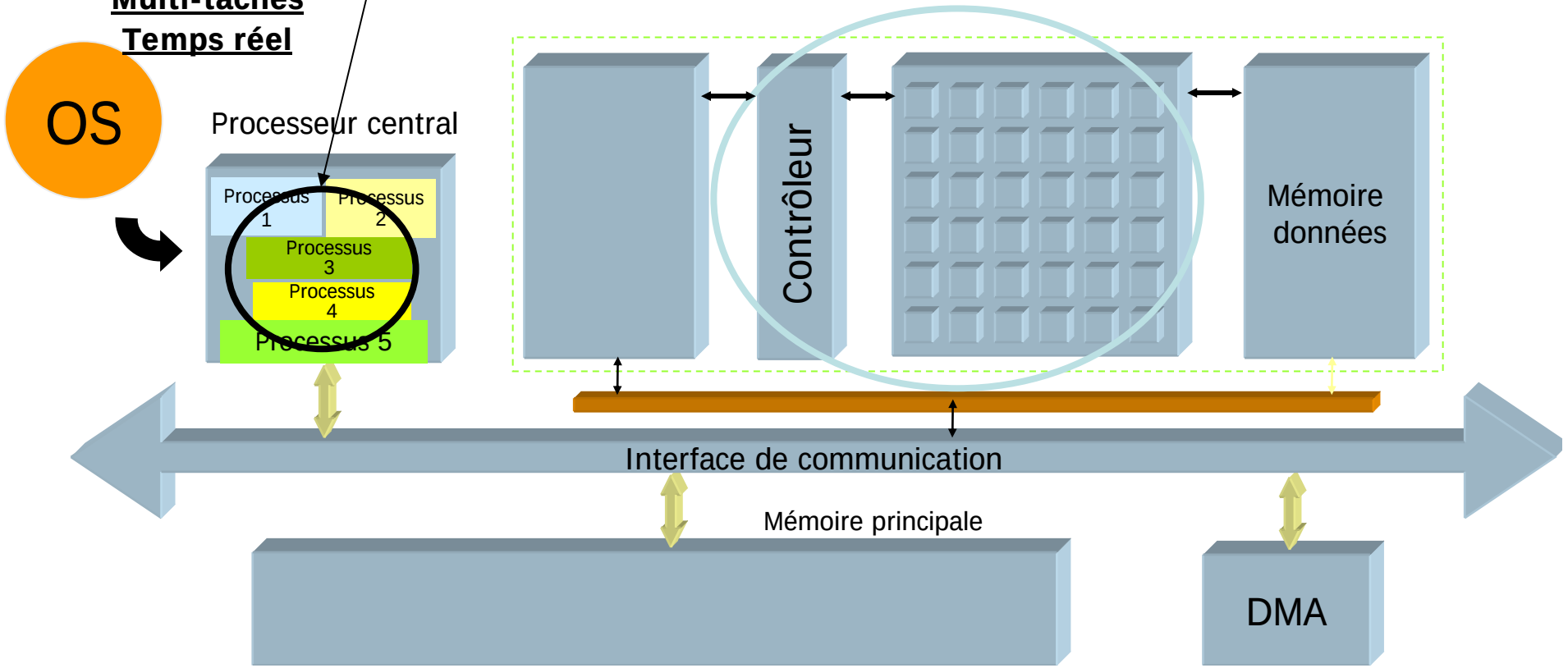
Architectures reconfigurables dynamiquement

Gestion dynamique des ressources !

Multiplexage temporel

Multi-tâches
Temps réel

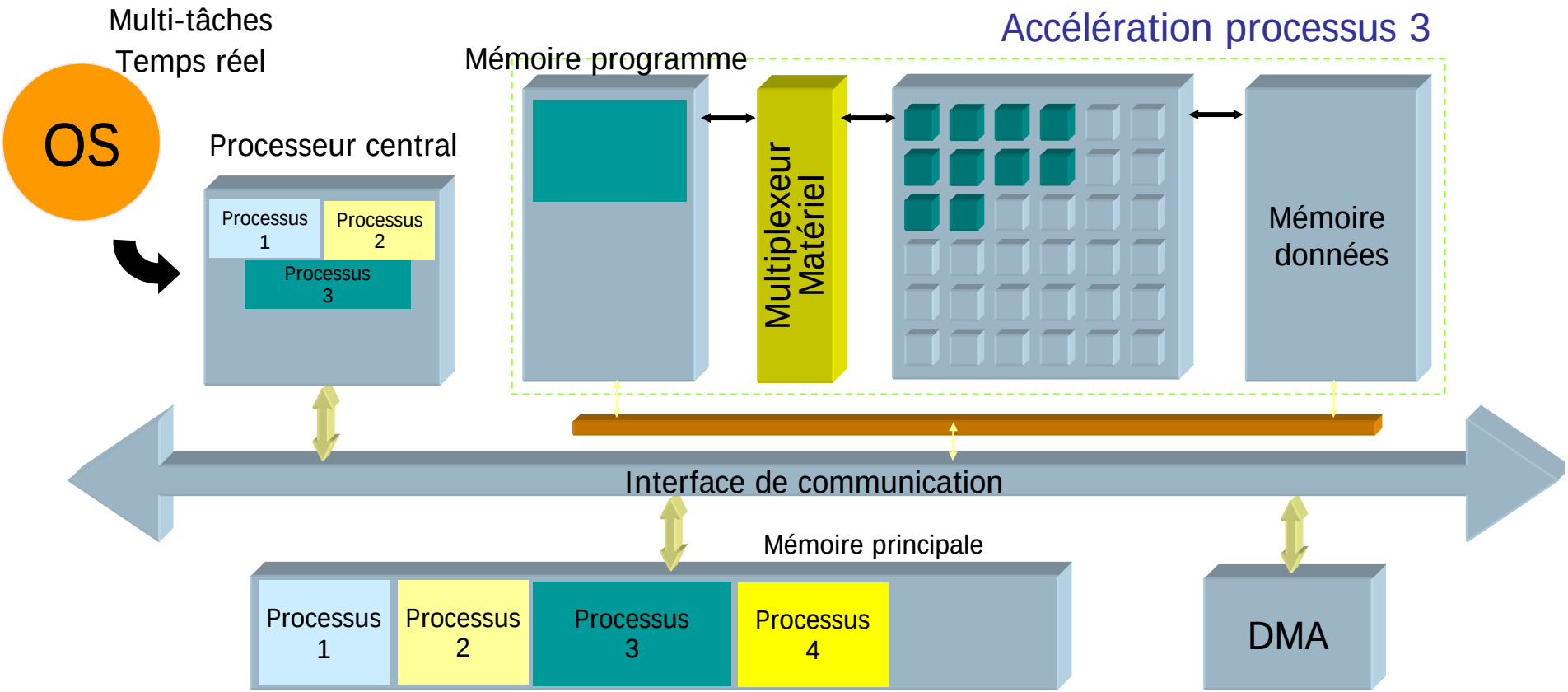
Méthode de multiplexage matériel



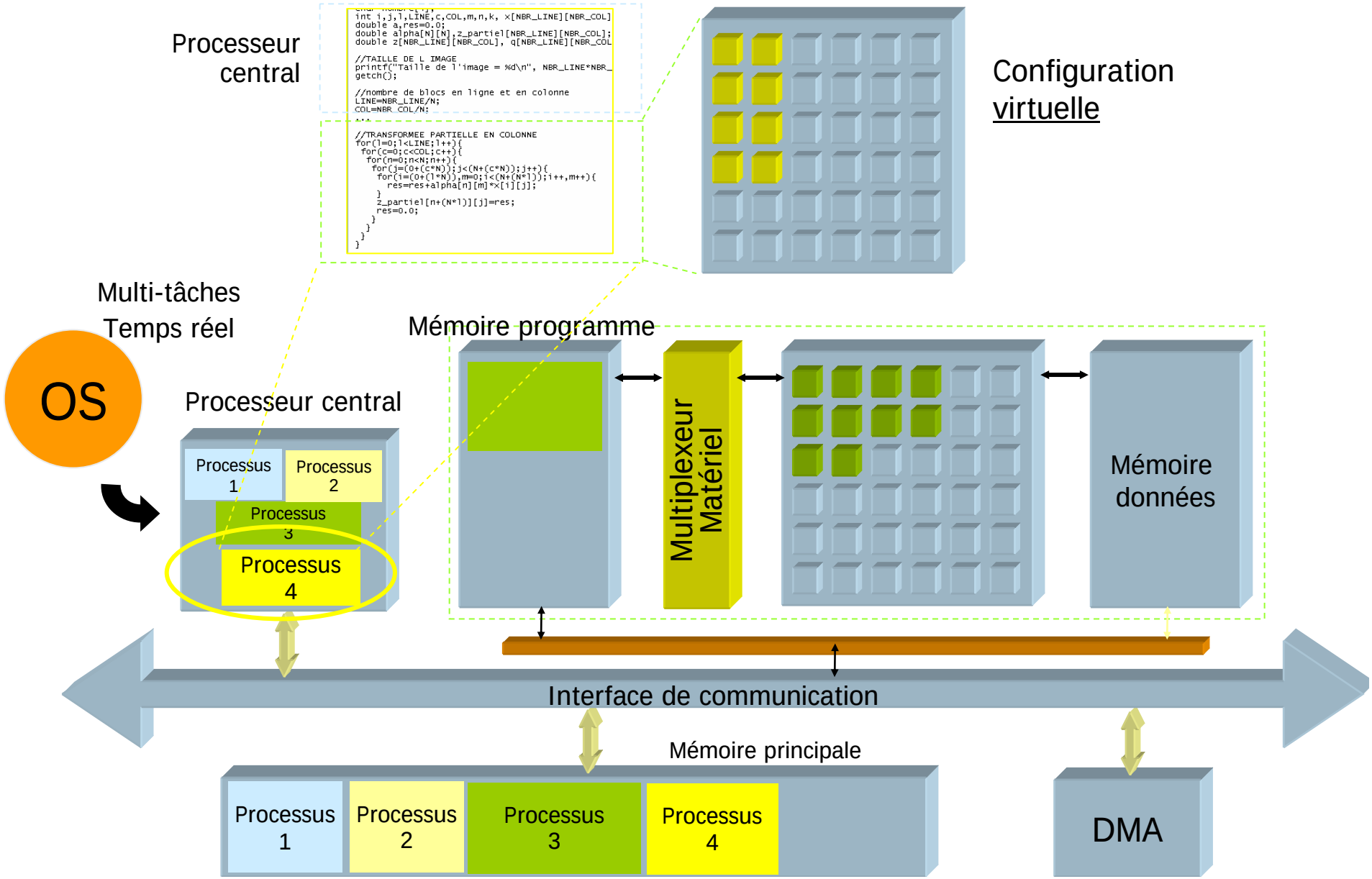
4- EXEMPLE DE GESTION DYNAMIQUE

Multiplexage matériel : *Saturne*

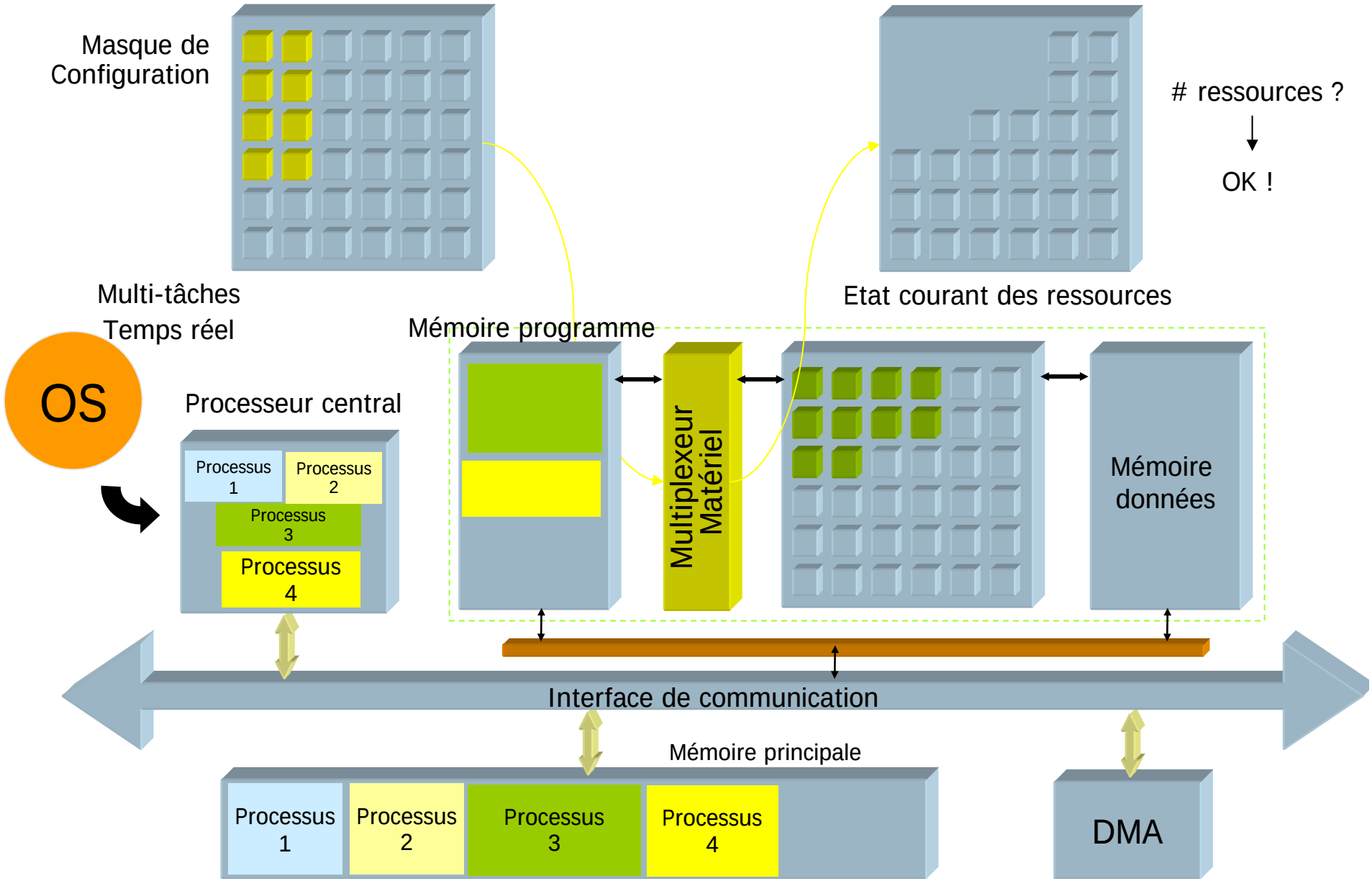
Parallélisme de Tâche



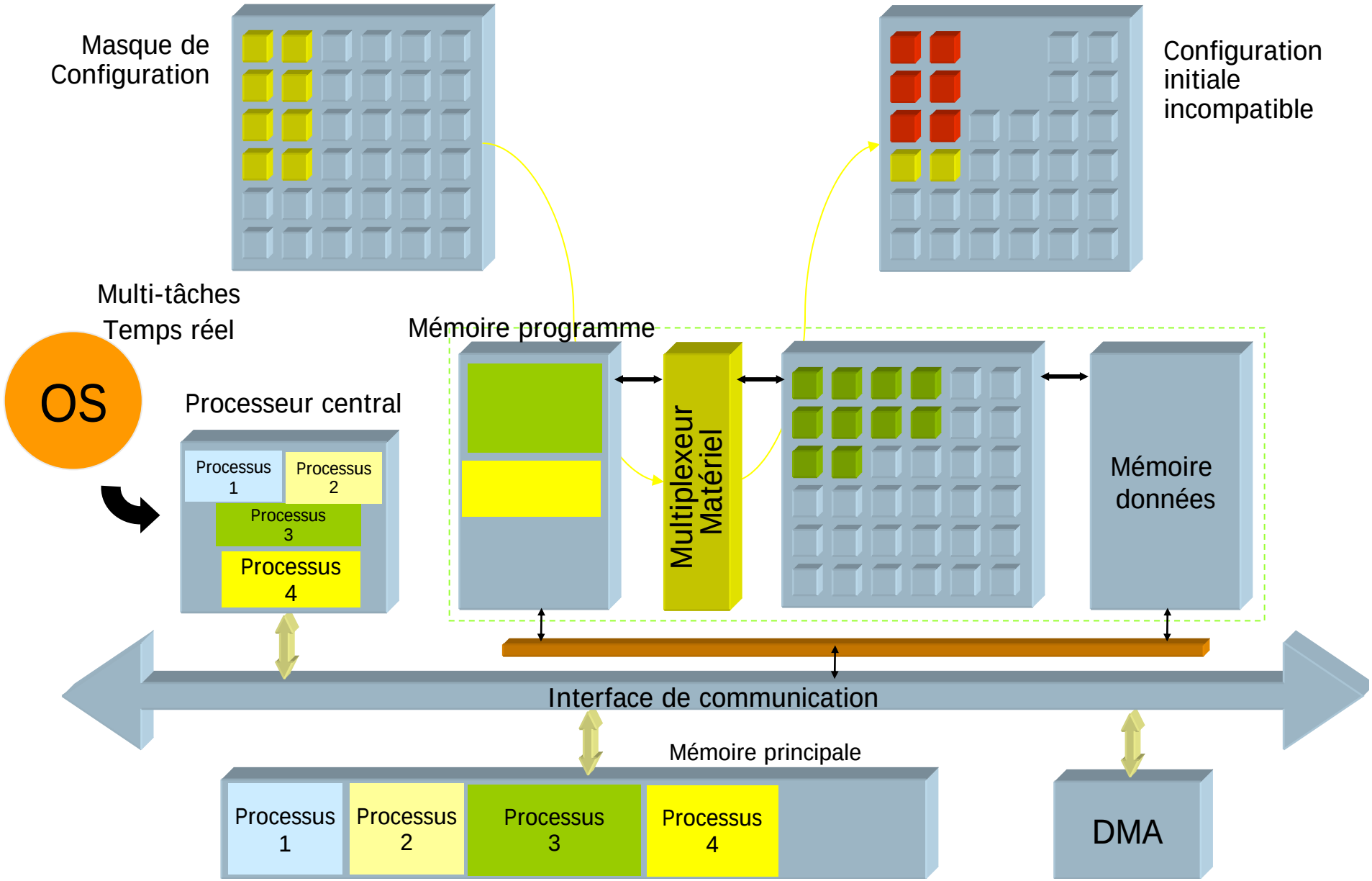
4- EXEMPLE DE GESTION DYNAMIQUE



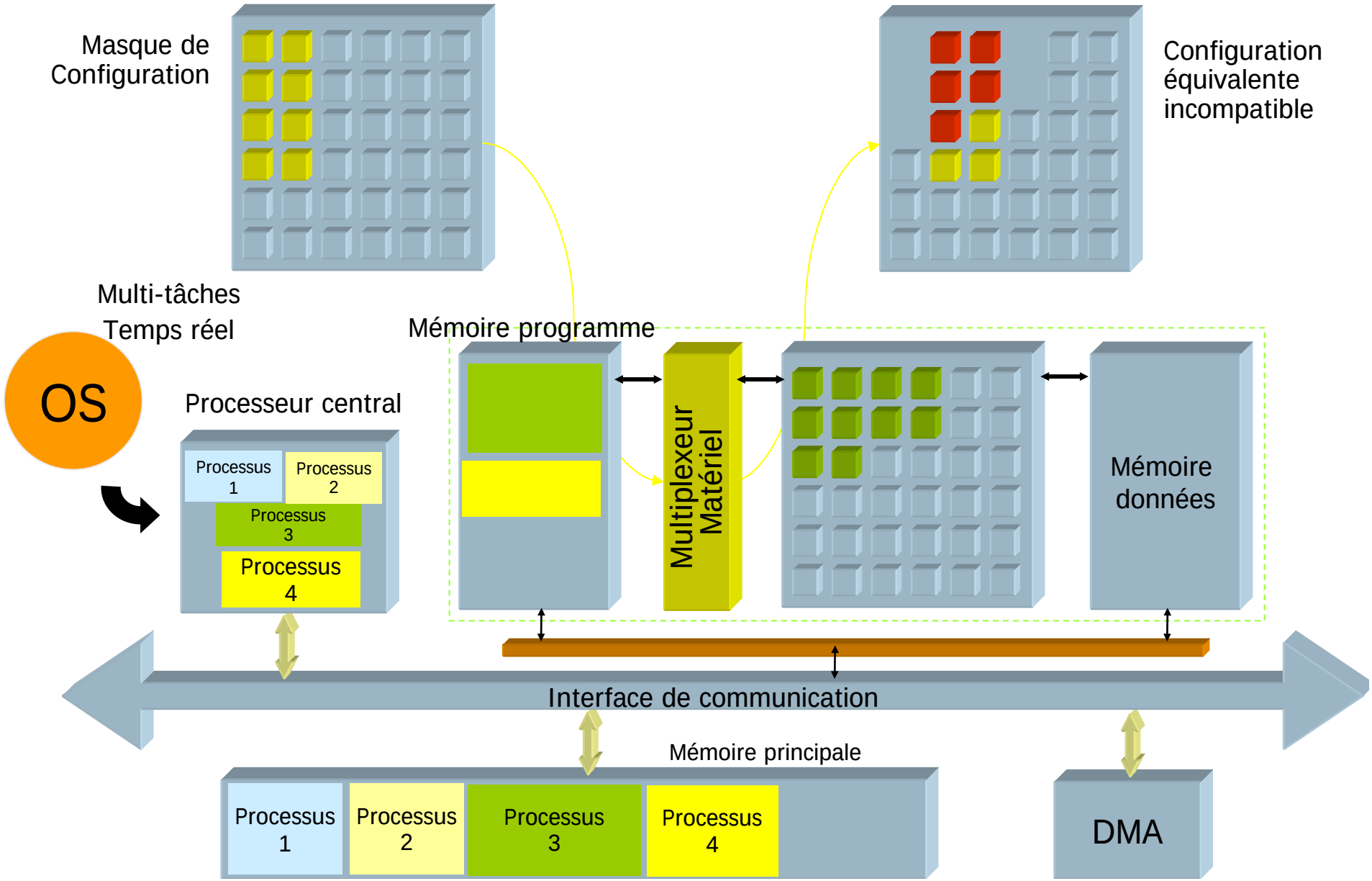
4- EXEMPLE DE GESTION DYNAMIQUE



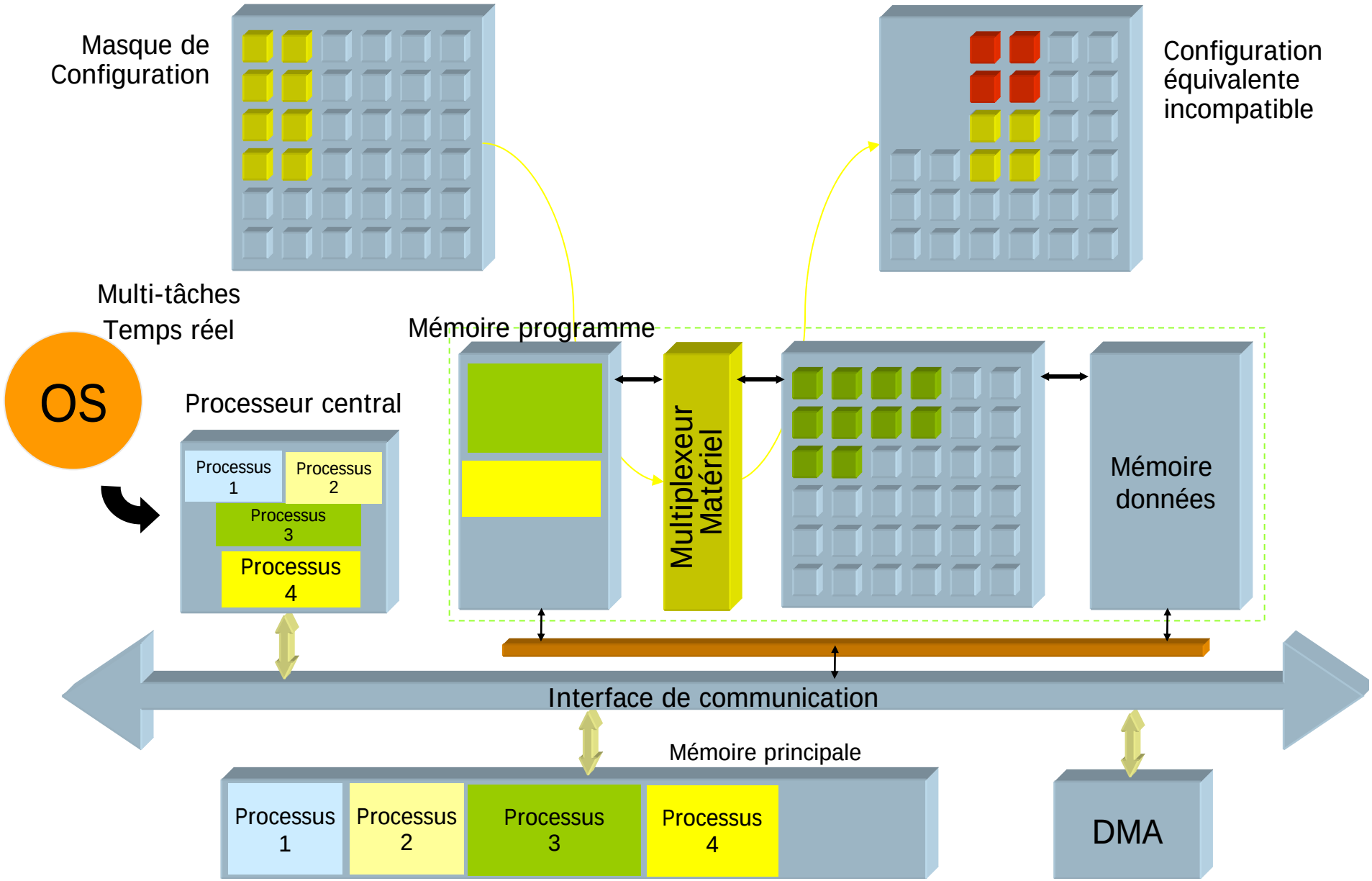
4- EXEMPLE DE GESTION DYNAMIQUE



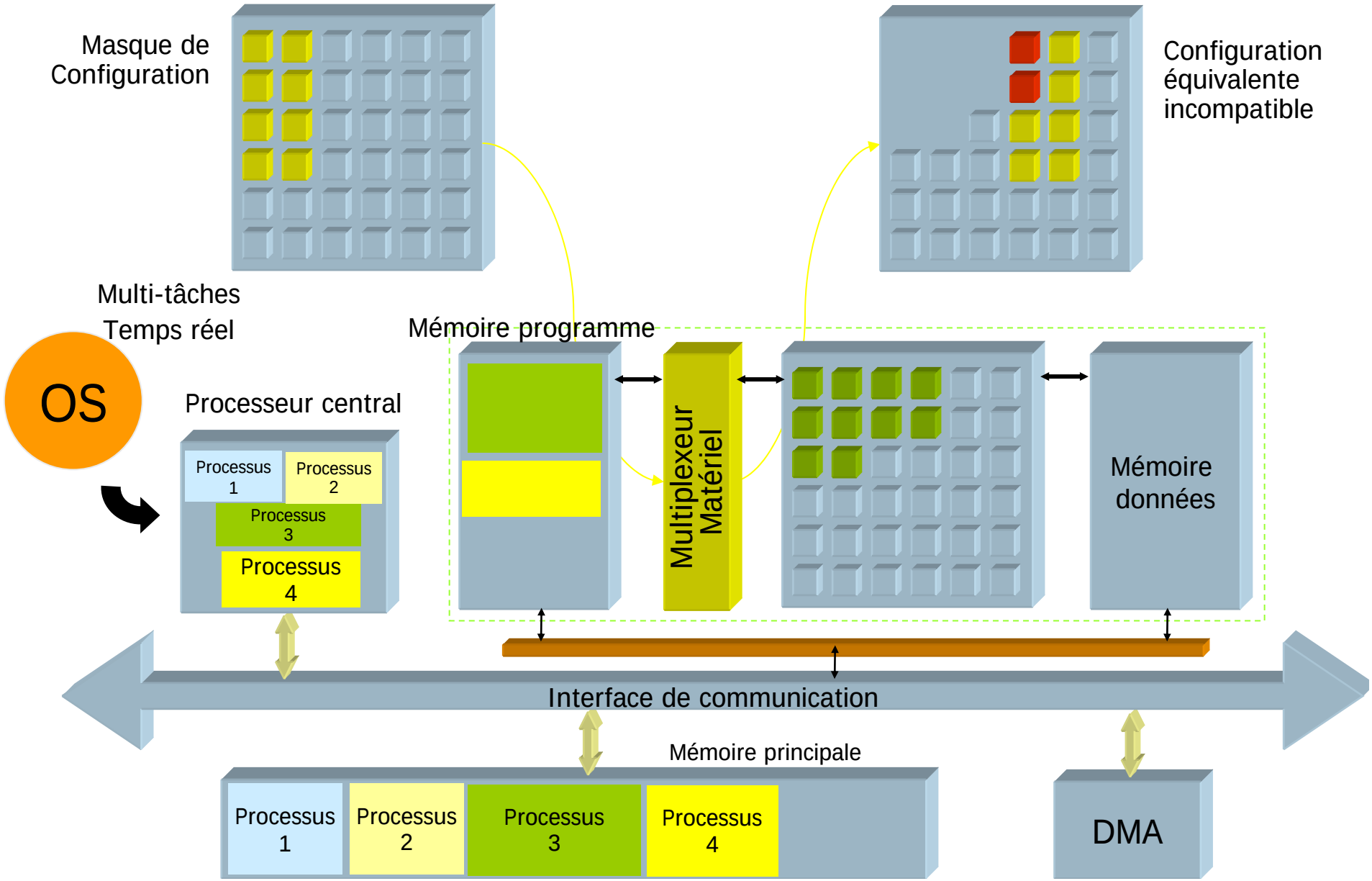
4- EXEMPLE DE GESTION DYNAMIQUE



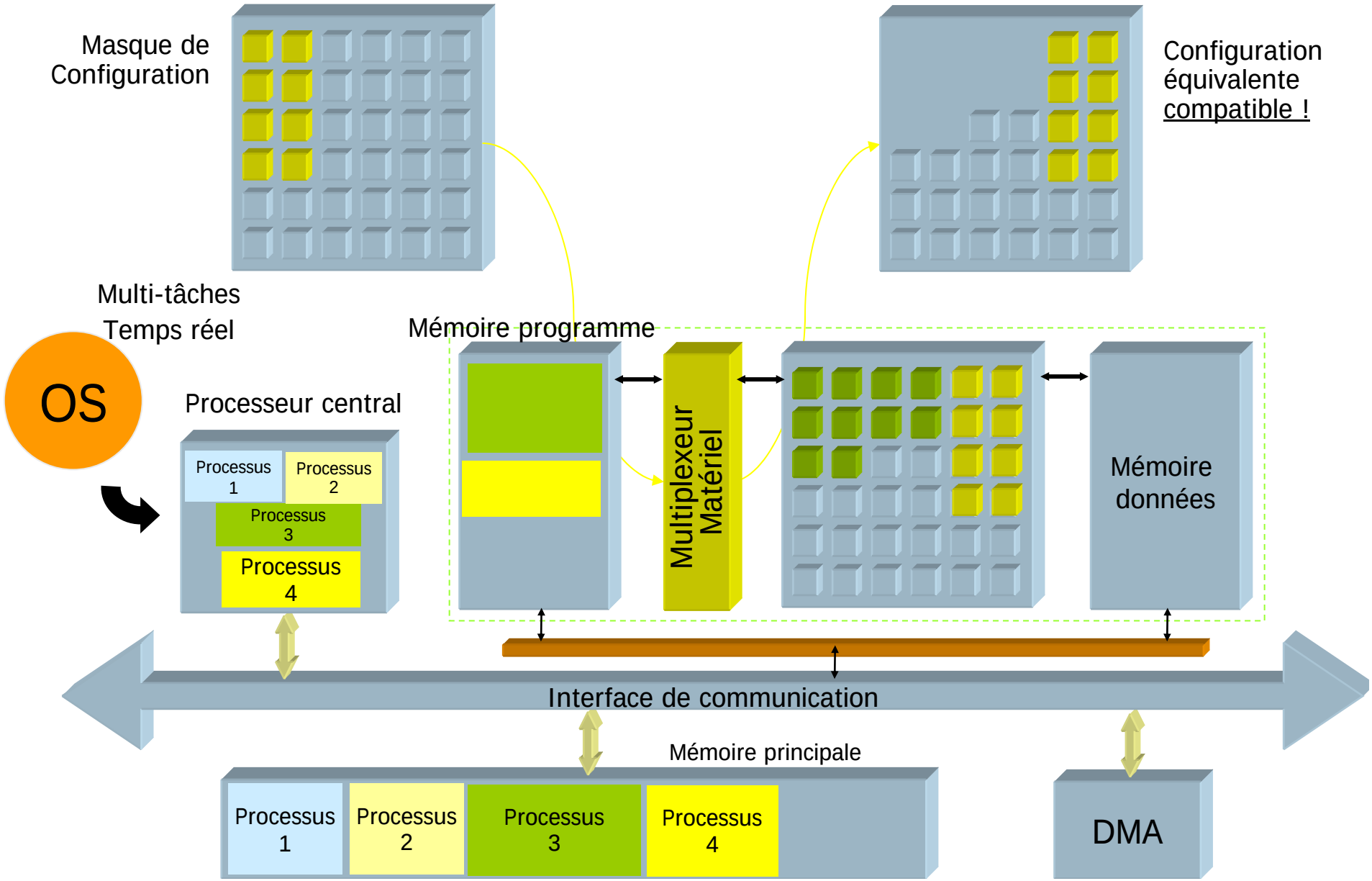
4- EXEMPLE DE GESTION DYNAMIQUE



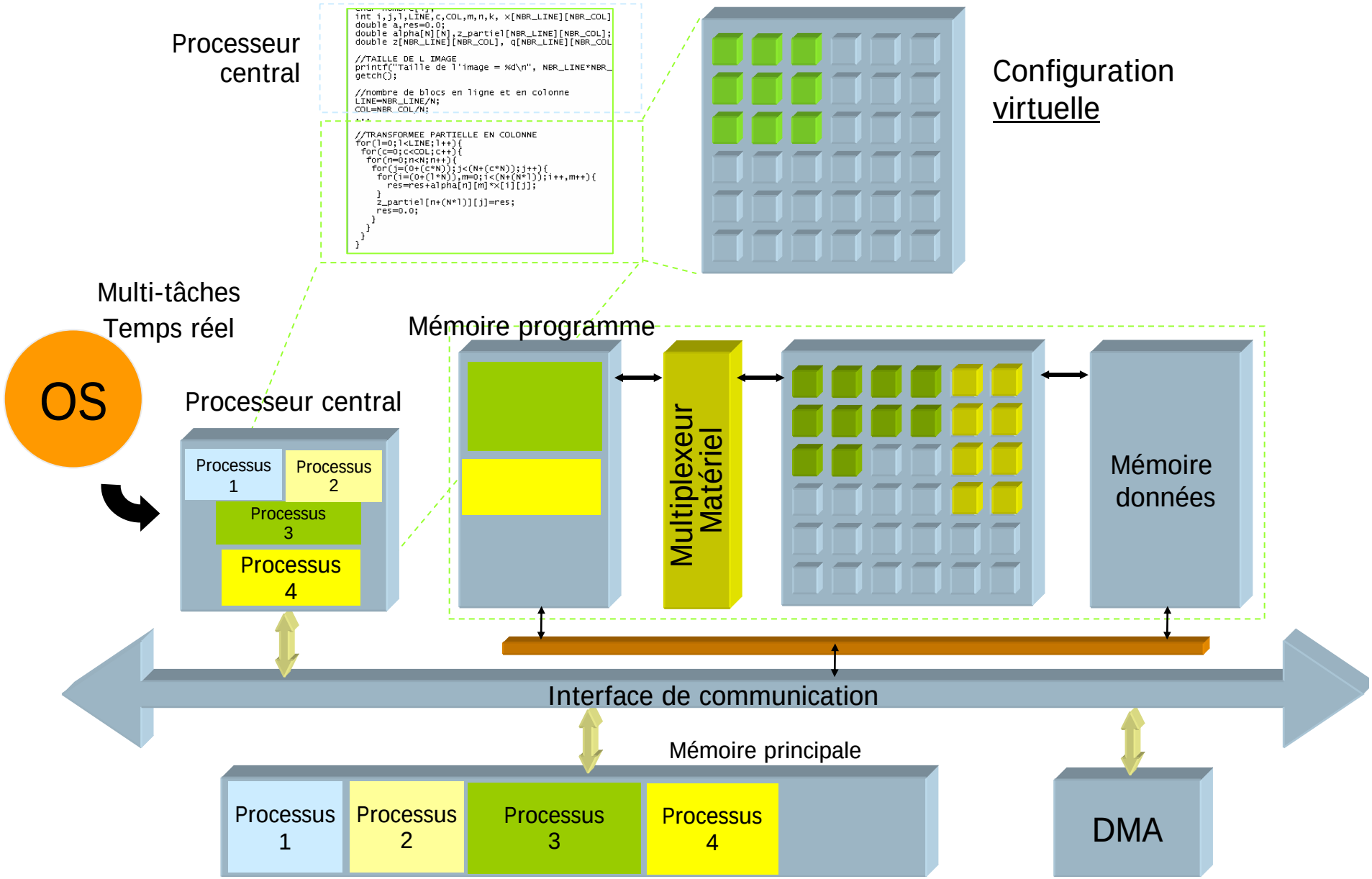
4- EXEMPLE DE GESTION DYNAMIQUE



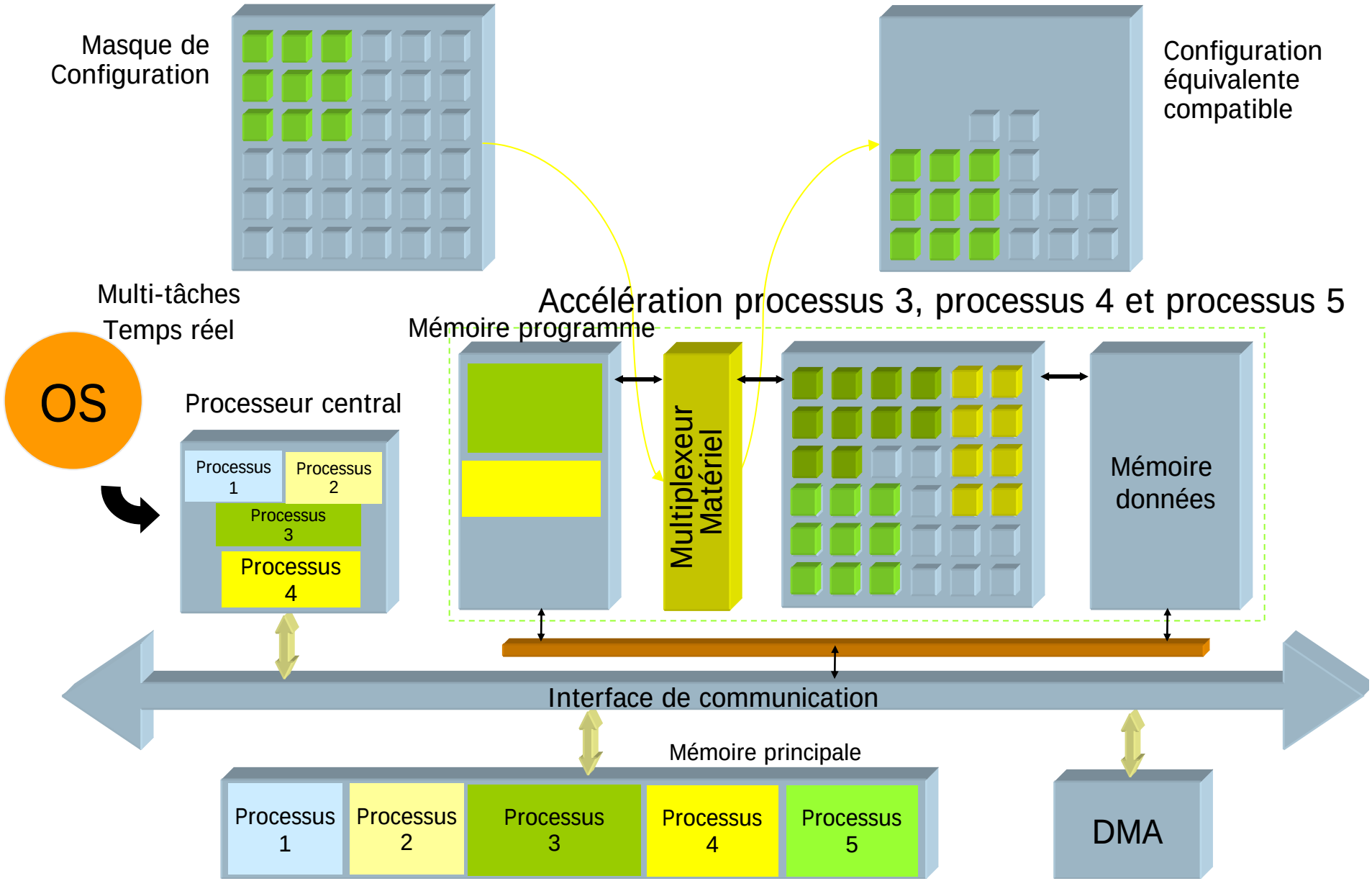
4- EXEMPLE DE GESTION DYNAMIQUE



4- EXEMPLE DE GESTION DYNAMIQUE

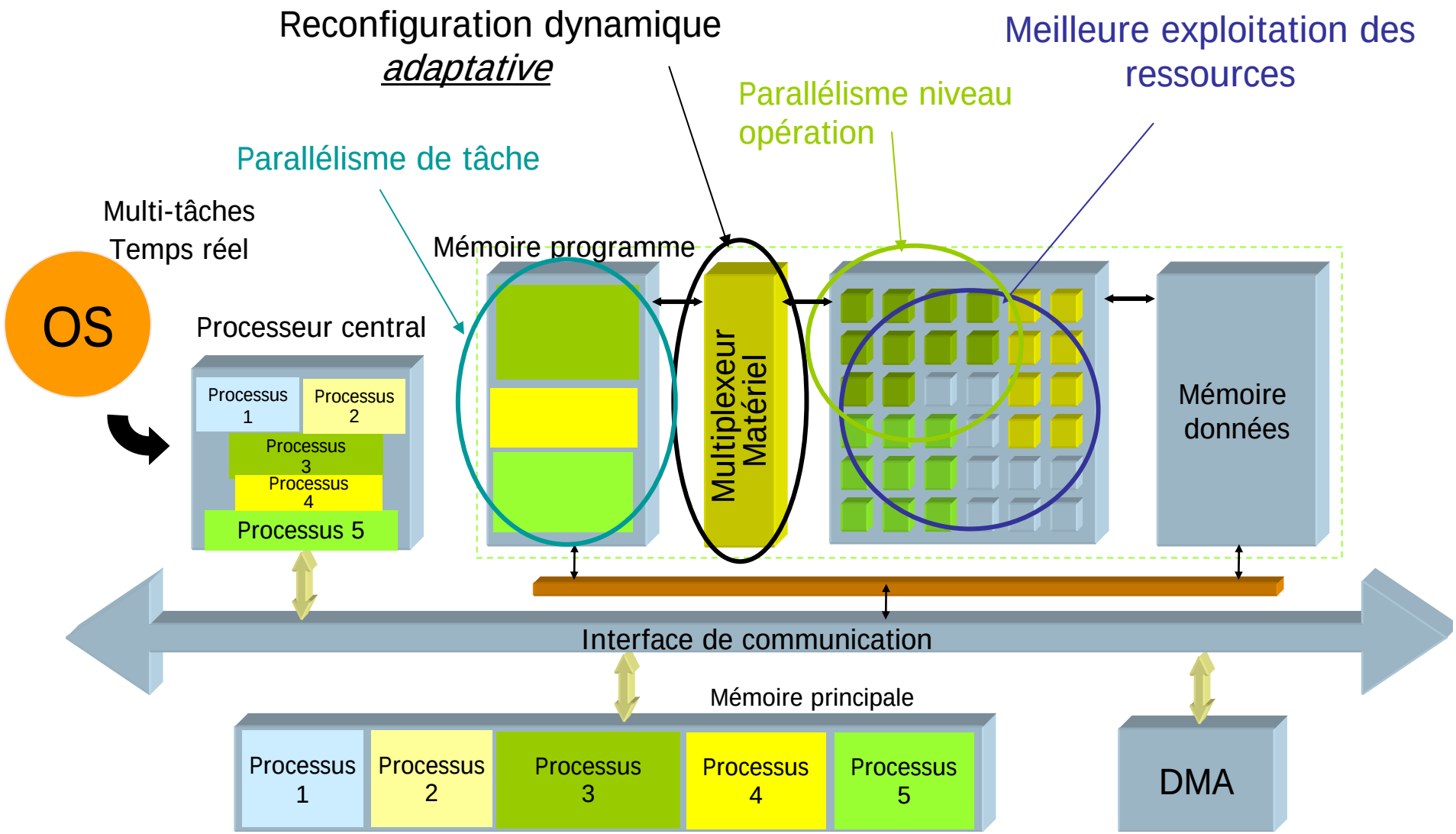


4- EXEMPLE DE GESTION DYNAMIQUE



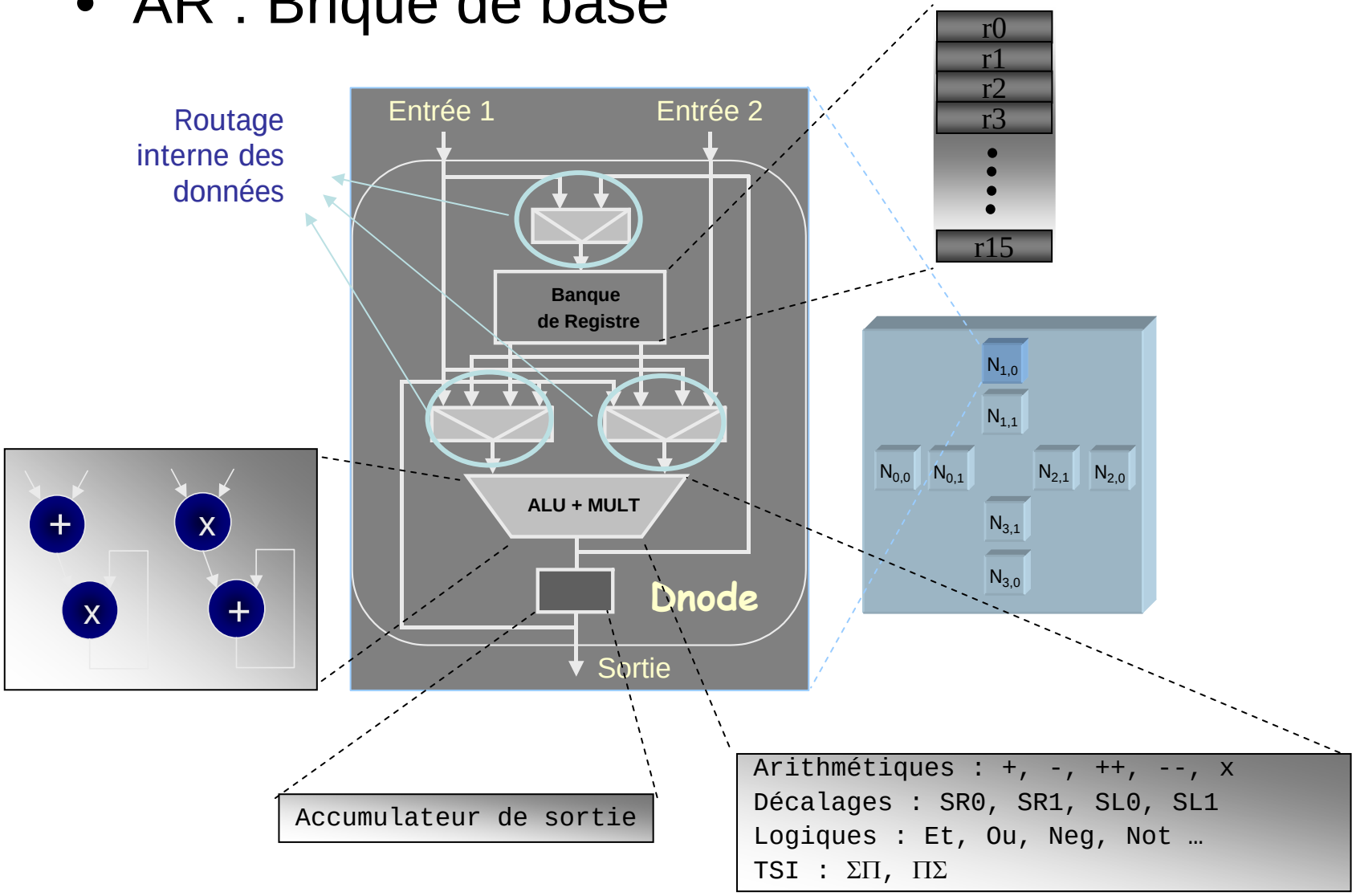
4- EXEMPLE DE GESTION DYNAMIQUE

Bilan



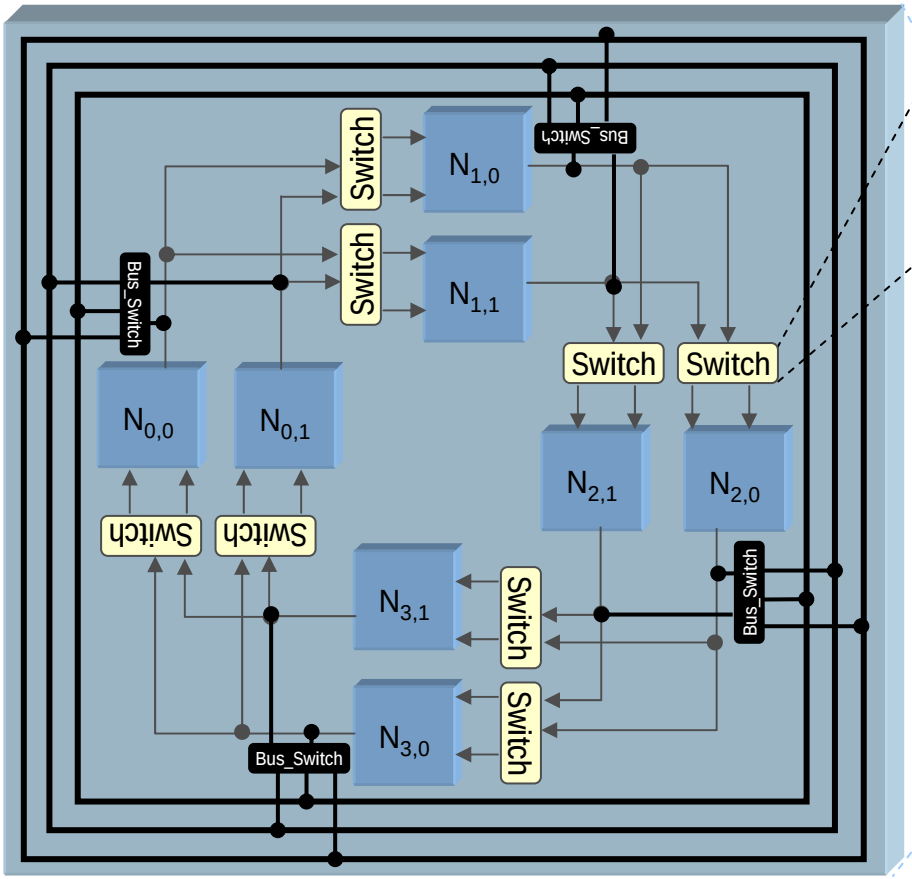
4- EXEMPLE DE GESTION DYNAMIQUE

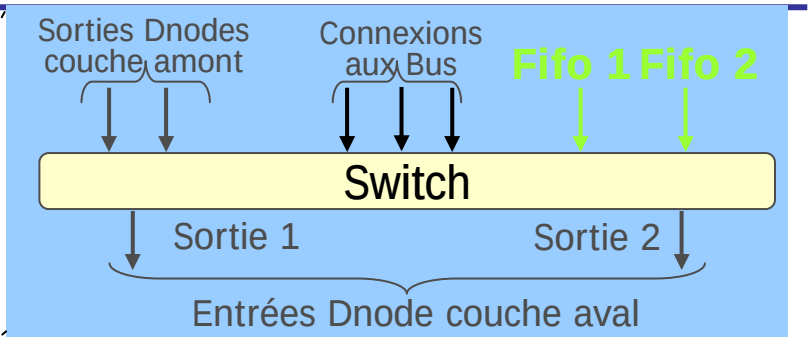
- AR : Brique de base

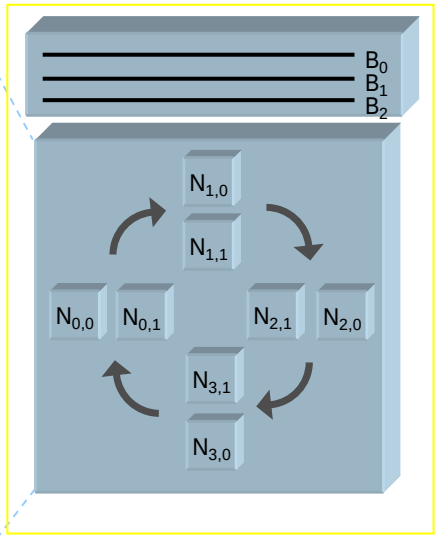


4- EXEMPLE DE GESTION DYNAMIQUE

- AR : Topologie







4- EXEMPLE DE GESTION DYNAMIQUE

- Parallélisme dynamique

Programme tâche 1

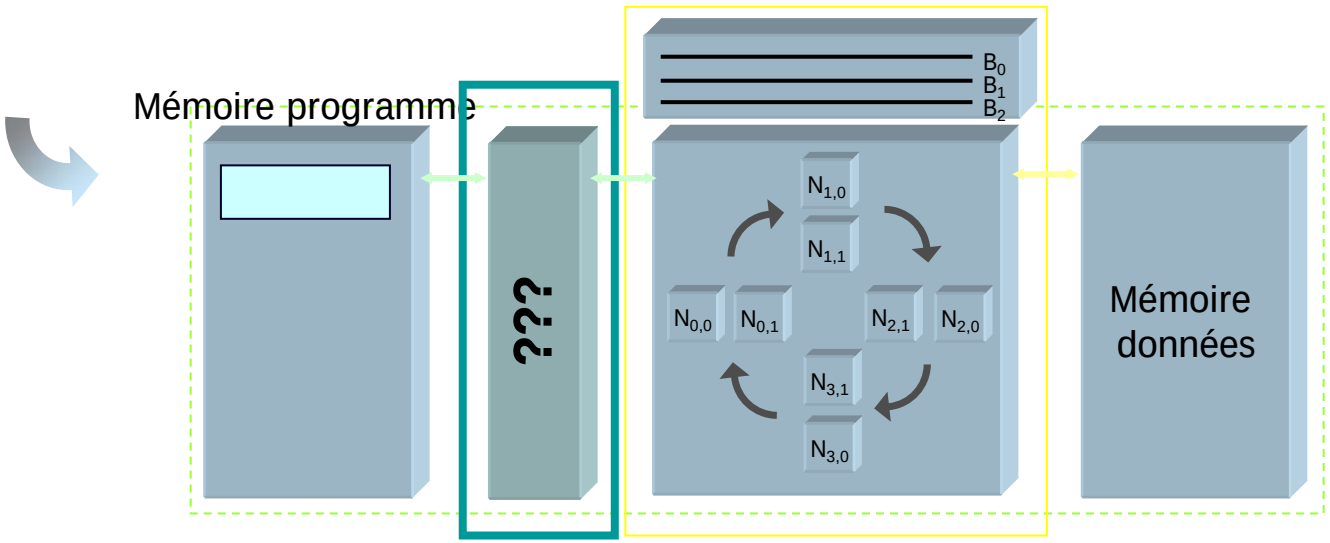
```
0000 EN-TETE TACHE 1
0001 M1: N1:add   N2:sub
0002 M2: N1:mac   N2:mac
```

Utilisation d'un en-tête de configuration pour chaque tâche

Ressources nécessaires

Facteur de duplication

Topologie



4- EXEMPLE DE GESTION DYNAMIQUE

- Parallélisme dynamique

Programme tâche 2

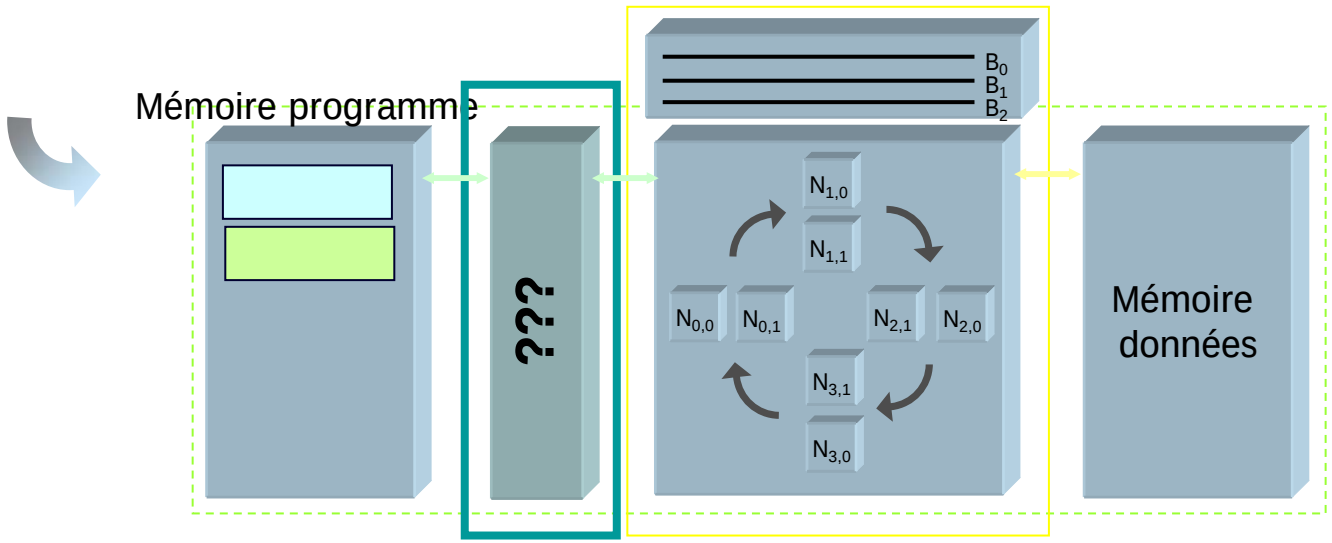
```
0003 EN-TETE TACHE 2
0004 M1: N1:sub   N2:sub
0005 M2: N1:ou    N2:nop
```

Utilisation d'un en-tête de configuration pour chaque tâche

Ressources nécessaires

Facteur de duplication

Topologie



4- EXEMPLE DE GESTION DYNAMIQUE

- Parallélisme dynamique

Programme tâche 3

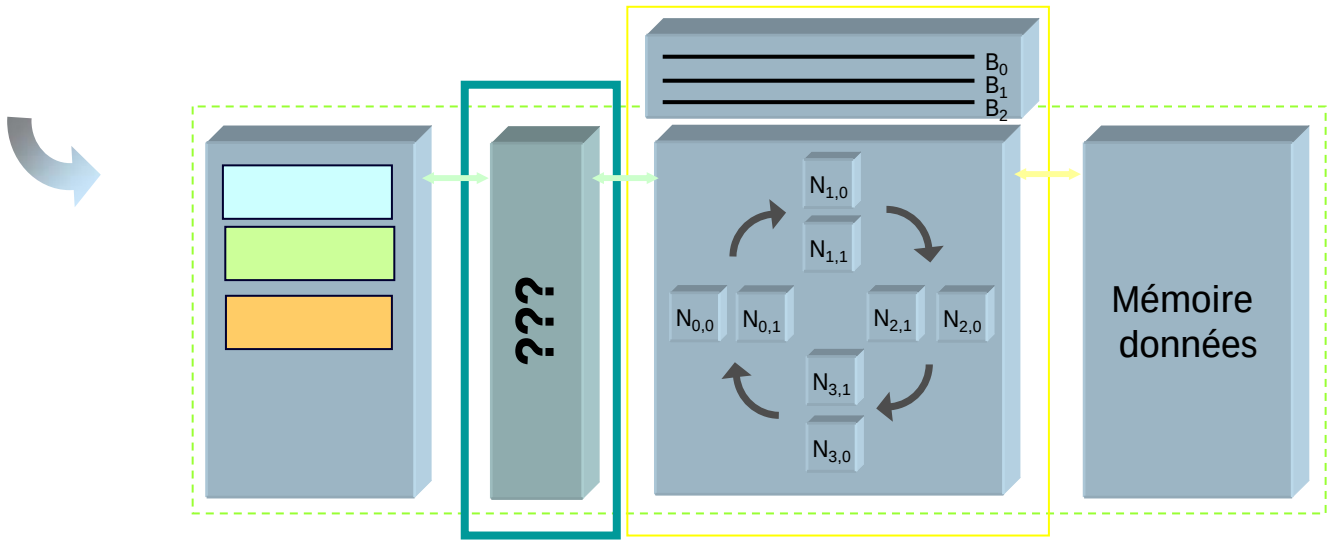
```
0006 EN-TETE TACHE 3
0007 M1: N1:add   N2:sub
0008 M2: N1:mult  N2:nop
```

Utilisation d'un en-tête de configuration pour chaque tâche

Ressources nécessaires

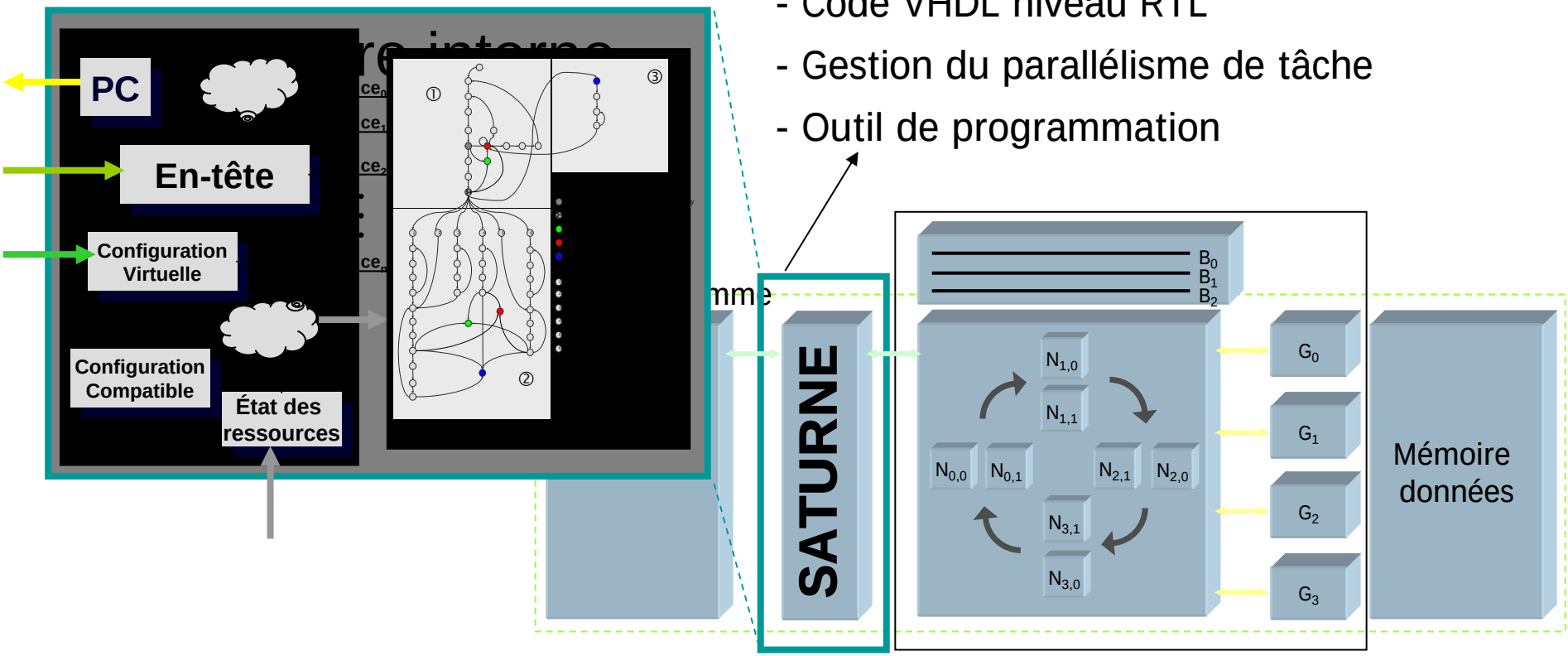
Facteur de duplication

Topologie



4- EXEMPLE DE GESTION DYNAMIQUE

- Code VHDL niveau RTL
- Gestion du parallélisme de tâche
- Outil de programmation



# lignes	# N	# B	# G	B_0	B_1	B_2	$N_{0,0}$	$N_{0,1}$	$N_{1,0}$	$N_{1,1}$	$N_{2,0}$	$N_{2,1}$	$N_{3,0}$	$N_{3,1}$	G_0	G_1	G_2	G_3

4- EXEMPLE DE GESTION DYNAMIQUE

- Simulation

Tâche 1

$$S=(a+b)^2$$

```
for(i=0;i<512;i++){
temp=a[i]+b[i];
s[i]=pow(temp,2);
}
```

Tâche 2

$$S1=(a+b)$$
$$S2=(a-b)$$

```
for(i=0;i<512;i++){
  s1[i]=a[i]+b[i];
  s2[i]= a[i]-b[i];
}
```

Tâche 3

$$S=(a^2-b^2)$$

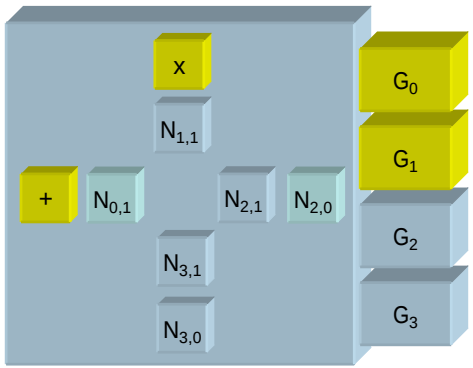
```
for(i=0;i<256;i++){
s[i]=(a[i]+b[i])*(a[i]-
b[i]);
}
```

Tâche 4

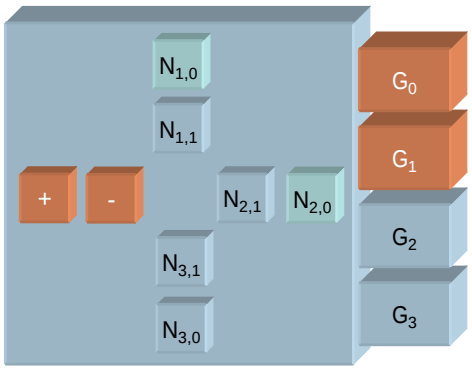
$$S=\frac{a^2-b^2}{2}$$

```
for(i=0;i<256;i++){
temp=(a[i]+b[i])*(a[i]-b[i]);
s[i]=temp/2;
}
```

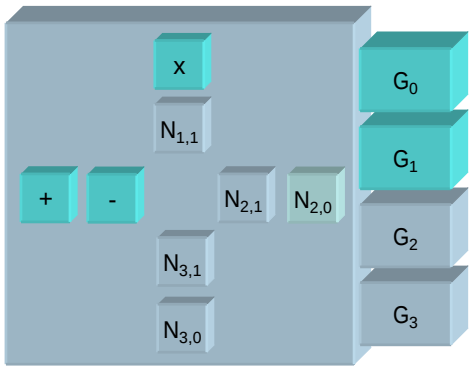
CONFIGURATION 1



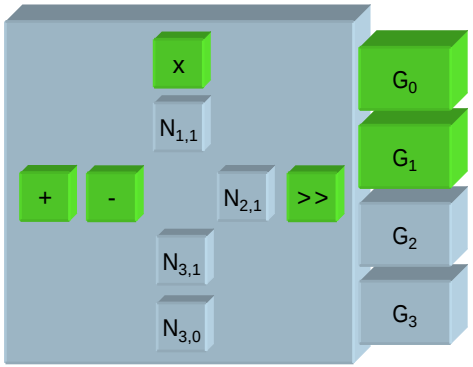
CONFIGURATION 2



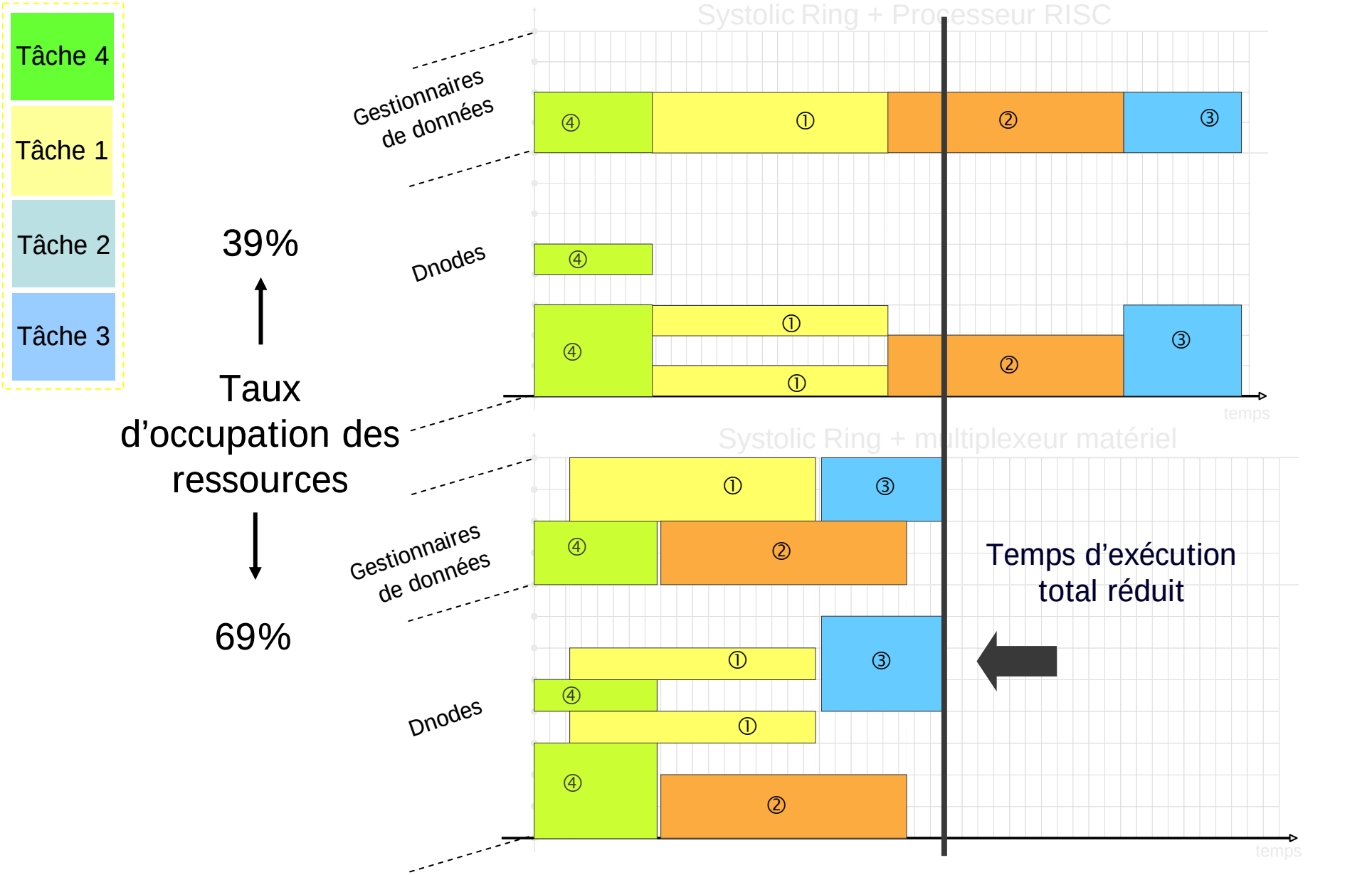
CONFIGURATION 3



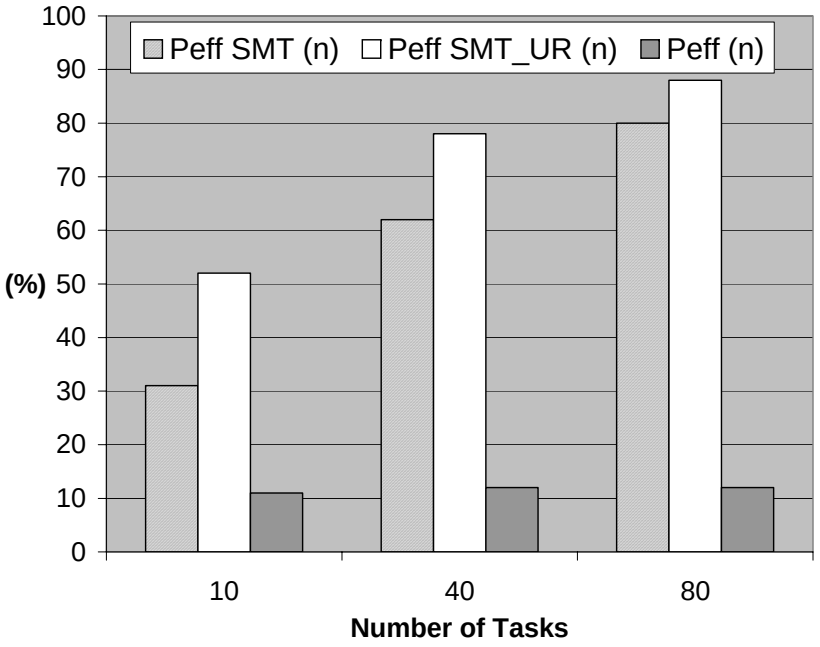
CONFIGURATION 4



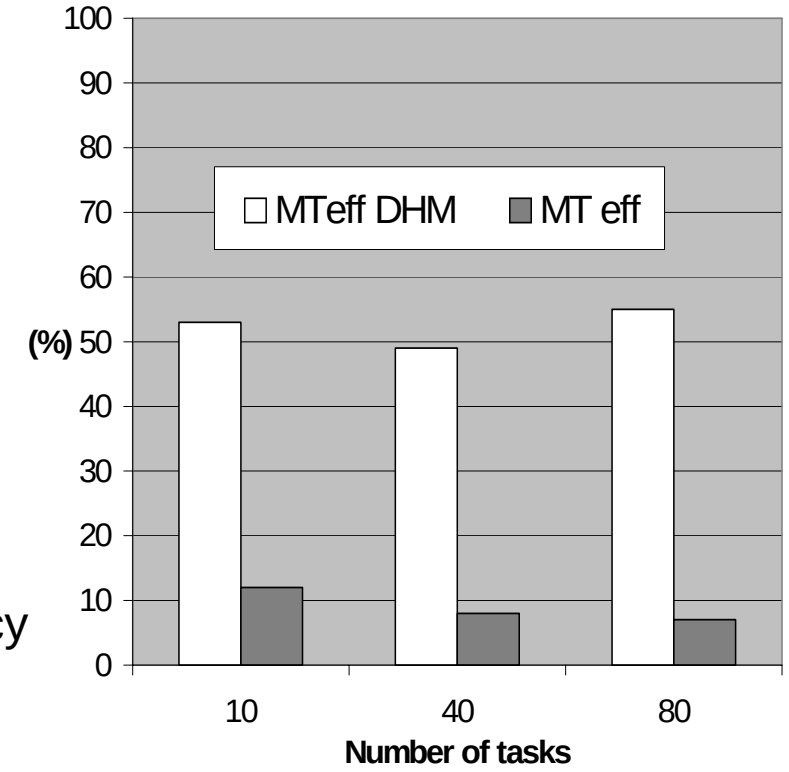
4- EXEMPLE DE GESTION DYNAMIQUE



4- EXEMPLE DE GESTION DYNAMIQUE



Peff : Processing efficiency



MTeff : Multi-tasking efficiency

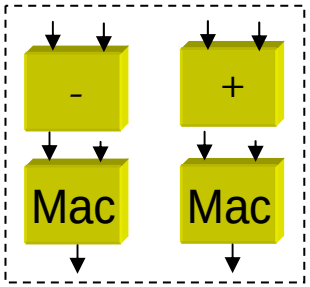
Moyenne sur 300 scénarios (10,40,80 tâches)

4- EXEMPLE DE GESTION DYNAMIQUE

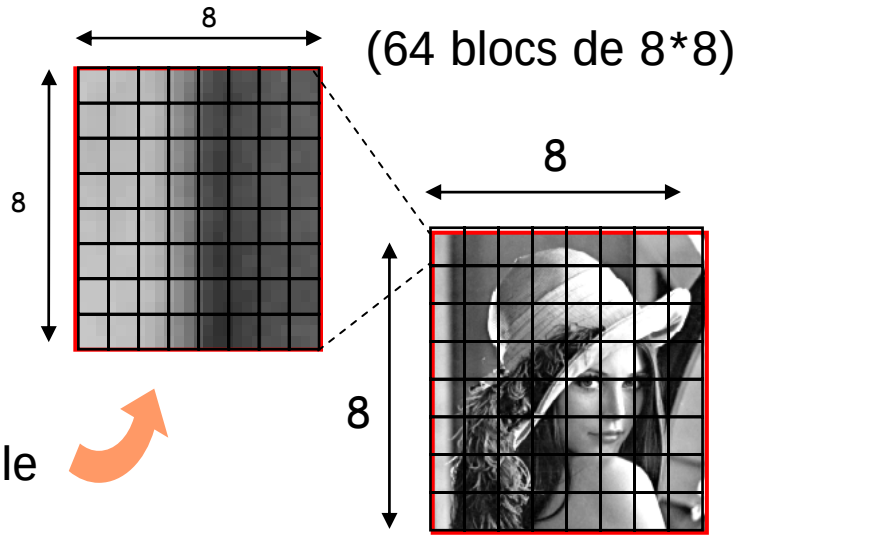
Multiplexage matériel

Parallélisme de Boucle

Exemple : DCT 2D (8*8) (MPEG-2)



Motif de traitement non déroulé exploitant le parallélisme spatial et temporel



(64 blocs de 8*8)

Image vidéo composée de 64*64 pixels

Pas de déroulage nécessaire pour temps réel en 64*64

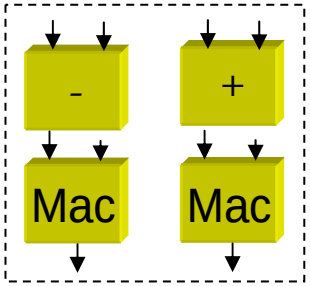
4- EXEMPLE DE GESTION DYNAMIQUE

Multiplexage matériel

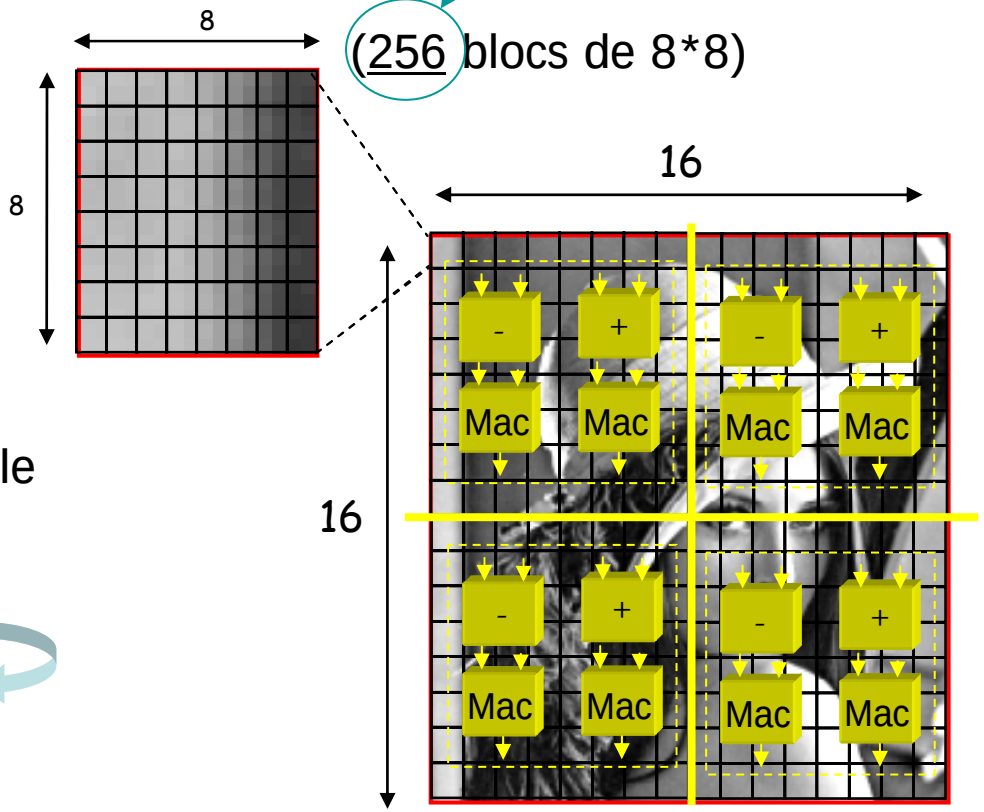
Parallélisme de Boucle

4 fois plus de blocs !

Exemple : DCT 2D (8*8) (MPEG-2)

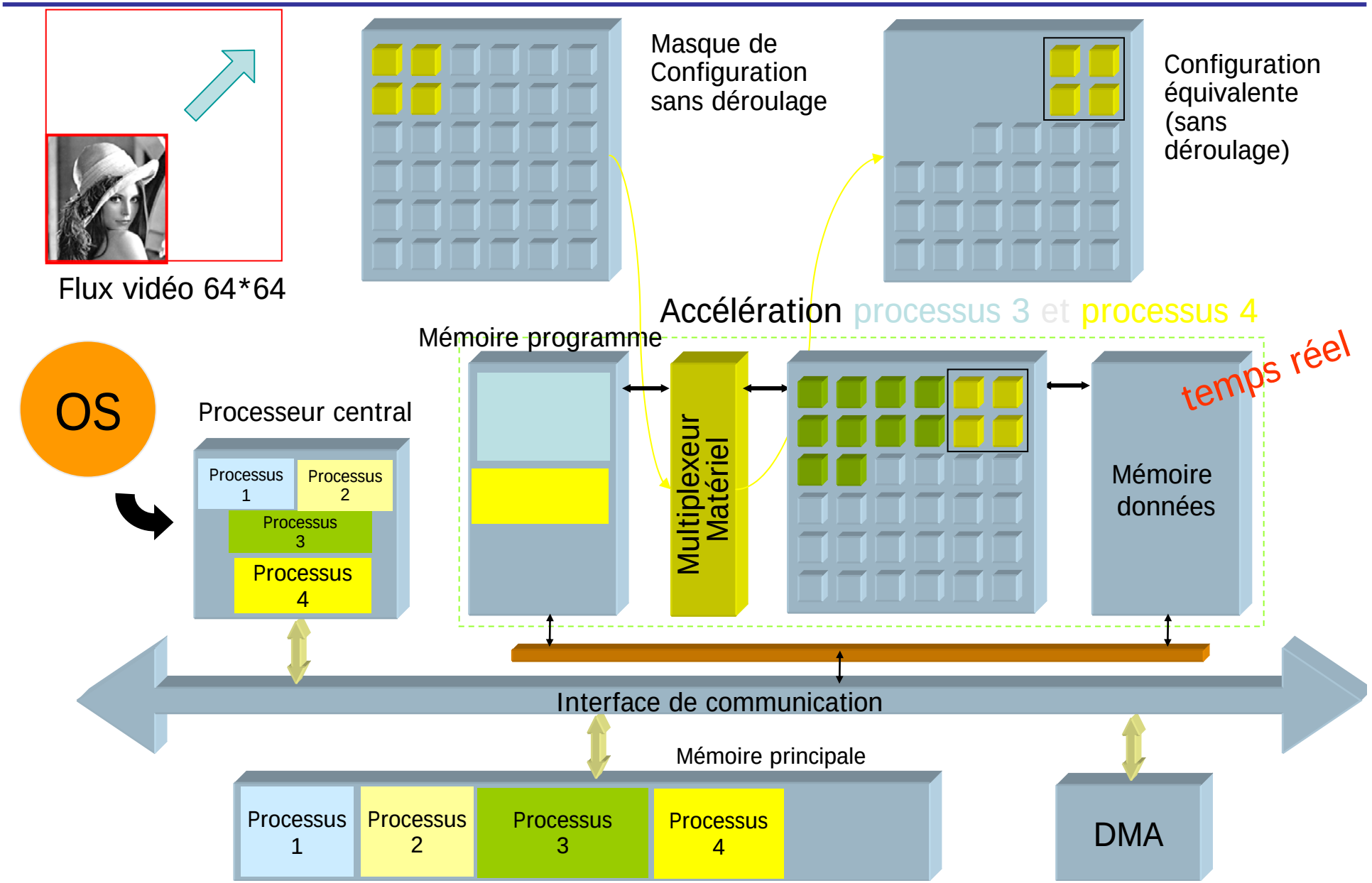


Motif de traitement non déroulé exploitant le parallélisme spatial et temporel



Déroulage de 4 pour temps réel en 128*128

4- EXEMPLE DE GESTION DYNAMIQUE



4- EXEMPLE DE GESTION DYNAMIQUE

Flux vidéo 128*128

Masque de Configuration sans déroulage

Configuration équivalente déroulée dynamiquement 4 fois

Accélération processus 3 et processus 4

Mémoire programme

OS

Processeur central

Processus 1
Processus 2
Processus 3
Processus 4

Multiplexeur Matériel

Mémoire données

temps réel

Interface de communication

Mémoire principale

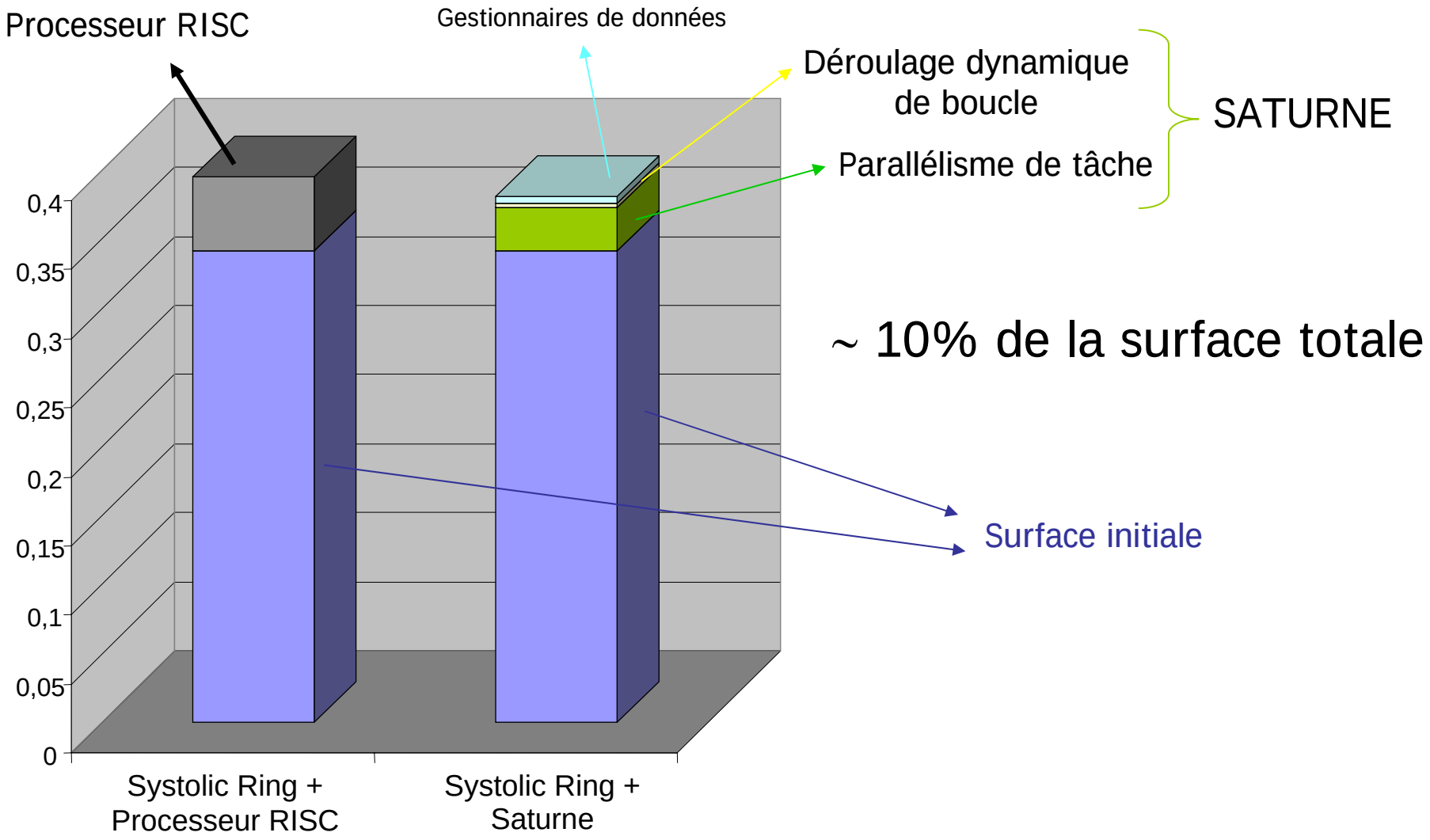
Processus 1
Processus 2
Processus 3
Processus 4

DMA

4- EXEMPLE DE GESTION DYNAMIQUE

- Synthèse logique

Synthèse (ST CMOS 0,13µ)



~ 10% de la surface totale

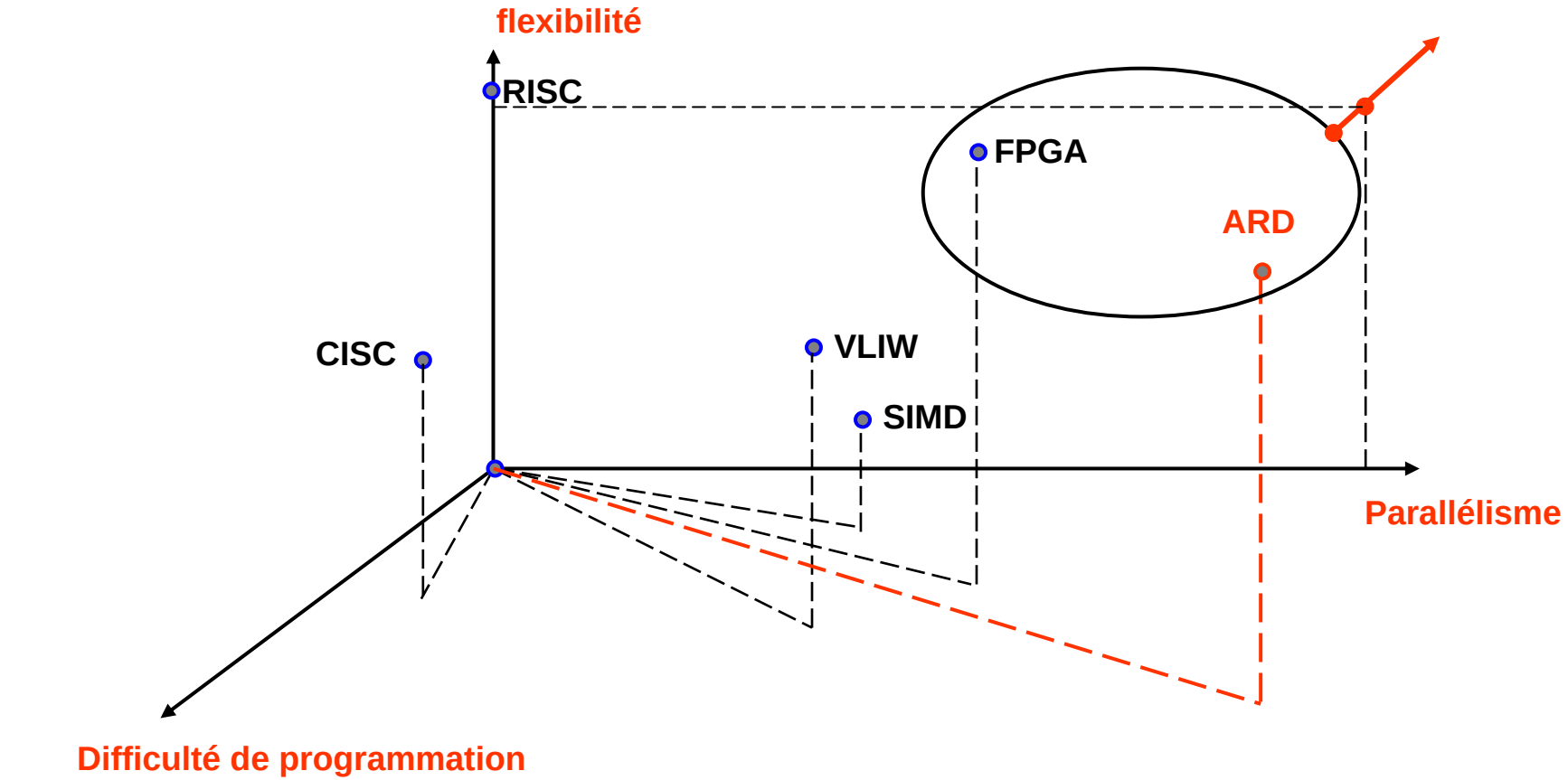
Sommaire

Partie 2 : Les architectures reconfigurables dynamiquement

1- INTRODUCTION & CONTEXTE
2- CLASSIFICATION D'ARCHITECTURES
3- QUELQUES EXEMPLES
4- EXEMPLE DE GESTION DYNAMIQUE
5- CONCLUSIONS & PERSPECTIVES

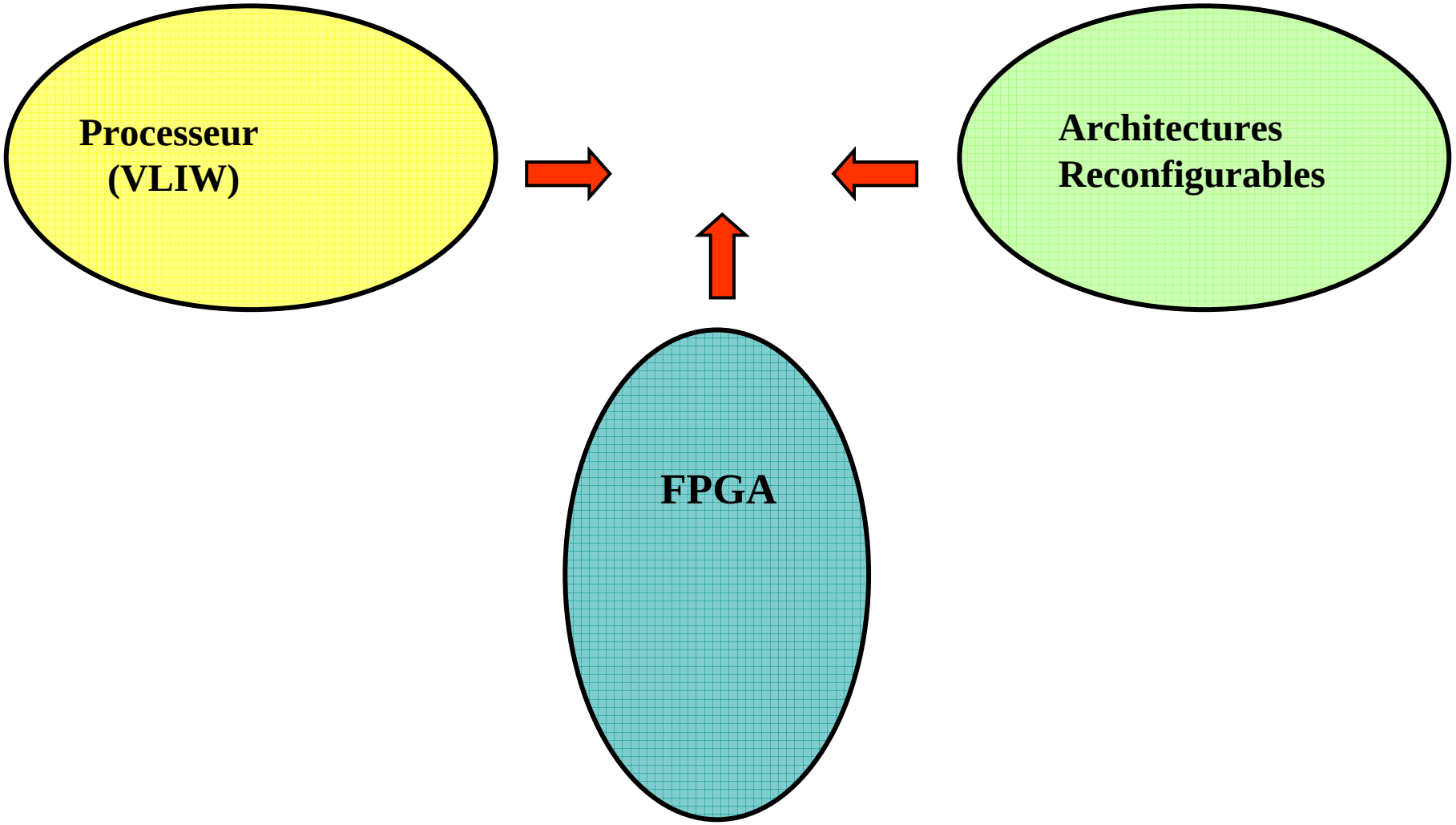
5- CONCLUSIONS & PERSPECTIVES

Maximiser flexibilité, performances et flexibilité de programmation…

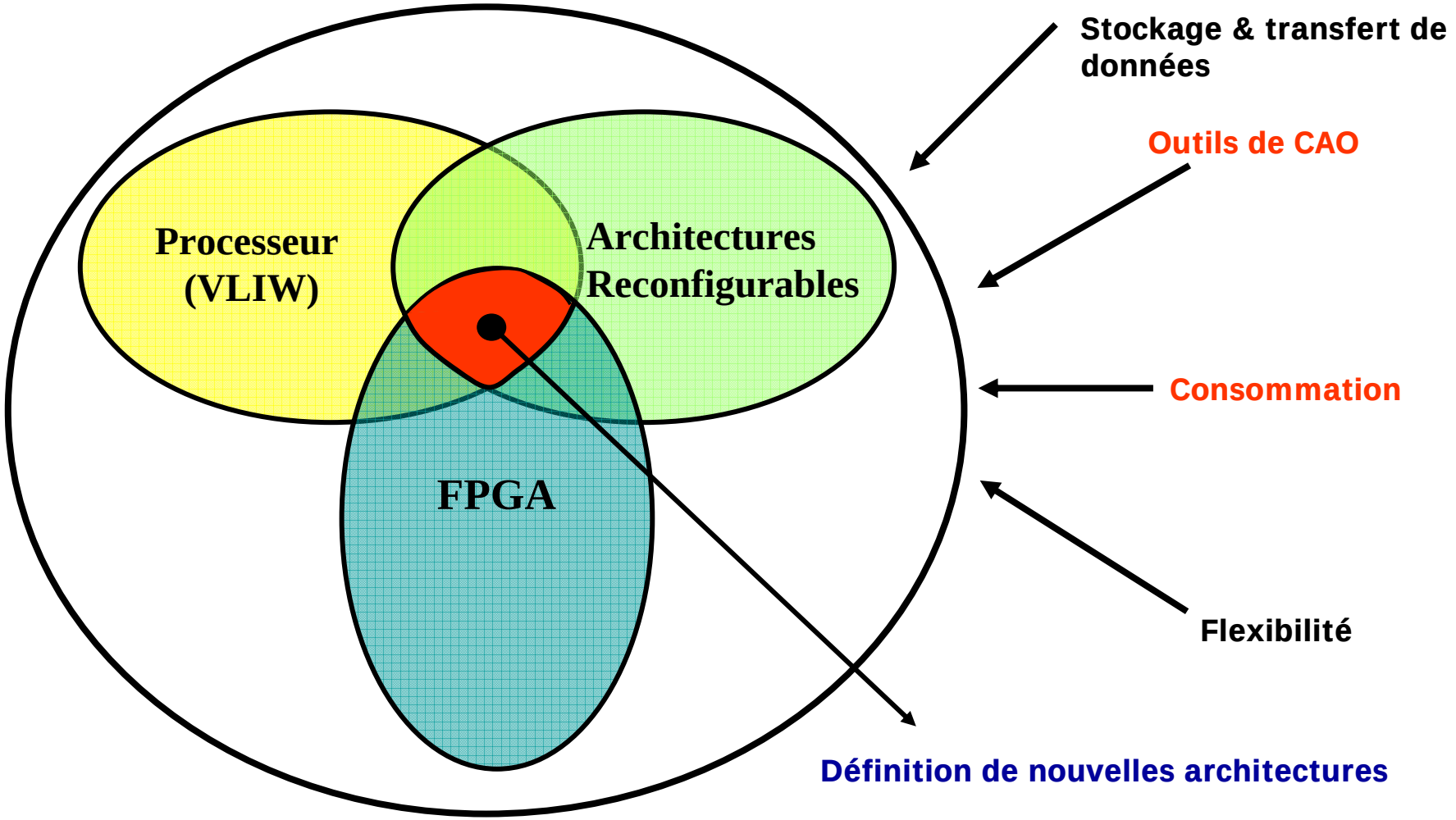


Classification d'architectures

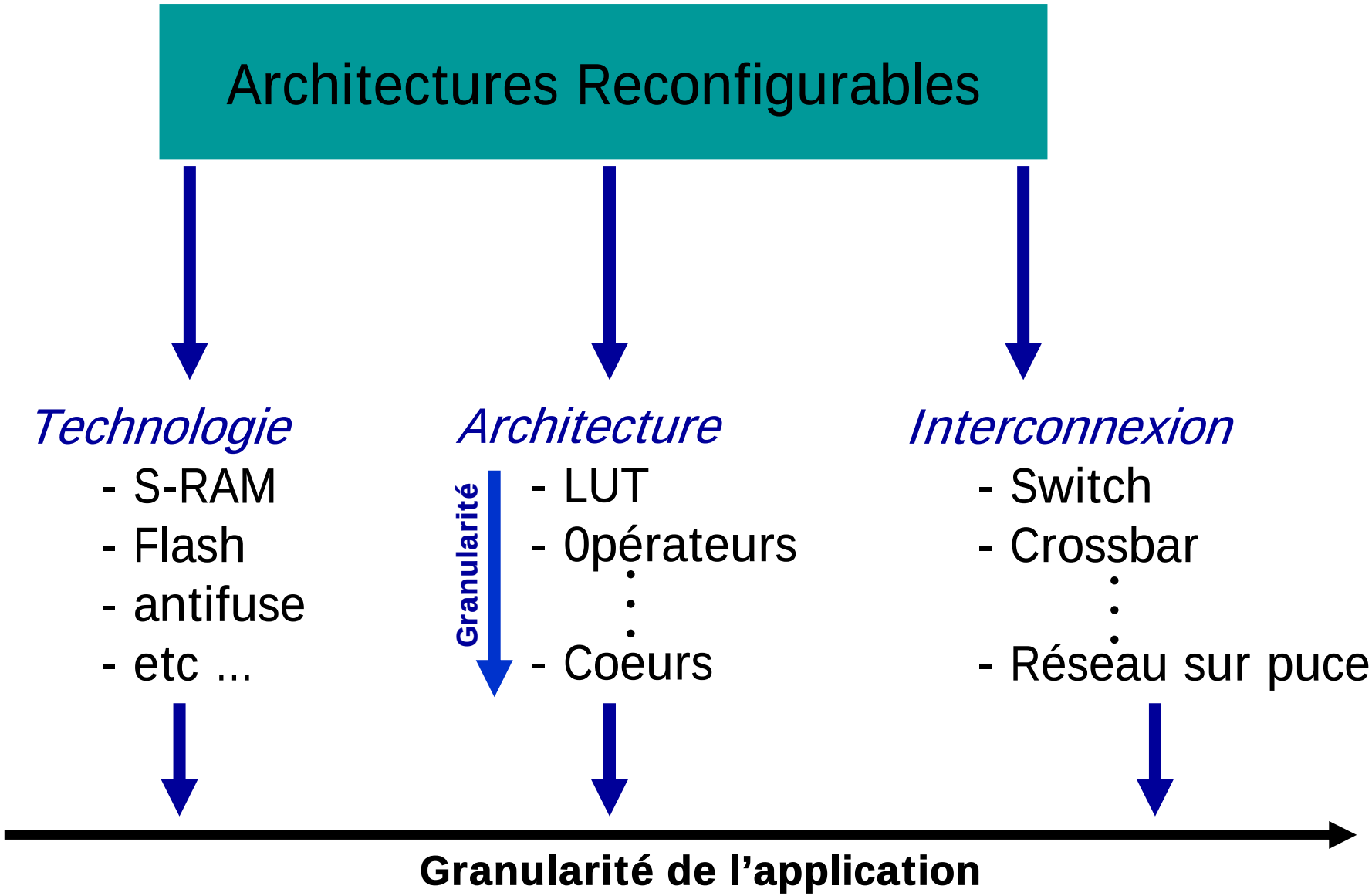
5- CONCLUSIONS & PERSPECTIVES



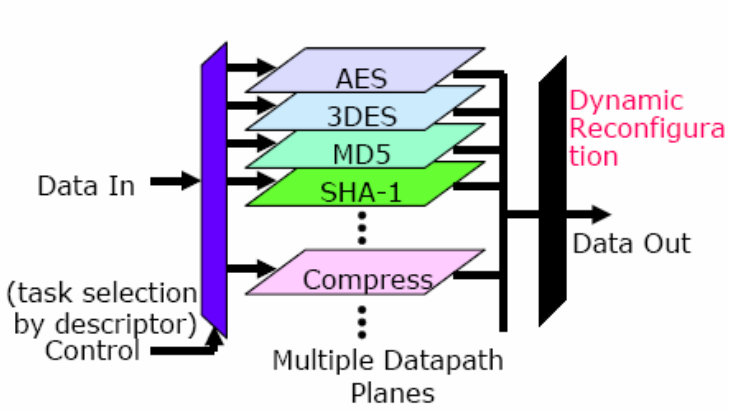
5- CONCLUSIONS & PERSPECTIVES

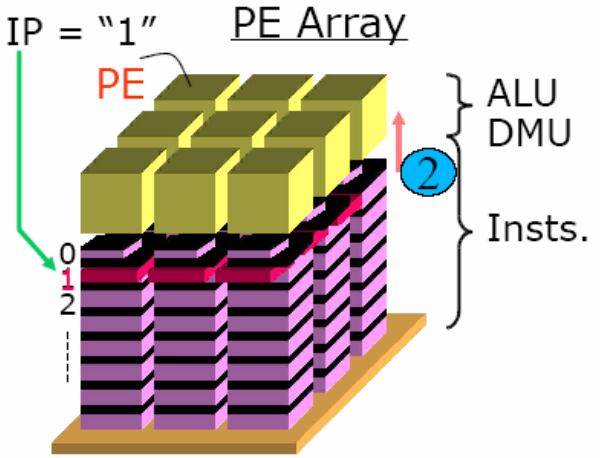


5- CONCLUSIONS & PERSPECTIVES

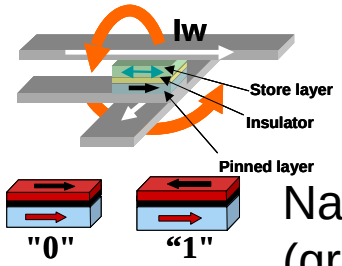


5- CONCLUSIONS & PERSPECTIVES

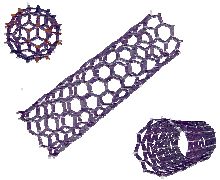


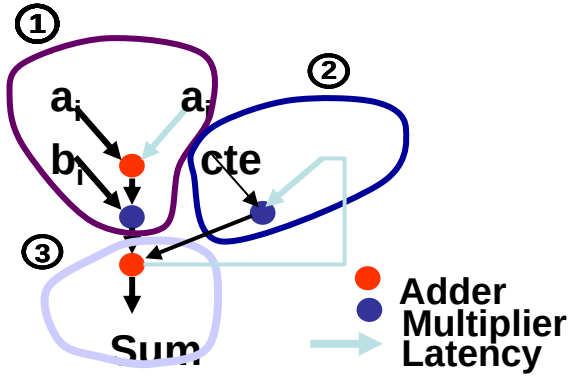


Grain épais, très fort niveau de parallélisme



Nano FPGA (grain très fin)





Outils CAO, compilateur (!)