Codesign ou Conception Conjointe Logiciel/Matériel

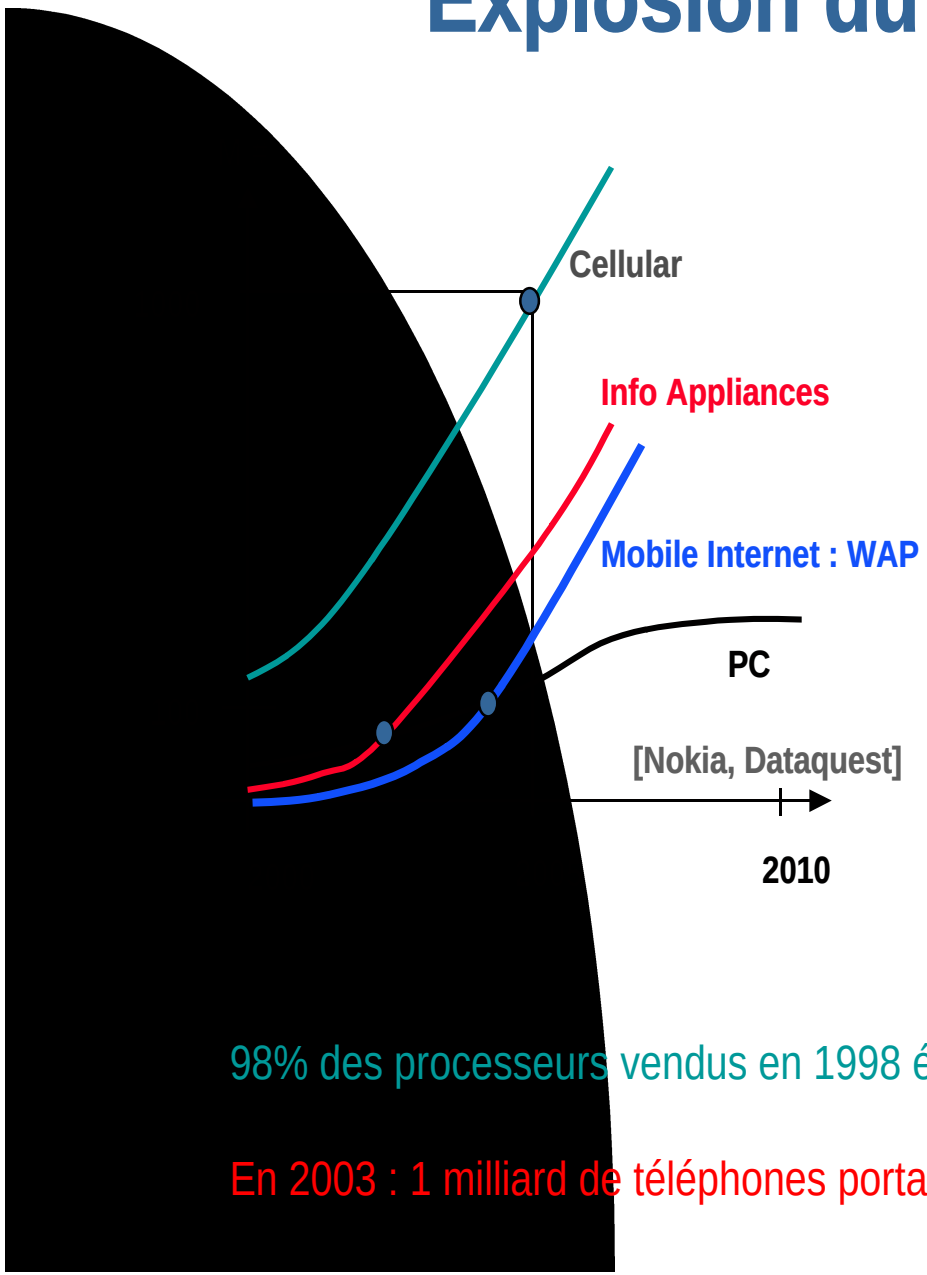
Jean Philippe Diguet - Guy Gogniat - Eric Martin
LESTER - Université de Bretagne Sud
http://lester.univ-ubs.fr

Plan

- Pourquoi le Codesign ?
- Flôt de conception général
- Modélisation des applications
- Estimations Système, Logiciel, Matériel
- Partitionnement

Pourquoi le Codesign ?

Explosion du numérique embarqué

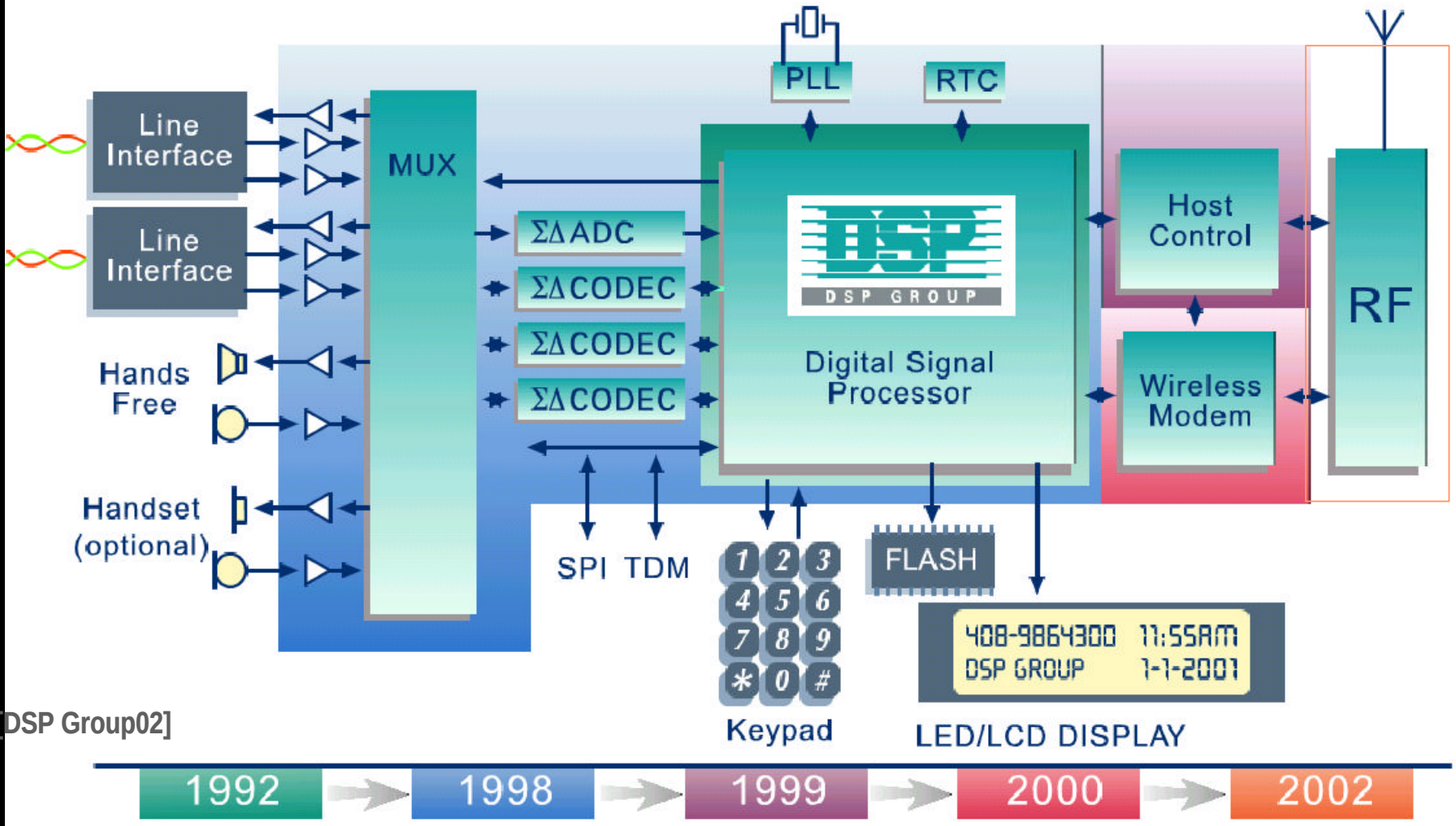


- Amélioration constante de la technologie (0.13µm)
- Accroissement constant de la complexité des applications (GSM, GPRS, UMTS…)
- Intégration des architectures hétérogènes (SOC, RSoC, …)

98% des processeurs vendus en 1998 équipent des systèmes embarqués

En 2003 : 1 milliard de téléphones portables contre 200 millions de PC

Complexité d'un téléphone portable



[DSP Group02]

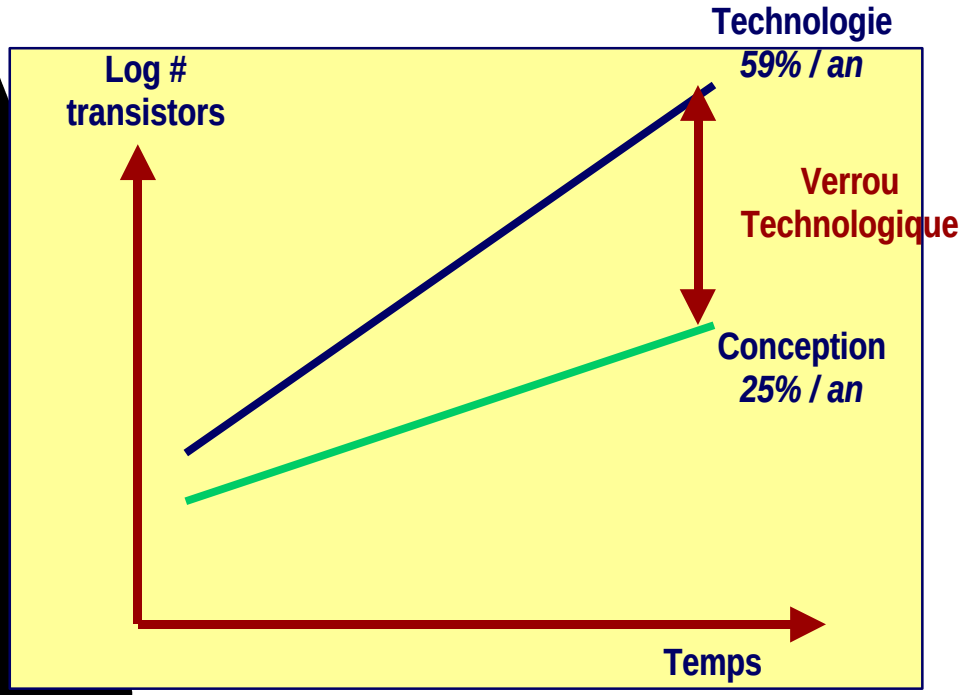
Conséquences sur le circuit

- La conception de circuit doit faire face à :
  - Intégration de systèmes (SoC)
  - Accroissement continu de la complexité
  - Rapide accès au marché
  - Matériel et Logiciel
  - Grande puissance de Calcul et/ou faible consommation

Les outils suivent-ils ?

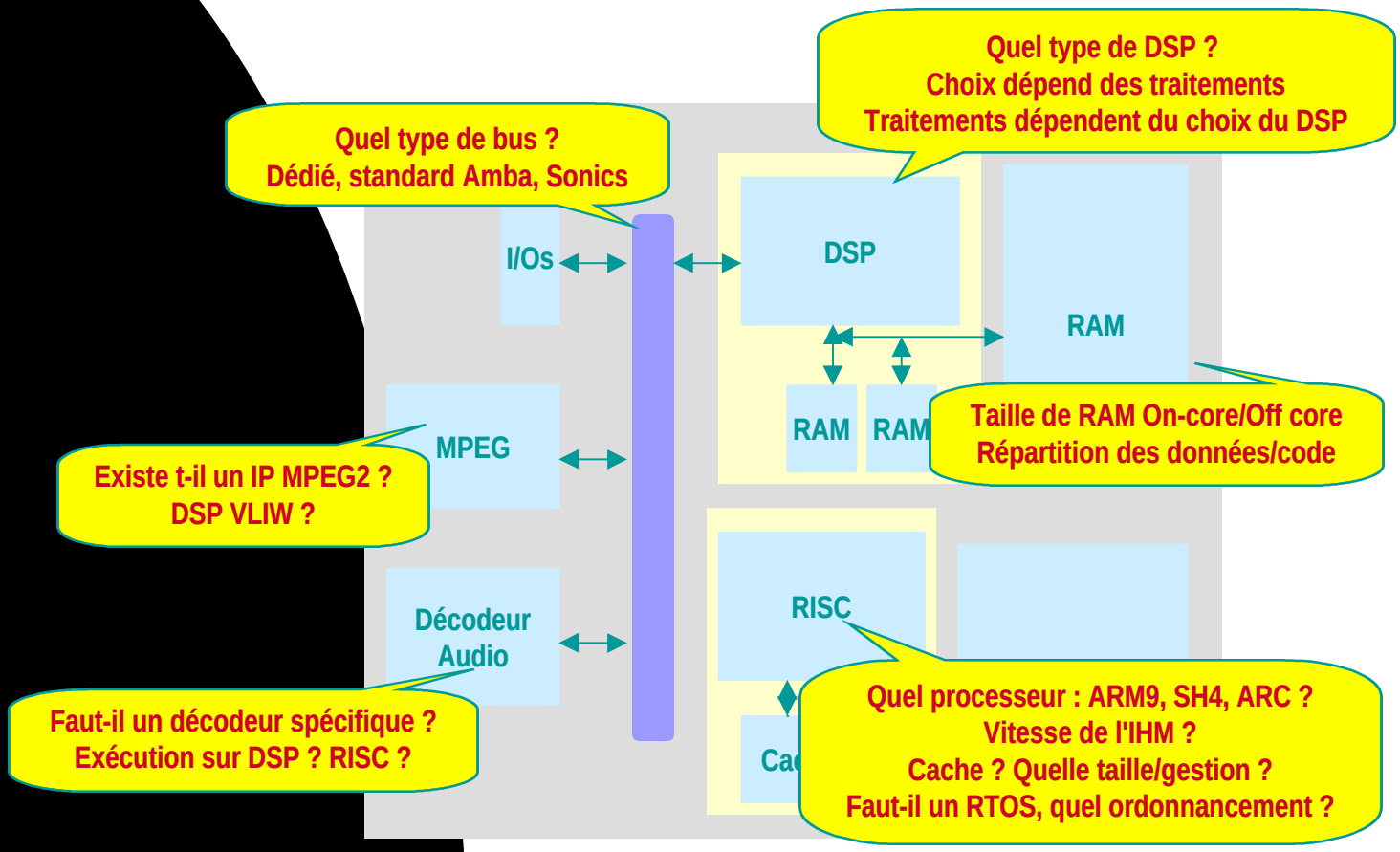
La crise de la conception vs technologie

- Le process technologique autorise un accroissement de la complexité de 59% par an
- L'efficacité des concepteurs n'augmente "que de" 25% par an



- Les outils de Codesign sont nécessaires pour permettre une exploration rapide de plusieurs architectures (RISC, FPGA, ASIC, IP, DSP, ASSP, Analogique …)

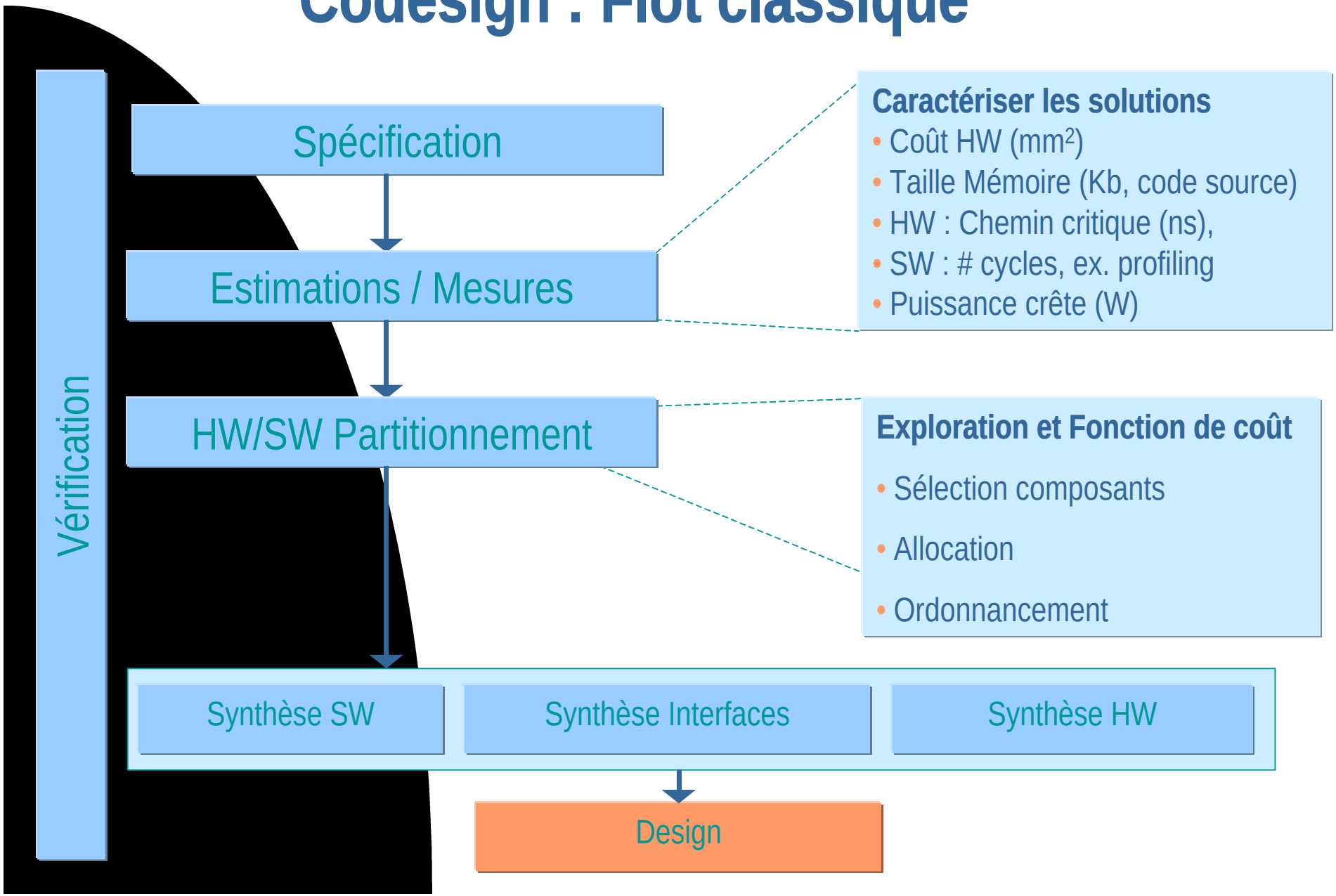
Quelle architecture, Quelles caractéristiques ?



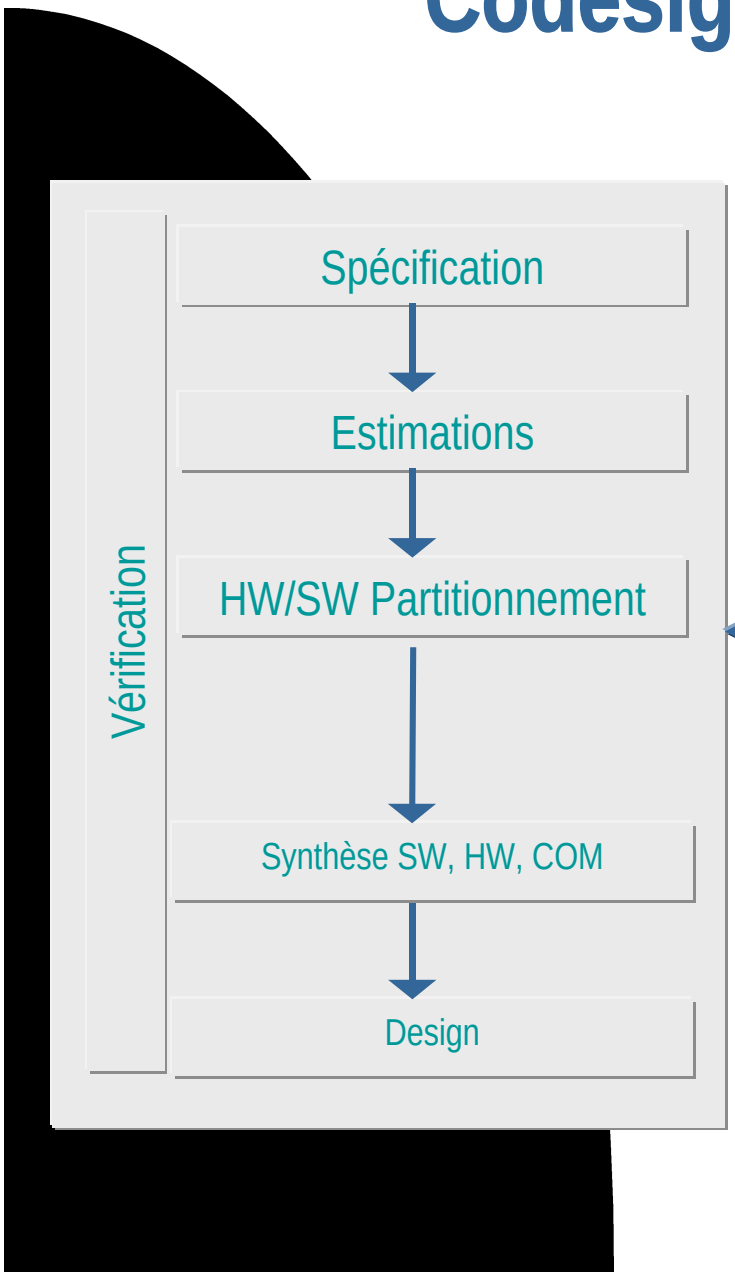
- Le Codesign vise à apporter ces réponses

Flôt de conception général

Codesign : Flot classique



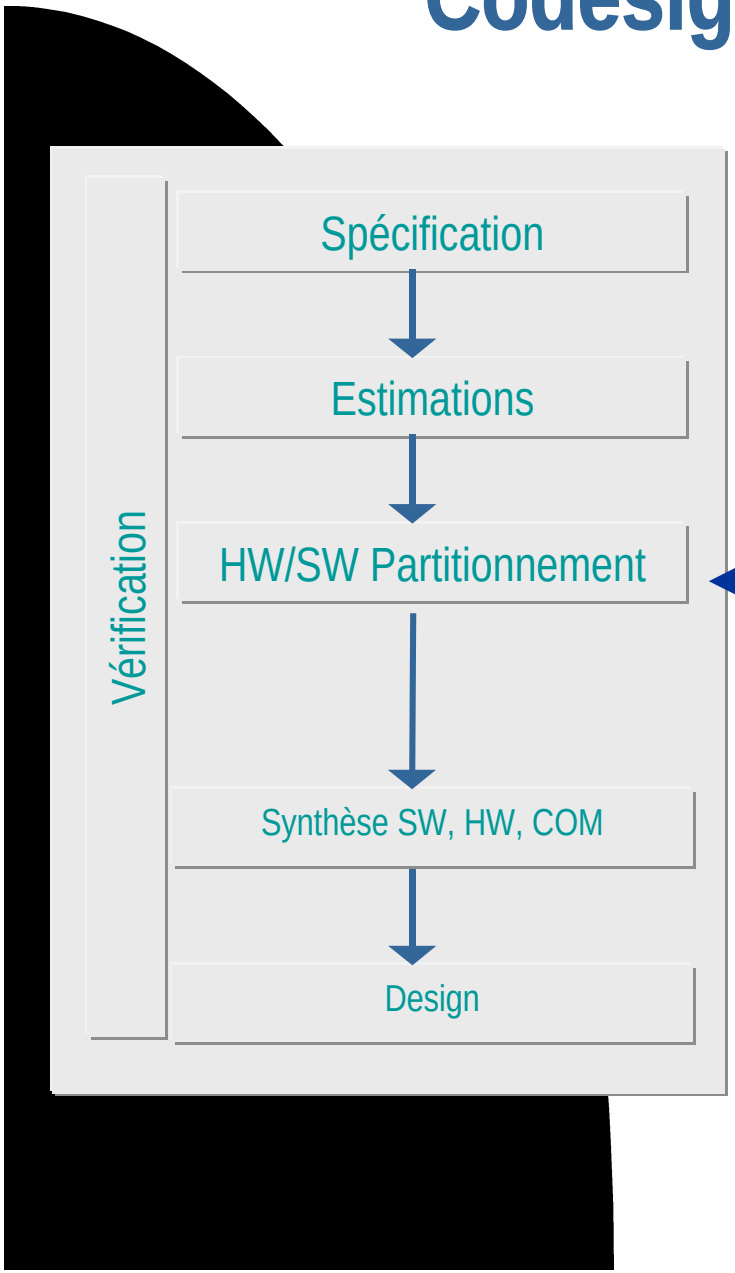
Codesign : Flot classique



Contraintes :

- Consommation max
  - ex: 1mw; ...
- Surface disponible
  - ex. 100mm2, 10.000 clbs; ...
- Débit / Délai / Fréquence
  - 25 images/s ; 10Mb ; …
- Mais aussi :
  - le rapport signal sur bruit
  - ...

Codesign : Flot classique



Modèles d'architectures (HW platform):

- Processeur (Cores : ARM7-9, StrongARM, Léon,…)
- DSP (Cores: IP 3DSP: SP-5, DSPgroup: {oak,palm},...)
- Modules HW (ASICs) (IPs ...)
- Matrices Reconfigurables (FPGAs) (IPs …)
- Mémoires
  - Flash, Caches, Mémoire DRAM, SRAM,
  - On/off-chip, On-off core
  - # Banques, # Ports
- Bus de communication
  - Amba, Sonicx, Ipbus, propriétaire,

Modélisation des applications

Modélisation des applications

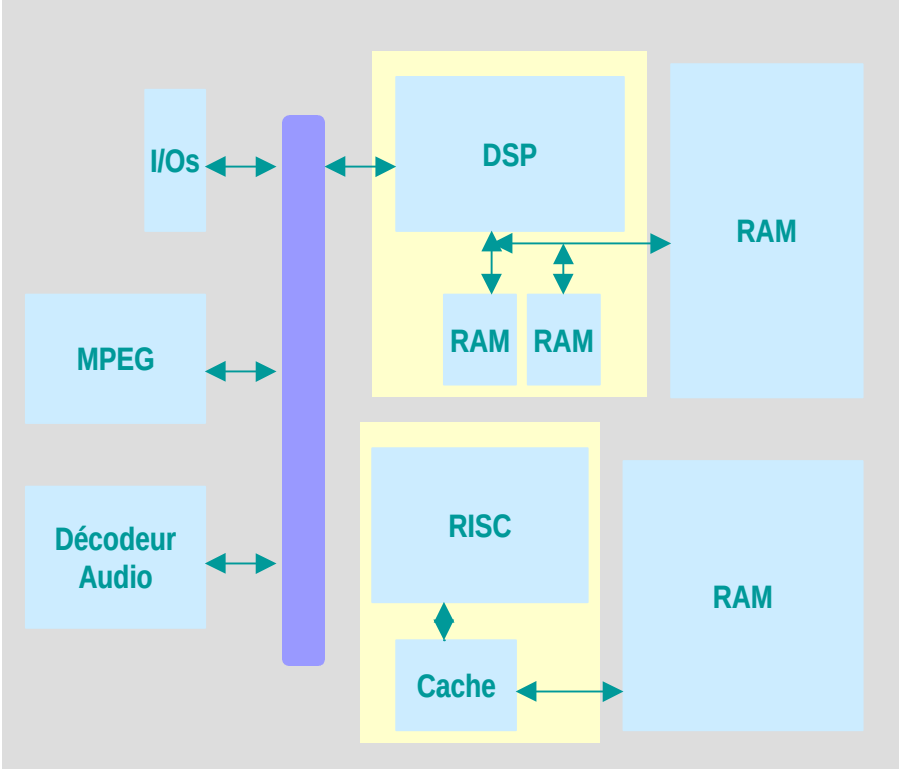
Application

If (x=0) then
$y=a \times b/2$
$z=u \times e^{jwt}$

Comment spécifier une application ?

- Méthodes
- Modèles
- Langages

Conception



Méthodes

- Une méthode conduit à exprimer une spécification selon une démarche précise.
- La méthode se caractérise par un modèle de réflexion exprimant comment procéder pour spécifier ou concevoir.
- Exemples de méthode : SADT, MCSE

- Une méthode peut nécessiter l'utilisation de un ou plusieurs modèles (FSM, SDF, SR, DE …).

- Difficile à formaliser, aujourd'hui même si il existe des méthodes elles sont peu employées (dommage la conception y gagnerait en efficacité !), problème de culture ?

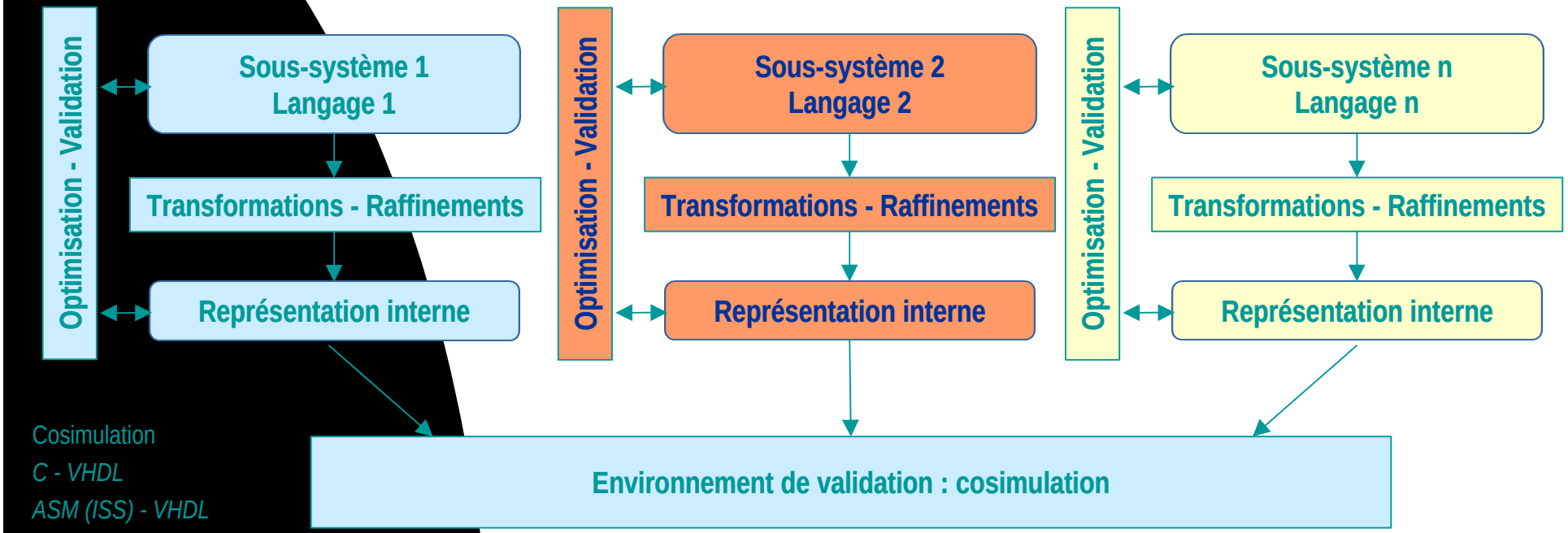
Langages et Modèles

Les applications sont hétérogènes. Comment les prendre en compte ?

- Approche "langage"
- Approche "modèle"

Approche "langage"

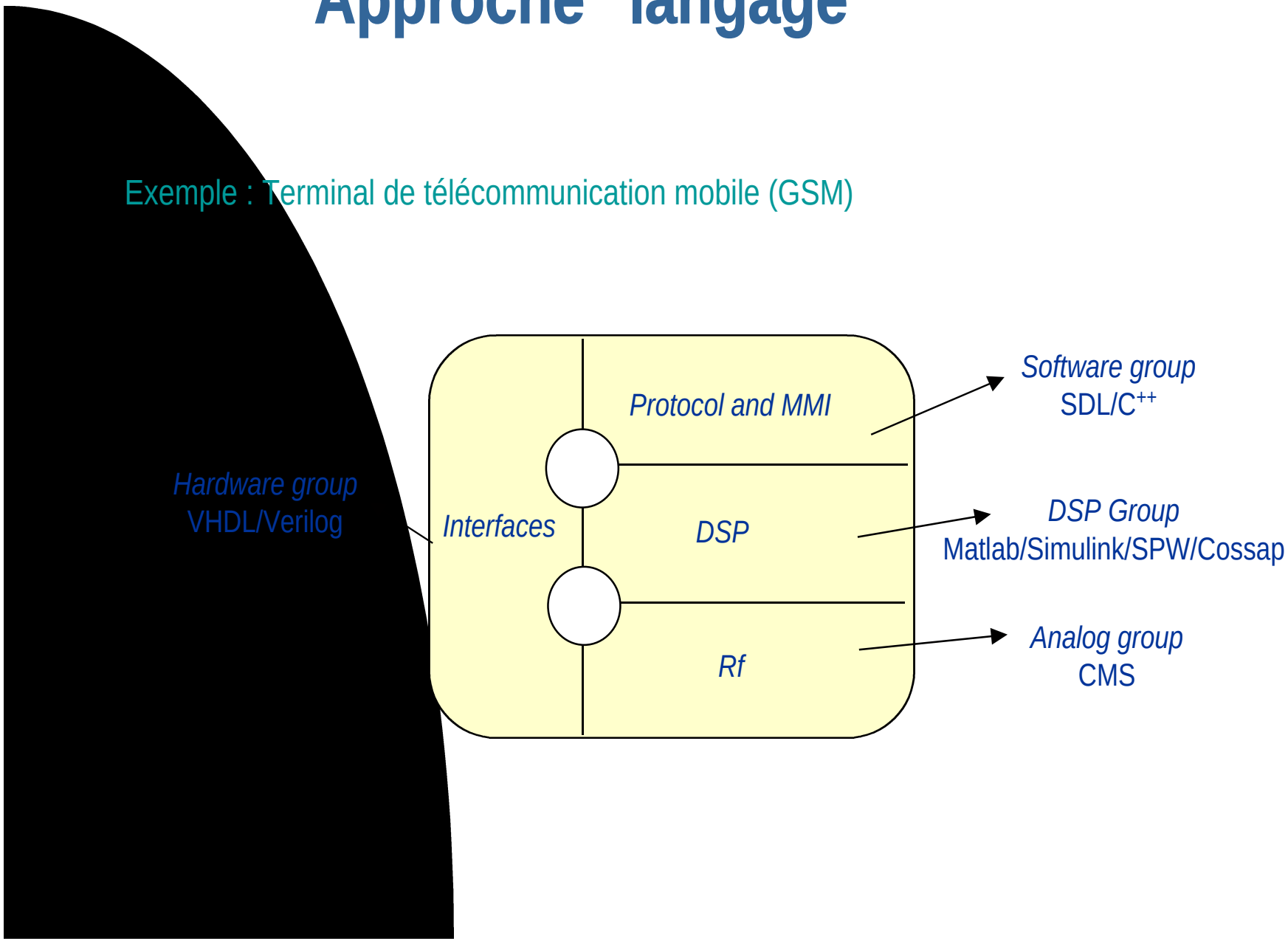
La spécification est décomposée en sous-systèmes (manuellement)



- Aucune optimisation globale : optimisation par sous-système (dépendant langage)
- Nécessite de décrire précisément les communications et les interfaces
- Les migrations éventuelles entre sous-systèmes ne sont pas aisées

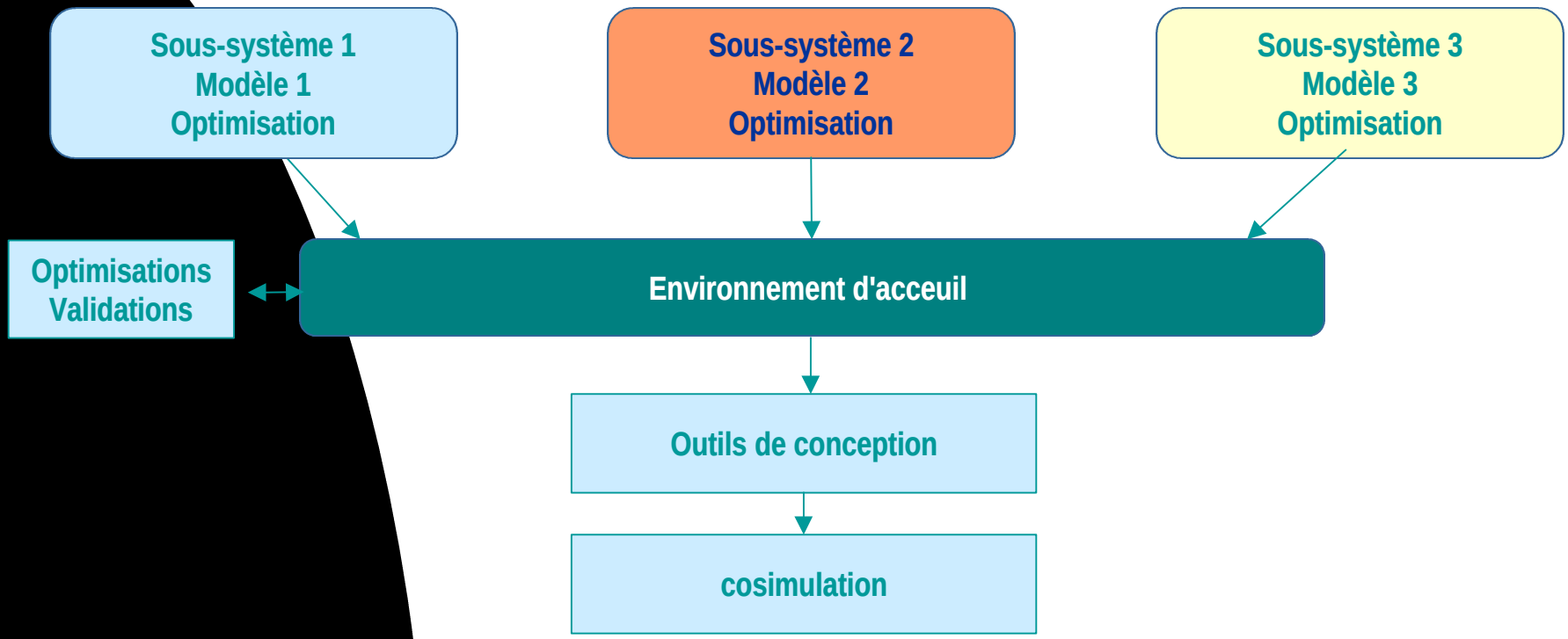
Approche "langage"

Exemple : Terminal de télécommunication mobile (GSM)



Approche "modèle"

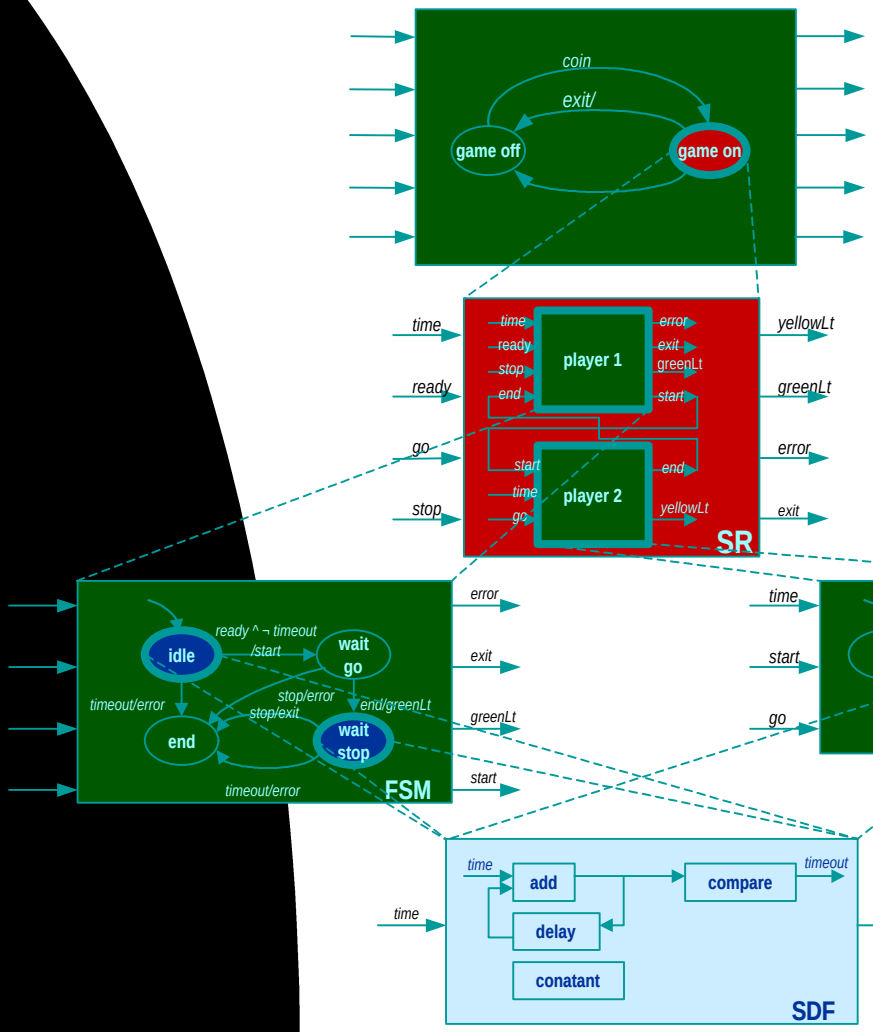
La spécification est décomposée en sous-systèmes (manuellement)



- Optimisations locales et globales
- Exploration de l'espace de conception plus aisée
- Les communications et les interfaces sont décrites de manière plus abstraite

Approche "modèle"

Exemple : *charts (Ptolemy - UC Berkeley)



Modélisation d'une application hétérogène

- Ce choix est guidé par plusieurs considérations :
  - Type de traitements dans l'application
  - Disponibilité et efficacité des outils d'analyse et de synthèse
    - **Preuve formelle de propriétés**
    - **Synthèse/compilation**
  - Possibilité de décrire des spécifications non-fonctionnelles (contraintes)

Choix du modèle de spécification en fonction du type de l'application

- Classiquement les concepteurs considèrent qu'un système est dominé :
- Par le contrôle
  - Réactions à des événements discrets
  - Pas d'hypothèse sur l'occurrence des événements
  - Tous les événements doivent être pris en compte
  - Contrôle/commande de processus

  - **Systèmes à événements discrets**
  - **Systèmes réactifs synchrones**
  - **Machine d'états finis**
  - **Réseaux de Petri**
  - **Processus concurrent et communicants (Graphe de tâches)**

- Par les données
  - Les sorties sont fonctionnellement dépendantes des entrées
  - Les occurrences des valeurs sur les entrées sont périodiques
  - Traitements intensifs
  - Traitement du signal et des images

  - **Kahn Process Networks**
  - **Dataflow Process Networks**

Langages associés aux modèles

- Systèmes réactifs synchrones
  - è Esterel, Lustre, Signal, ECL
- Systèmes à événements discrets
  - è VHDL, Verilog

✦ **Codesign Finite State Machine**

- Machine d'états finis
  - è StateCharts, SynCharts, …
- Processus concurrents
  - è CSP, Occam, Lotos

✦ **SDL (Standard de l'ITU)**

- Dataflow Process Networks
  Kahn Process Networks
  - è Dynamic Data Flow
    Synchronous Data Flow
- Approche objets
  - è Java, C++, OO-VHDL

- Exemple d'outils associés :
  - VCC de Cadence : asynchronisme des CFSM, C, C++, ECL
  - CoCentric de Synopsys : SystemC
  - Coware : SystemC

Les modèles d'exécution

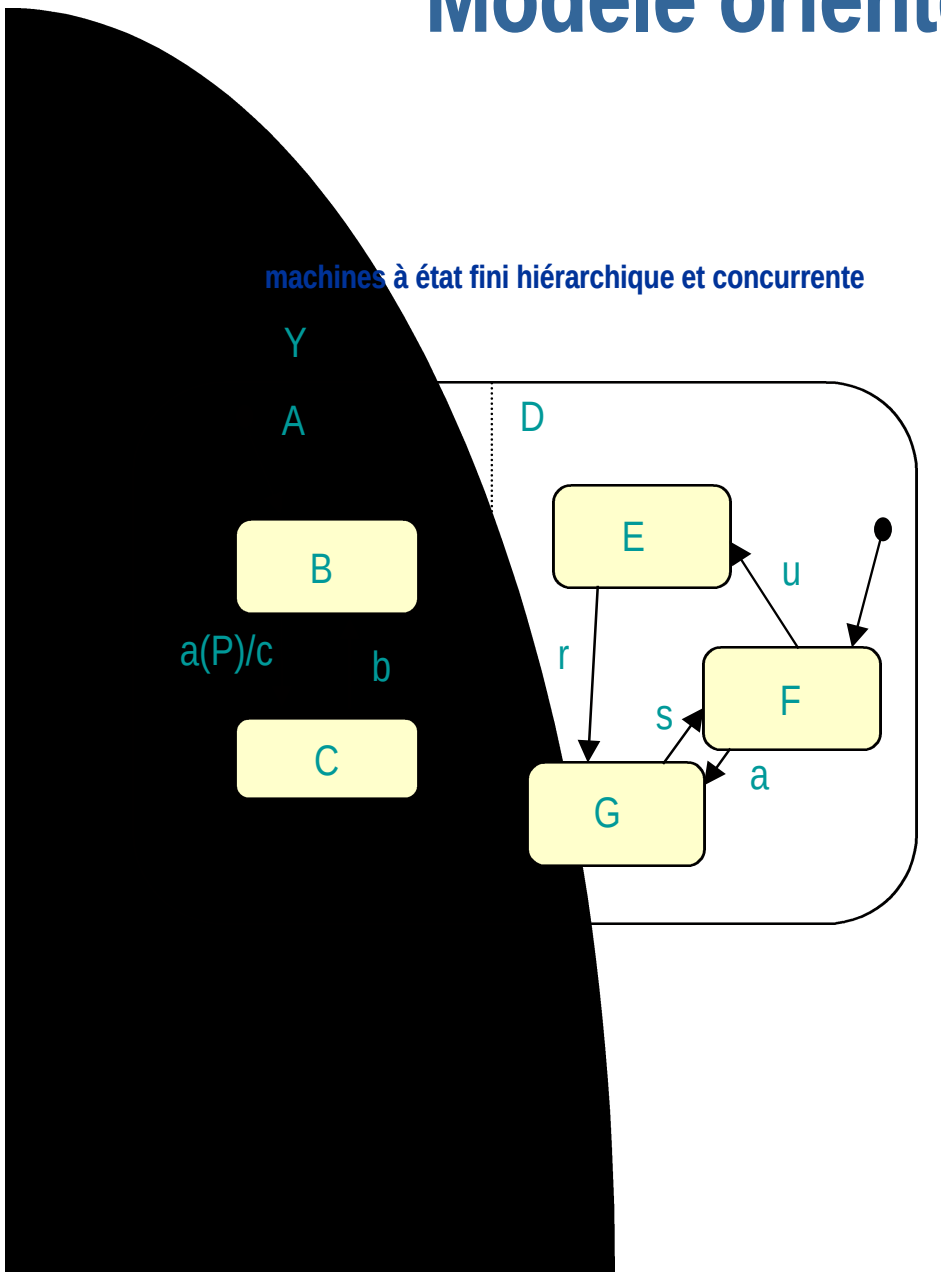
- Il existe différents types de modèles d'exécution :

  FSM, réseau de)

  - Modèles orientés traitement (e.g. graphe flot de données)
  - Modèles hétérogènes (e.g. : control flowgraph)

Modèle orienté contrôle :

machines à état fini hiérarchique et concurrente

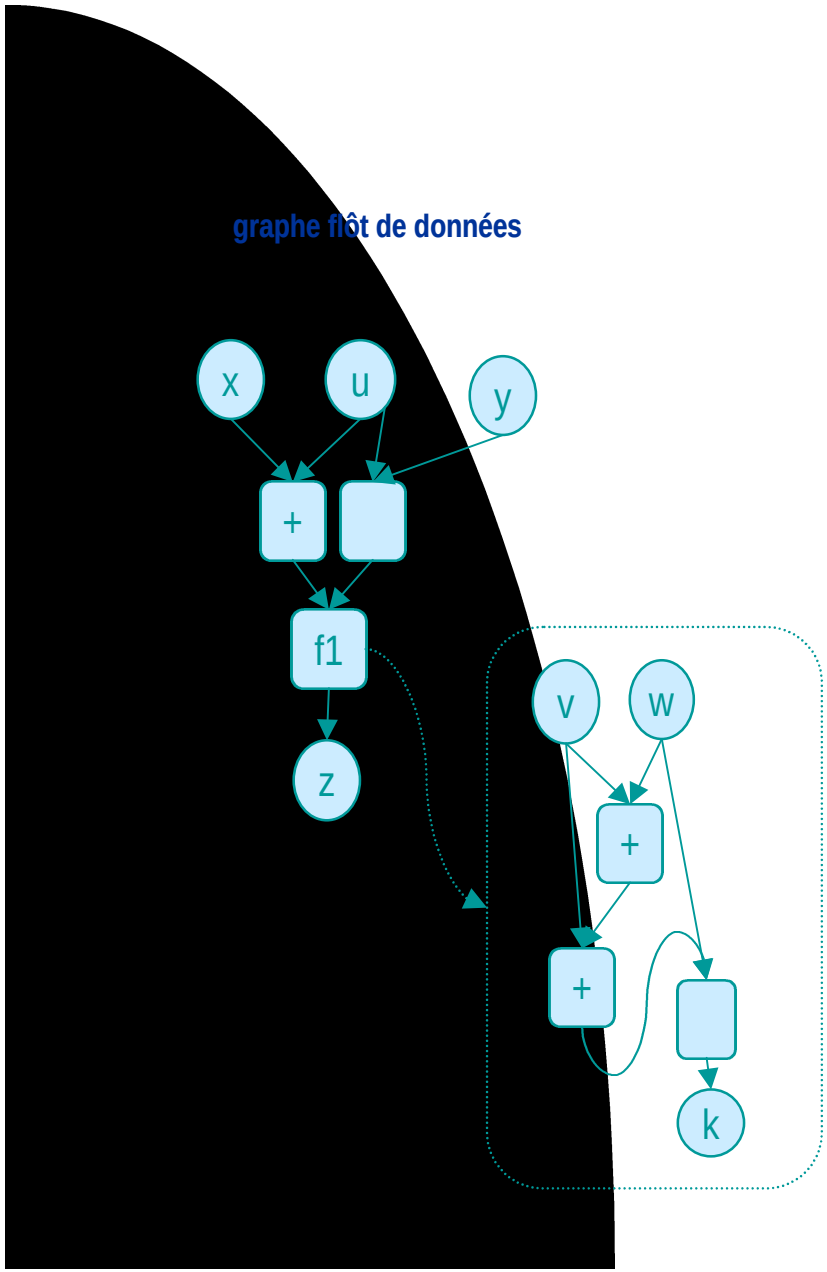


Avantage

F **Bien adapté pour la modélisation :**
- **Concurrence,**
- **Systèmes complexes**

Inconvénient

F **Se limite uniquement à la modélisation des structures de contrôle.**

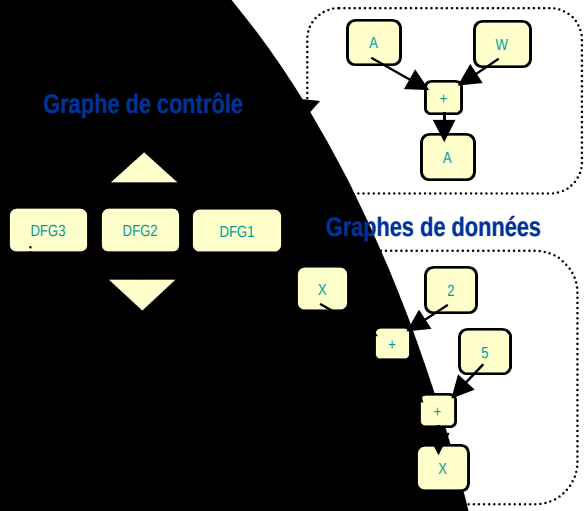


- Permet de modéliser la hiérarchie.
-

- Se limite uniquement à la modélisation des dépendances de données.
- Pas de modélisation des comportements temporels ou des structures de contrôle.

Modèles hétérogènes

graphe de contrôle et de données - CDFG



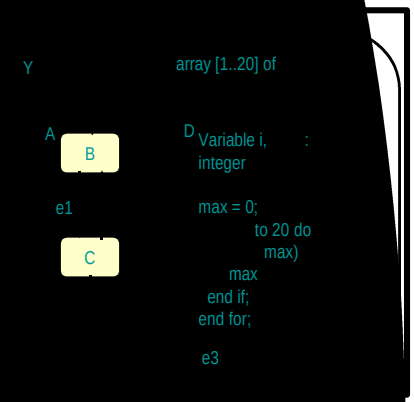
Avantage

- Représentation des structures de contrôle.
- Représentation des dépendances de données

Inconvénient

- Pas de concurrence.
- Pas d 'interruption.

Traitement et machine de contrôle - FSMD



Avantage

- Représentation des structures de contrôle.
-
- Représentation des différents états du système.

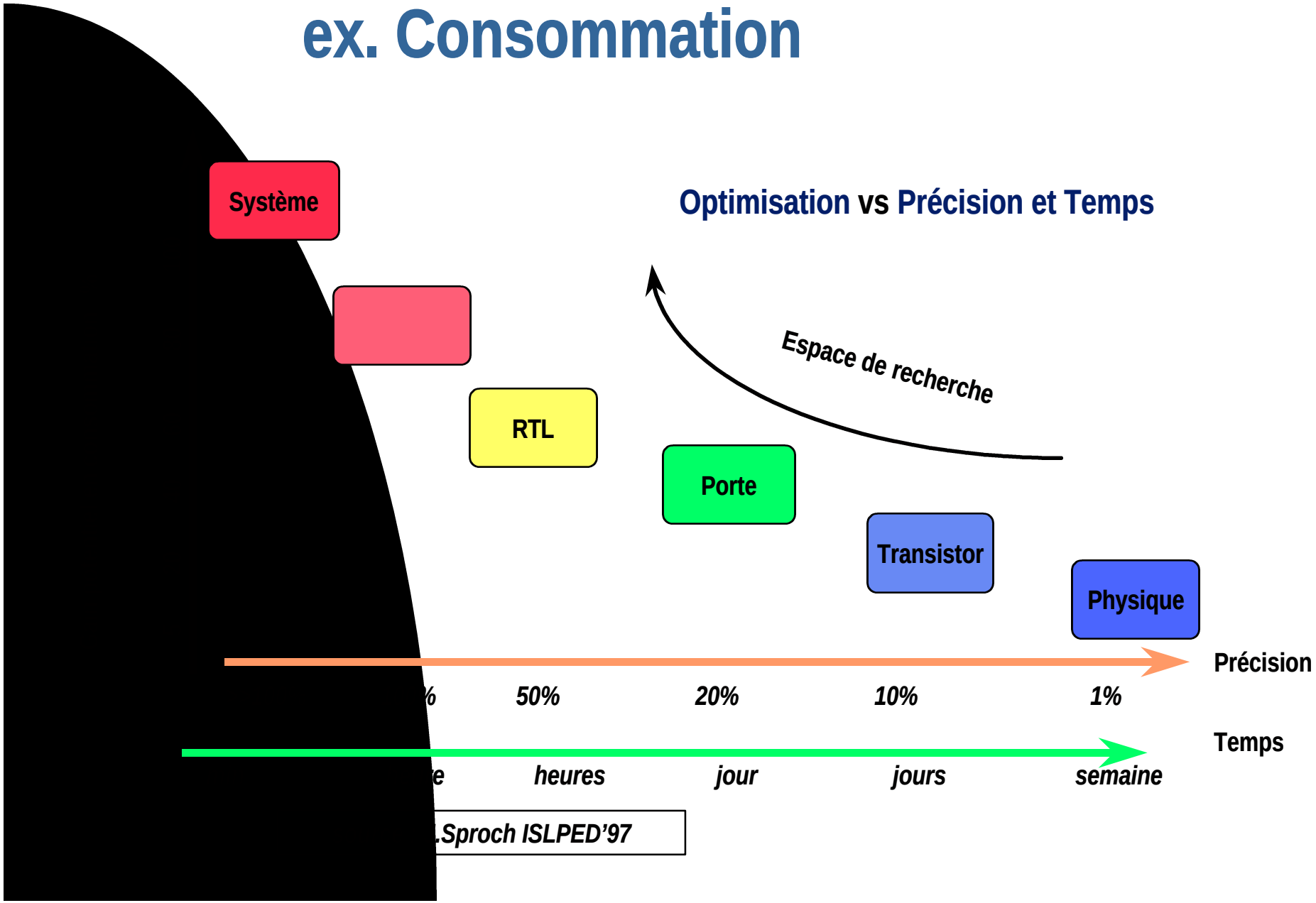
- Limité au niveau des mécanismes de communication entre process

Estimation logiciel, matériel, système

Estimation : Problématique

- Choix Estimation VS Mesures
  - Rapidité
  - Borne minimum ou maximum
  - Indépendant du compilateur / outil de la synthèse ➡ d'où guidage possible
- Compromis : Niveau d'Abstraction et Précision
  - haut niveau (algo) : rapide mais peu précis
  - bas niveau (porte ou instruction) : précis mais trop complexe
  - solution : affinages successifs
- Dépendance vis à vis des données :
  - Ex. Compression d'images MPEG4 : charge de transferts et de calculs proportionnelle aux mouvements et nombre d 'objets
  - PB : Trouver les vecteurs de test pertinents
  - Solution du pire cas : sûr mais grave surdimensionnement possible
  - Solution du cas moyen : efficace mais les contraintes ne sont pas garanties

Estimation : Problématique
ex. Consommation



Estimation logiciel

Estimation Logiciel

- Données recherchées :
  - Performance
  - Consommation

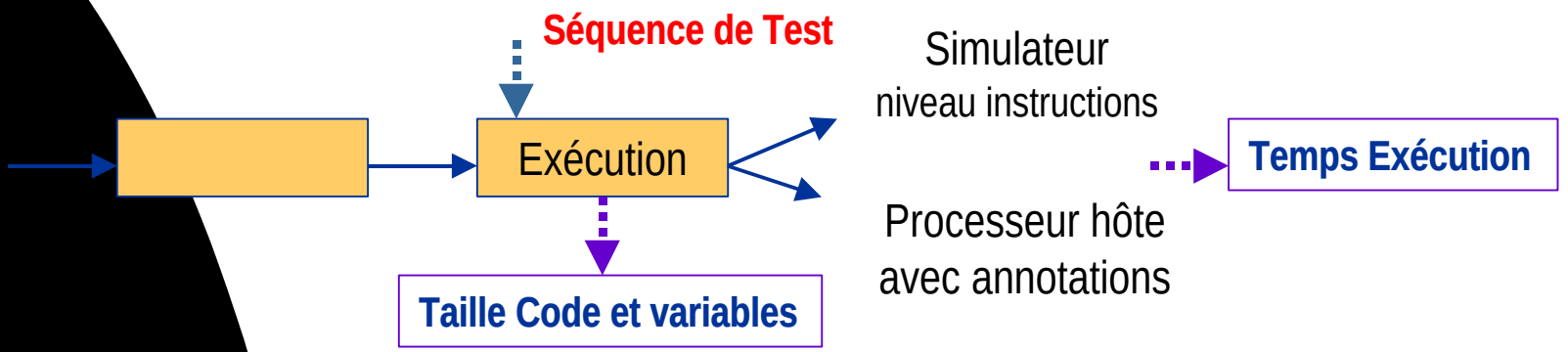
    vs SW mais aussi SW1 SW2 vs
  - Ex. Nombre important de cores
  - Paramétrage des mémoires On-core

  Profiling
  - Estimation du Nb
- Deux Types de méthodes
  - Mesure

    Estimation analytique : Reposant sur un modèle recyblable

Estimation Logiciel (1)

Estimation par compilation / exécution-simulation



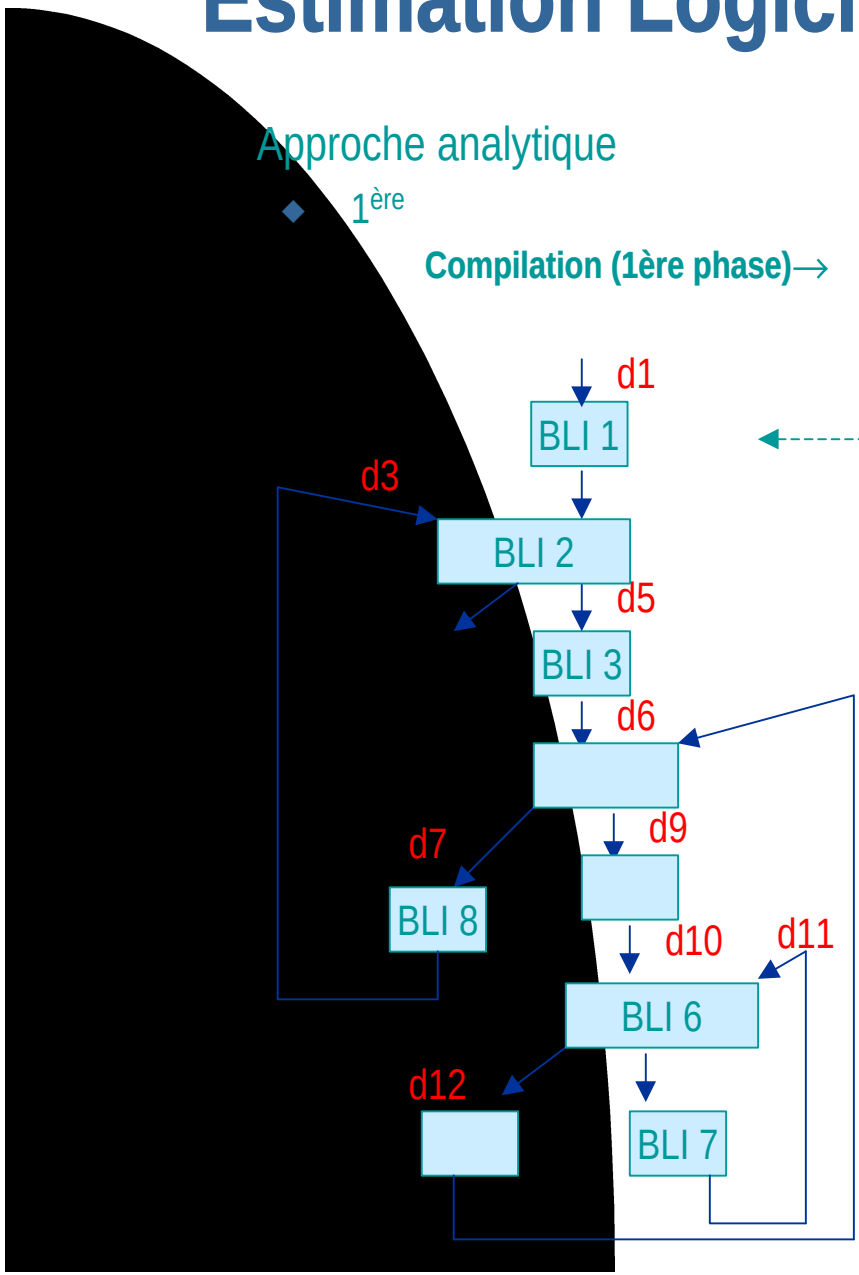
- *dependent*
- **Code à la main faisable plus compact et performant que la version automatique**
- Simulation longues

Estimation Logiciel

Approche analytique

- 1ère

Compilation (1ère phase)→ **→ CDFG**



Bloc Linéaire d'Instruction (ou Data Flot Graph) caractérisé par :

•

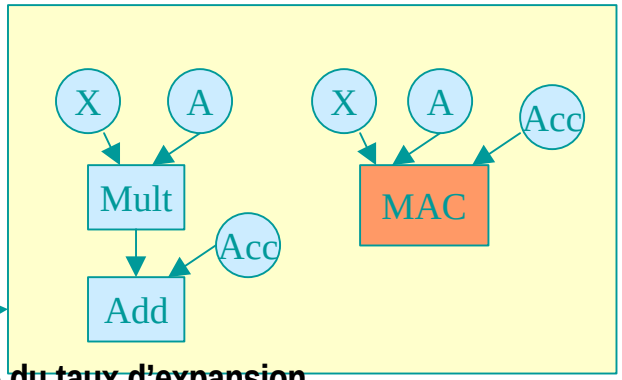
Xi : Nb de passage dans le BLI i

Les arc sont valués par la quantité de données

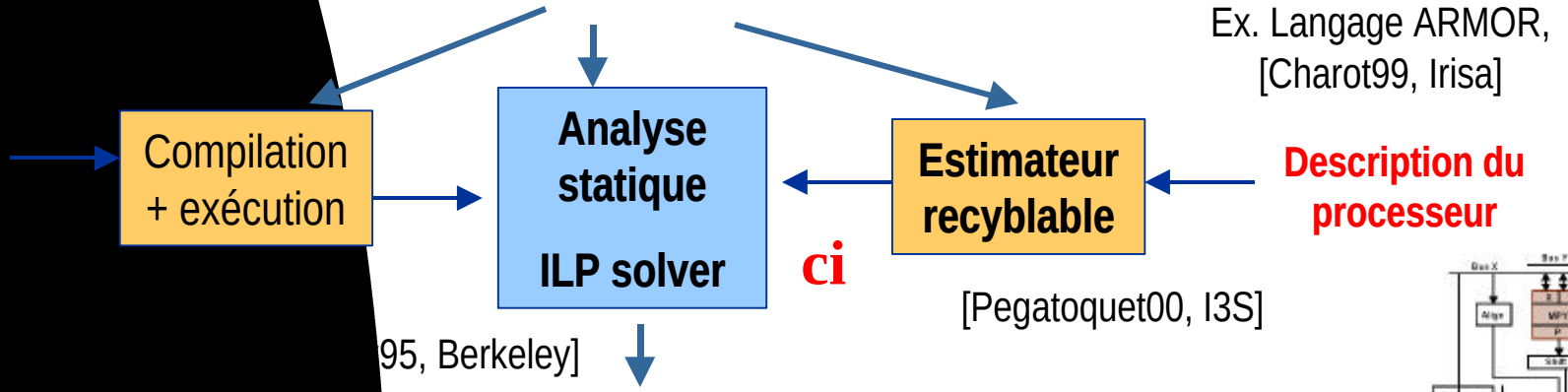
Ex. FFT [Malik95, Berkeley]

Estimation Logiciel (3)

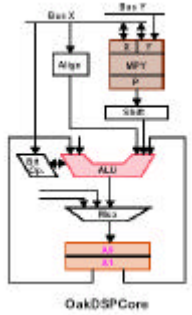
- Approche analytique *Pegatoquet Auguin*
  - 2ème *Profiling*
  - 3 é é
  - 4 é *pattern matching*
  - 5 é
  - 6ème étape : Ordonnancement [ex.]



Spilling ↗ temps et ≈ 30% du taux d'expansion

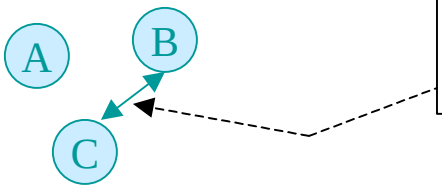


Ex. Langage ARMOR,
[Charot99, Irisa]



(4)

- Autres points :
  - **[Malik95]**



A et B ne peuvent être dans la même ligne de cache car appartiennent à deux BLI différents [Malik95]

    - ***Acropolis*** *Data transfer and storage issues in multimedia processors* ***[Cathoor00, Acropolis, Imec]***
  - Estimation de consommation
    - **A partir d'un modèle du processeur** ***[Laurent, Julien 02, LESTER]***
    - **Problème majeur (consommation) : prise en compte des cache miss …**
  - Prise en compte des *overheads* dus au RTOS
    - **travaux en cours …**

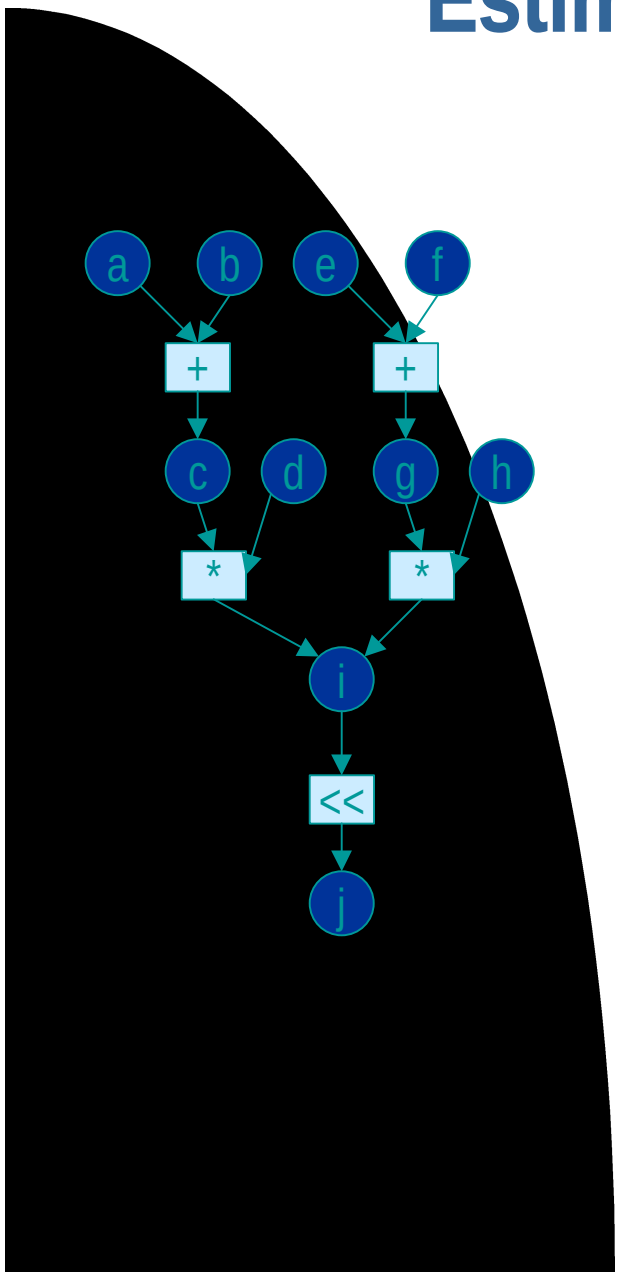
Estimation matériel

Estimation matériel

- Plusieurs paramètres doivent être estimés.
  - Surface
  - Vitesse
  - Consommation
- Plusieurs cibles peuvent être considérées
  - ASIC
  - FPGA
- Plusieurs modèles sont utilisés pour spécifer les applications
  - DFG
  - CDFG
  - "FSM"

- Il est nécessaire de définir un modèle d'architecture cible
  - Registre généraux, Bus généraux
  - Contrôle : ROM, Logique
  - Générateur d'adresse
  - Mémoire : Double port, simple port…
- L'architecture est-elle entièrement estimée ?
  - Chemin de données
  - Contrôleur
  - Mémoire
  - Communication

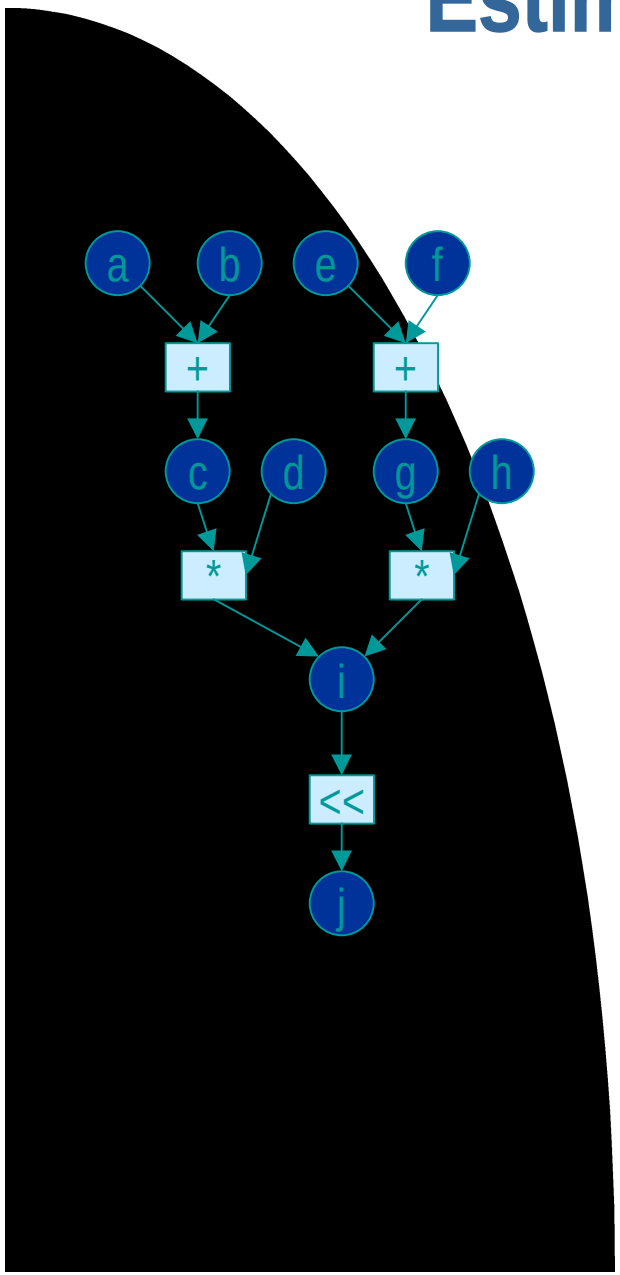
Estimation des ressources



- La contrainte de temps est fixée.
- Il faut estimer le nombre d'unités fonctionnelles (ressources) qui permettent d'exécuter l'application tout en respectant la contrainte de temps.

- Exemple : estimation des ressources
  - Texe = 3
    - **2 additionneurs**
    - **2 multiplieurs**
    - **1 registre à décalage**
  - Texe = 4
    - **1 additionneur**
    - **1 multiplieur**
    - **1 registre à décalage**

Estimation des performances



- Les unités fonctionnelles sont fixes.
- Il faut estimer le temps d'exécution optimal de l'application sur les unités fonctionnelles de l'architecture.
- Se rapproche de l'estimation système (dans un processeur les ressources sont fixées).

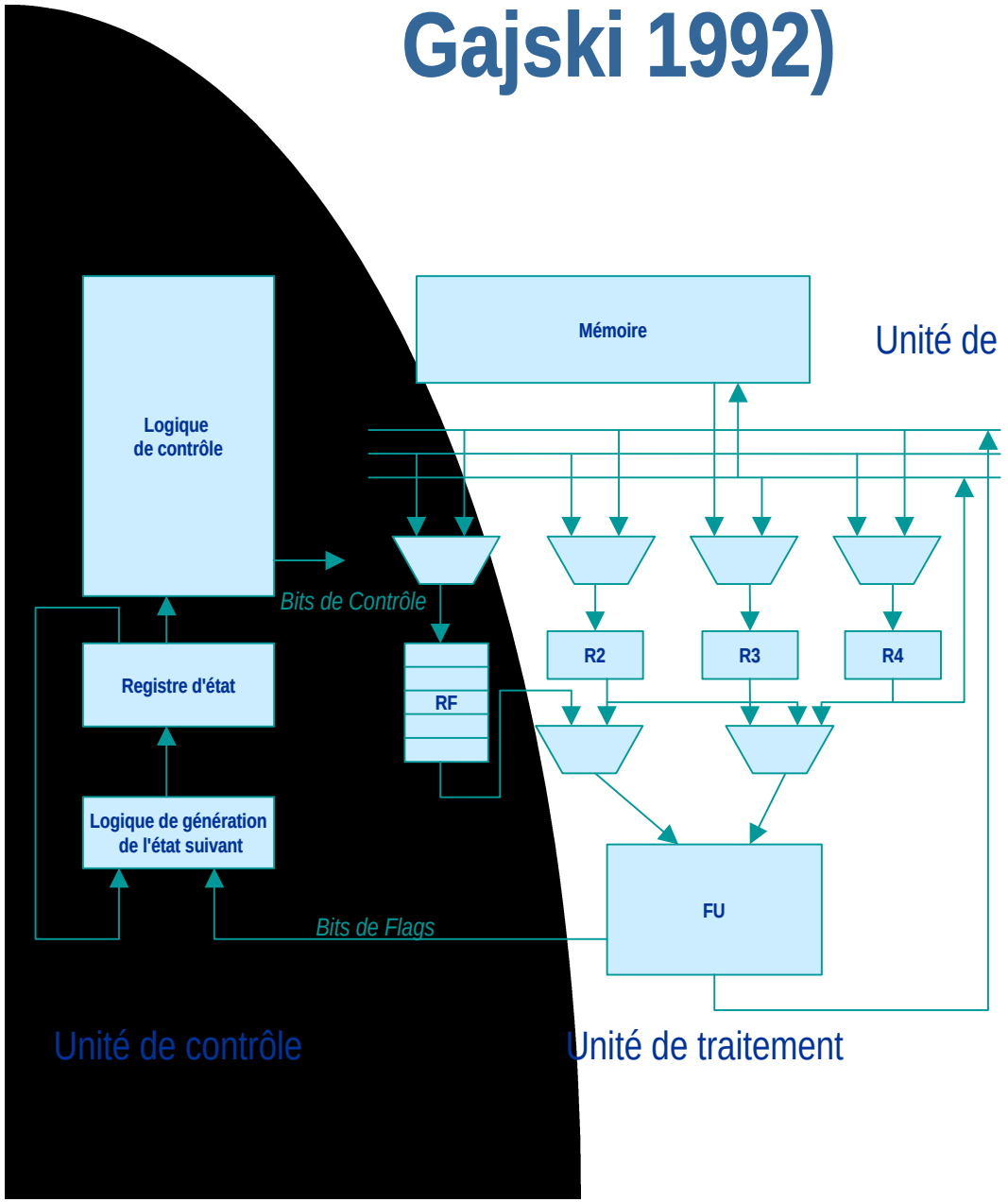
- Exemple : estimation des performances
  - 2 additionneurs
  - 1 multiplieur
  - 1 registre à décalage
    - **Texe = 4**
  - 2 additionneur
  - 2 multiplieurs
  - 1 registre à décalage
    - **Texe = 3**

Estimation des ASICs

- Estimation de l'unité de traitement (DFG).
  - Bornes minimale/maximale absolues des opérateurs.
  - Méthode par le minimum de recouvrement.
  - Méthode par détection des cycles perdus.
  - Méthodes probabilistes.
- Estimation de l'architecture (SpecChart)

Auteurs	*Méthode*	*Complexité*	*Architecture*	*Spécification*	*Remarque*
Rabaey et Potkonjak [Rabaey94]	Détection de cycles perdus	$< O(nc^2)$	Traitement	DFG	Borne supérieure et inférieure
Sharma et Jain [Sharma93]	Minimum de recouvrement	$O(nc^2)$	Traitement	DFG	Borne inférieure
Rim et Jain [Rim94]	Formulation ILP	O(n + c)	Traitement/contrôle	DFG	Borne inférieure du nombre de pas de contrôle
Diguet [Diguet96]	Probabilités non conditionnées	O(n)	Traitement	DFG	Caractéristique en fonction d'une contrainte de temps variable
Diguet [Diguet96]	Probabilités conditionnées	O(n)	Traitement	DFG	Caractéristique en fonction d'une contrainte de temps variable
Diguet [Diguet96]	Probabilités globales	O(n)	Traitement	DFG	Caractéristique en fonction d'une contrainte de temps variable
Narayan et Gajski [Narayan92]	Analyse BLI + Fréquence CFG		Traitement/Contrôle/Mémoire	SpecCharts	Prise en compte du contrôle et de la hiérarchie

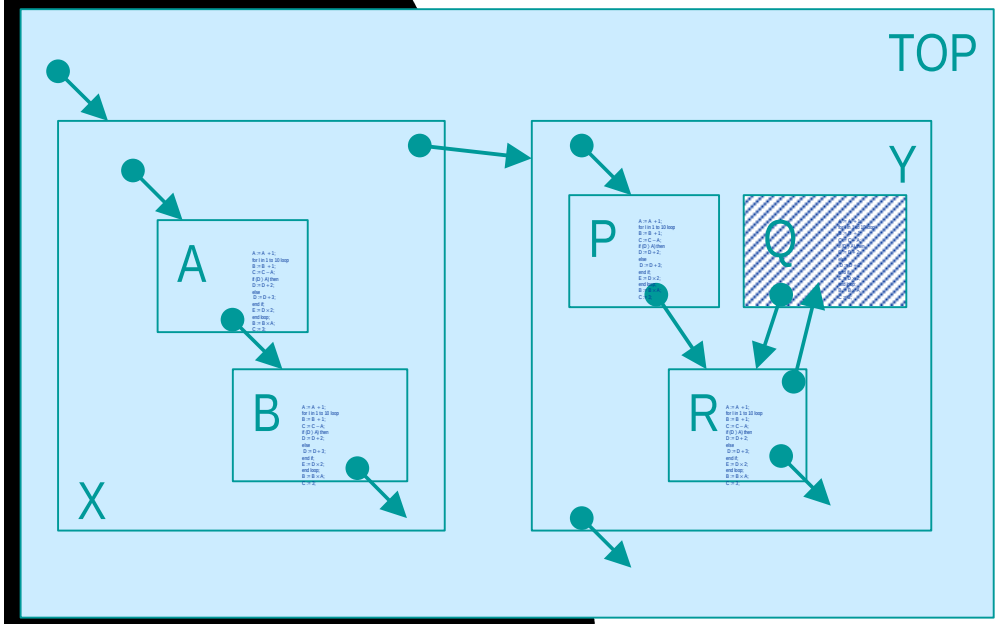
Estimation des ASICs (Narayan & Gajski 1992)



- Estimation de performances et de surface pour ASIC.
- Estimation des unités de traitement, mémorisation et contrôle.

- Modélisation de l'application avec SpecChart (diagramme d'états concurrents/hiérarchique).
- Librairie de ressources disponibles et temps d'exécution.
- Fréquence d'horloge fixée.
- Modèle d'architecture.

Estimation de performances



- Modélisation de l'application avec SpecChart (diagramme d'états concurrents/hiérarchique).
- Pour chaque blocs de base (DFG) d'un état de la modélisation, calculer :

Le nombre de micro-états nécessaires : ExecTime(b)

La fréquence moyenne d'exécution : Freq(b)

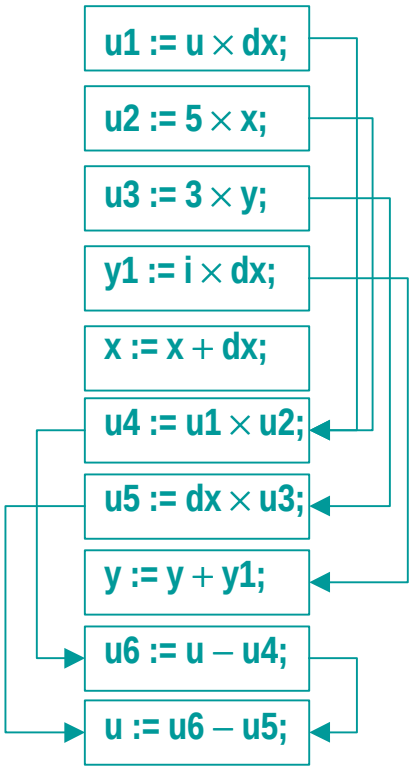
- Pour chaque état calculer le temps d'exécution

$$\text{ExecTime(Etat)} = \sum_{\text{pour tous les blocs de base}} [\text{ExecTime(b)} \times \text{Freq(b)}]$$

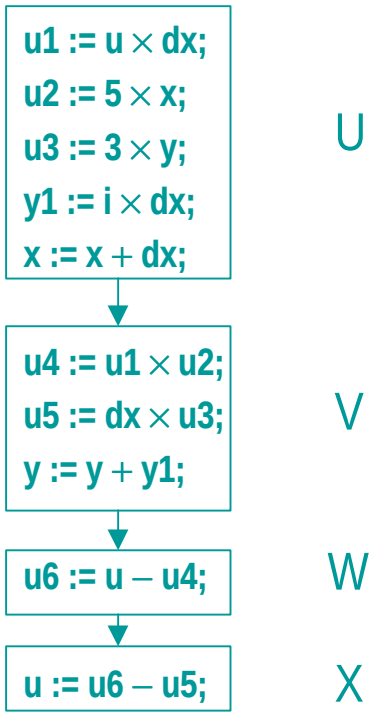
Estimation d'un bloc de base (1)

```
u1 := u × dx;
u2 := 5 × x;
u3 := 3 × y;
y1 := i × dx;
x := x + dx;
u4 := u1 × u2;
u5 := dx × u3;
y := y + y1;
u6 := u – u4;
u := u6 – u5;
```

Soit le bloc de base suivant



Chaque instruction est associée à un nœud différent
Les arcs représentent les dépendences d'exécution



Décomposition en macro-noeuds

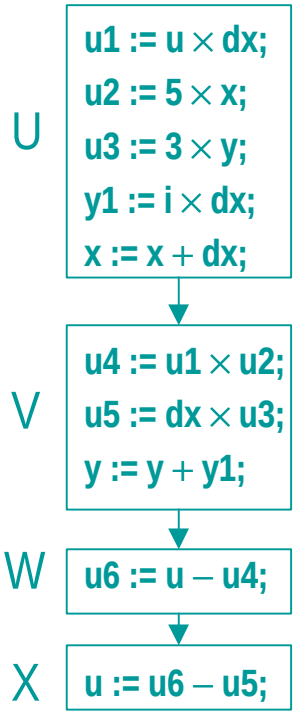
Estimation d'un bloc de base (2)

n(+) = 1
ClkDelay(+) = 1

n(×) = 2
ClkDelay(×) = 2

n(–) = 1
ClkDelay(–) = 1

Unités fonctionnelles allouées
et temps d'exécution

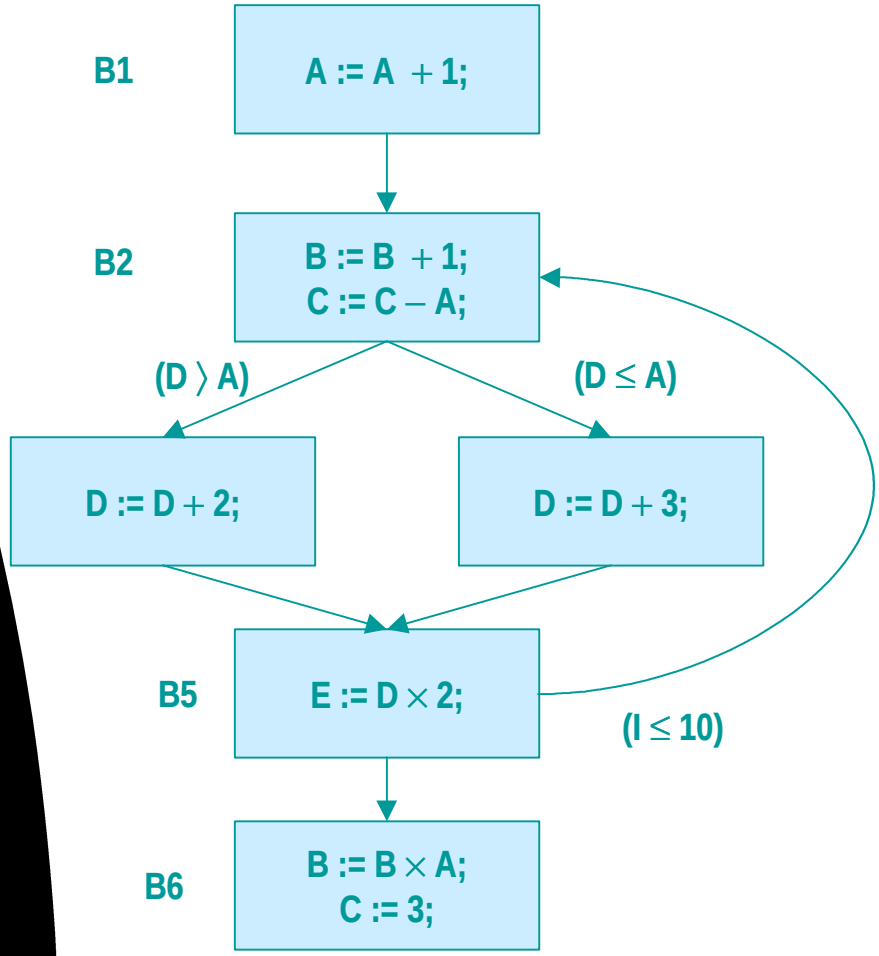


	micro-états des macro-nœuds
u(+) = 1 => [(u(+) * n(+)) / ClkDelay(+)] = 1 u(×) = 4 => [(u(×) * n(×)) / ClkDelay(×)] = 4	max = 4
u(+) = 1 => [(u(+) * n(+)) / ClkDelay(+)] = 1 u(×) = 2 => [(u(×) * n(×)) / ClkDelay(×)] = 2	max = 2
u(–) = 1 => [(u(–) * n(–)) / ClkDelay(–)] = 1	1
u(–) = 1 => [(u(–) * n(–)) / ClkDelay(–)] = 1	1
Nombre total de micro-états, ExecTime(b) =	8

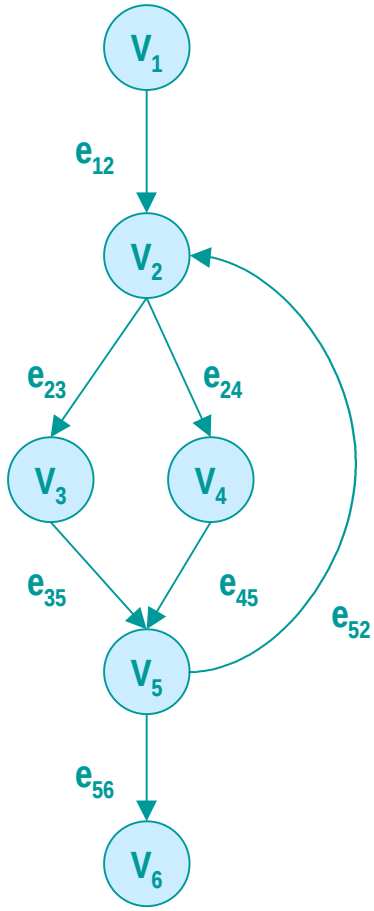
Estimation d'un état : programme VHDL (1)

```
A := A + 1;
for I in 1 to 10 loop
        B := B + 1;
        C := C – A;
        if (D ⟩ A) then
                D := D + 2;
        else
                D := D + 3;
        end if;
        E := D × 2;
end loop;
B := B × A;
C := 3;
```

Fonction VHDL

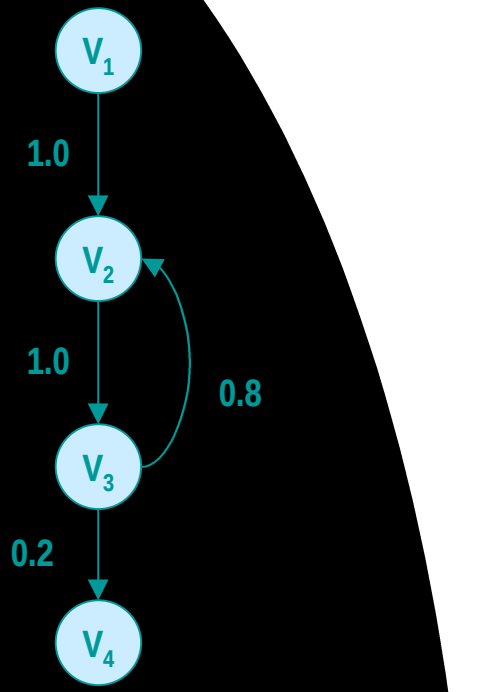


Décomposition en blocs de base

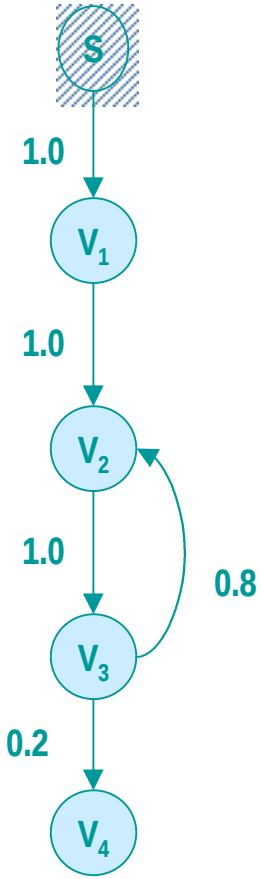


Graphe flôt de contrôle équivalent

Estimation d'un état : programme VHDL (2)



CFG enrichi avec les probabilités de branchement (*équiprobabilités, probabilités définies par le concepteur ou obtenues par profiling*)



CFG avec le nœud S pour définir la probabilité d'exécution du CFG
(utile dans le cas de CFG hiérarchiques)

$F(S) = 1.0$

$F(V_1) = 1.0 \times F(S)$

$F(V_2) = 1.0 \times F(V_1) + 0.8 \times F(V_3)$

$F(V_3) = 1.0 \times F(V_2)$

$F(V_4) = 0.8 \times F(V_3)$

Solution :

- $F(V_1) = 1$
- $F(V_2) = 5$
- $F(V_3) = 5$
- $F(V_4) = 1$

Equations traduisant le nombre d'exécution de chaque nœud

Estimation des FPGAs

Référence	Cible	Modèle de spécification	Description	Commentaire
Narayan & Gajski	ASIC	Modèle comportemental hiérarchique & concurrent (SpecChart)	Estimation temps/ surface, caractérisation traitement, mémoire, contrôle	Ne permet pas l'exploration automatique de plusieurs solutions
Kliman 1994	FPGA	Bibliothèque de benchmarks	L 'application est partitionnée, chaque sous partie est remplacée par un circuit benchmark	Très dépendante de la base de données
Xu & Kurdahi 1997	FPGA (XC4000)	Description au niveau logique	Modèle fin du processus de mapping	Nécessite une étape de synthèse de Haut Niveau
Inst. Féd. de Techn. Suisse 2000	FPGA	DFG	Caractérisation algorithmique (ex: nb add^{os}) + caractérisation du modèle de mapping (ex : nb bits / CLB pour l 'add^{o})=> combinaison linéaire	Modèle de spécification très limitatif

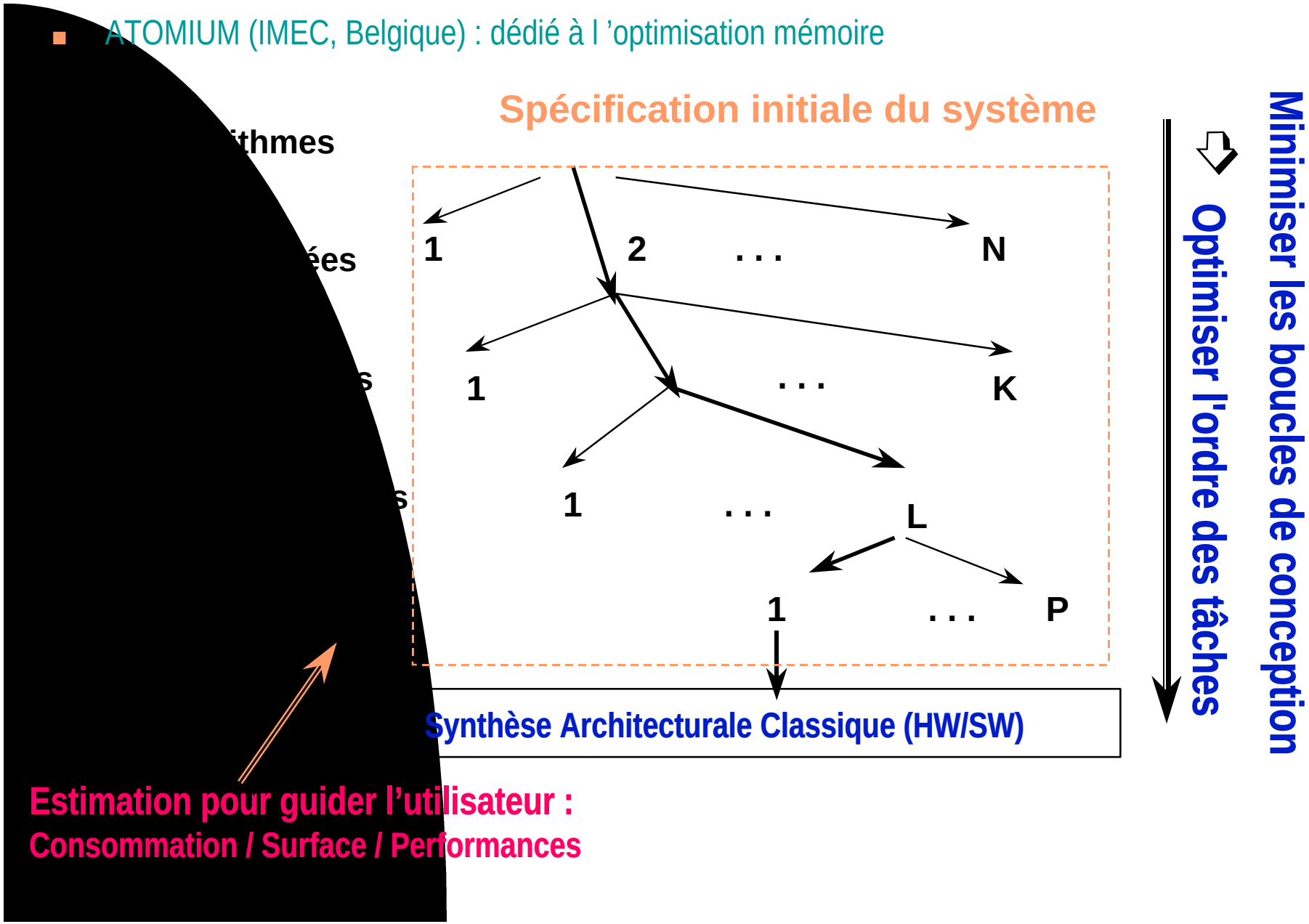
Estimation système

Estimation Système

- Niveau algorithmique
  - Avant le choix de la plate-forme matérielle
- Objectifs
  - Exploration de l'espace des solutions
  - Aide à la décision du choix de l'architecture
  - Pourquoi : l'impact des décisions % niveau d 'abstraction
- Explorer le parallélisme
  - Niveau transferts des données et traitements
  - Offrir ≠ solutions idéalement du «tout séquentiel» au «parallélisme max»
  - Ex. choix des déroulages de boucles
- Caractérisation : orientation Traitement / Contrôle / Mémoire
  - [Vahid95, Univ.Irvine] : *closeness metrics*, [Guerra94, Bekeley] : *Design Guidance,* [Diguet95, Lasti] *Probability Based Guidance*, [Thomas99, Lester] : *orientation metrics*
- Modèle d'architecture abstrait

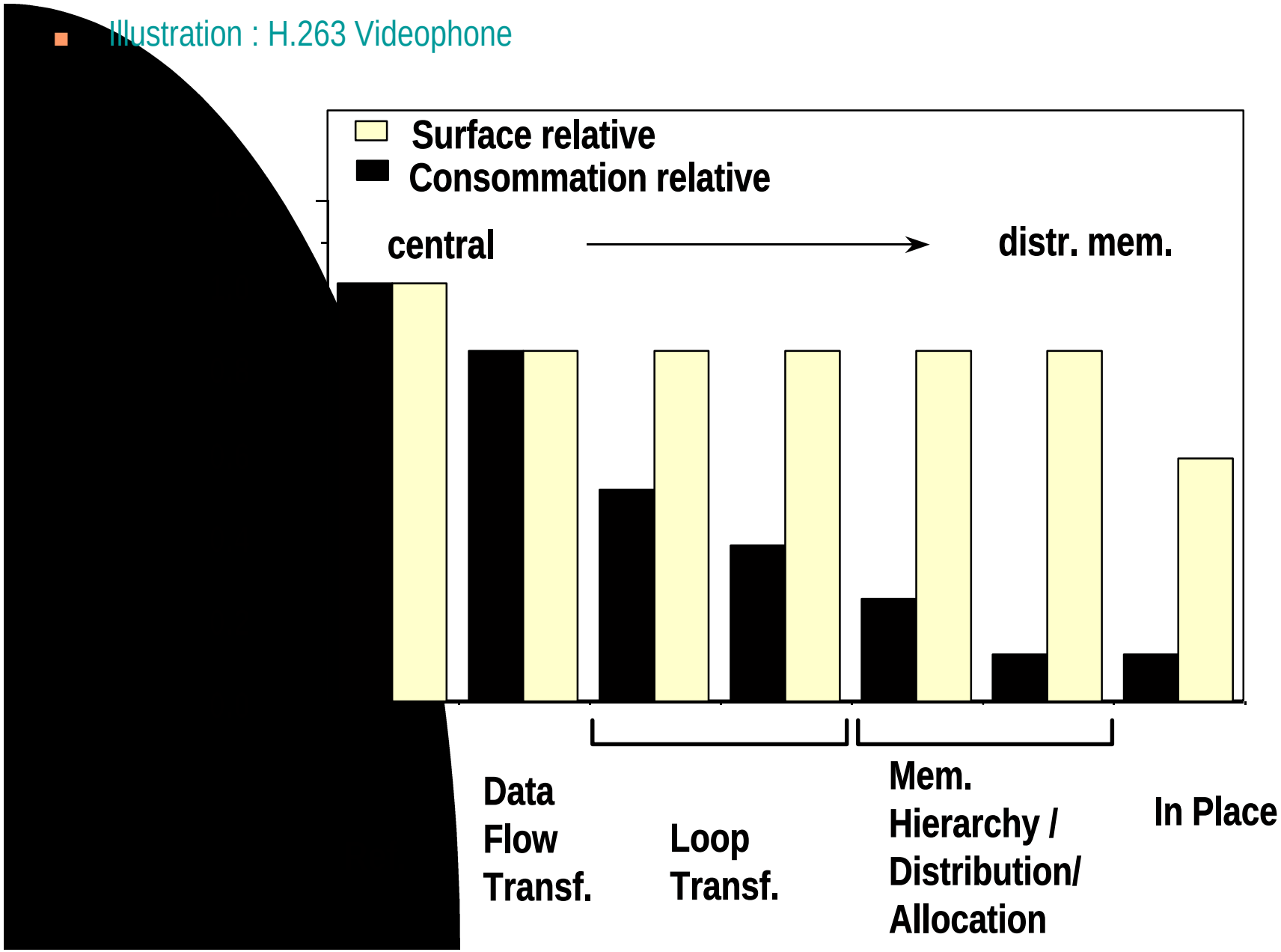
Estimation Système : ex. Atomium

- ATOMIUM (IMEC, Belgique) : dédié à l ’optimisation mémoire



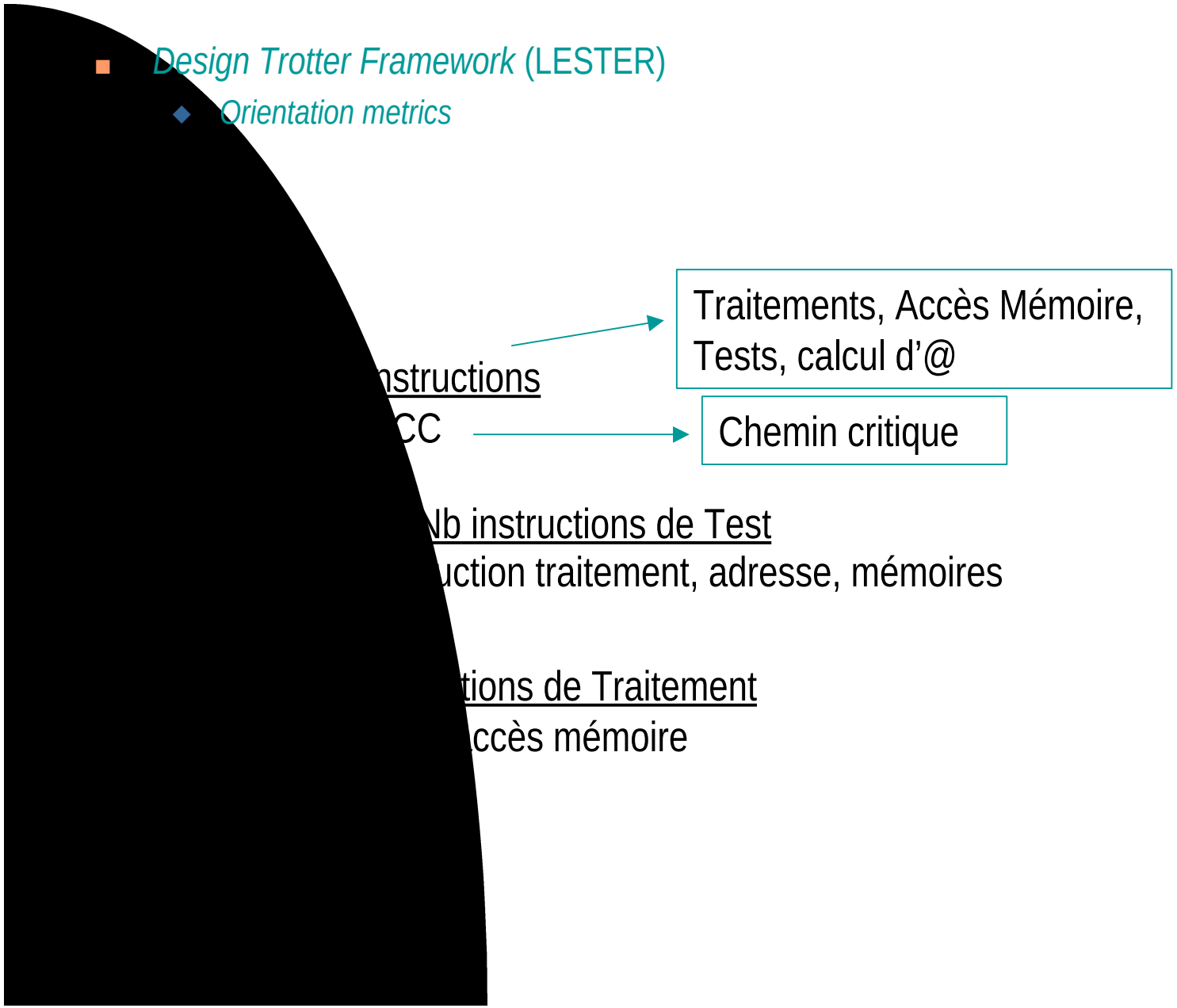
... ex. Atomium

- Illustration : H.263 Videophone



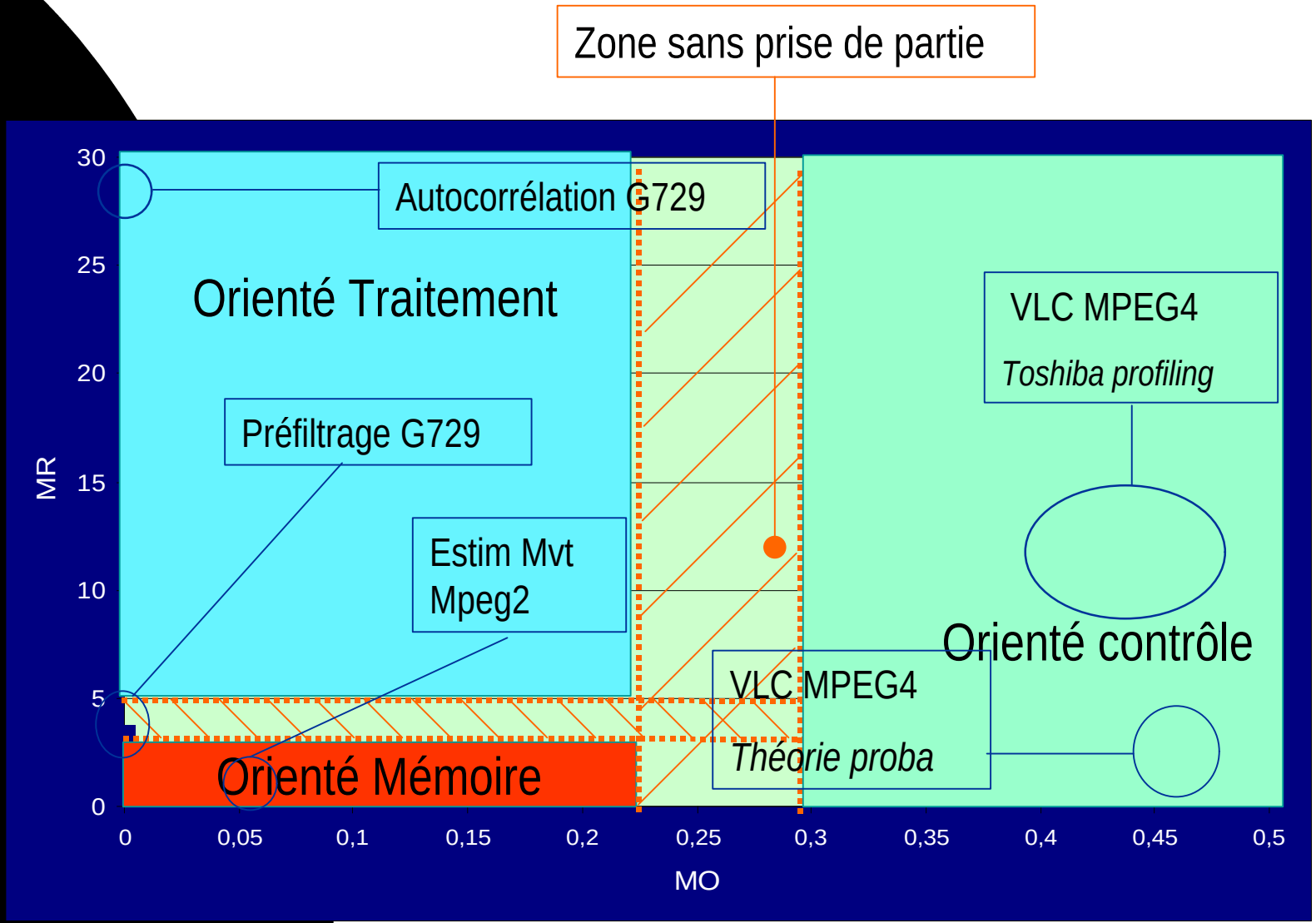
Estimation Système : ex. DTF

- *Design Trotter Framework* (LESTER)
  - *Orientation metrics*



Estimation Système : ex. DTF

- *Design Trotter Framework* (LESTER)



Partitionnement

Partitionnement

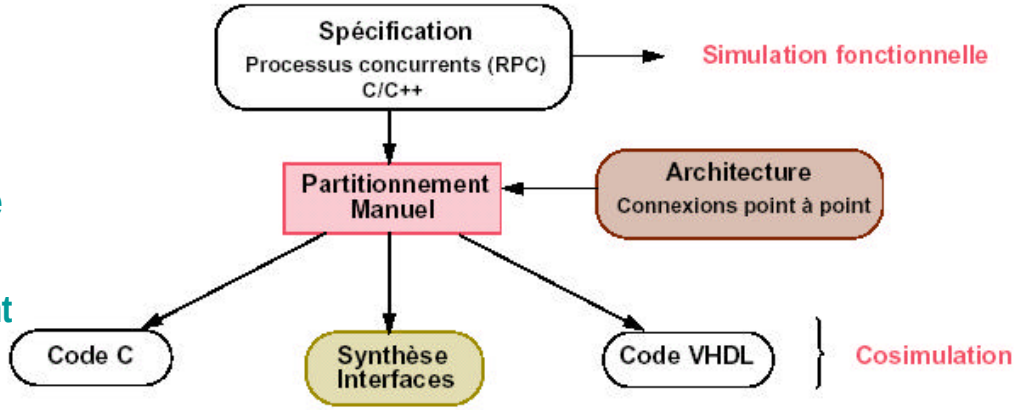
- Caractéristiques d ’un outil / méthode de partionnement
  - Automatique / Interactif / Manuel
  - Fonction de coût et contrainte
    - contrainte de temps minimalisation de la surface et/ou consommation
    - contrainte de surface (ressources) minimisation du temps et/ou conso.
  - Type d ’architectures
    - Multi-Processeurs
    - Mono-Processeur et ASIC (Accélérateur)
    - Mono-Processeur et Co-processeur et/ou Accélérateurs
    - Mono-Processeur + RTOS
  - Niveau de granularité considéré
    - Grande : fonction ou tâche (RTOS)
    - Moyenne : boucle
    - Faible : BLI
  - Méthode de résolution (algorithmes/heuristiques d’optimisation)
  - Modèle d’application (CFSM, DFG, CDFG, HCDFG,GT, ...)

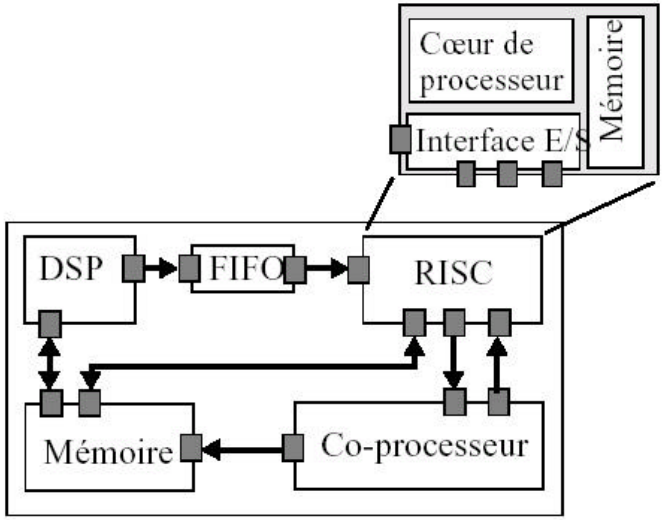
Partitionnement (Manuel-1)

- Partitionnement **manuel** car
  - Partitionner = Problème NP complet !
  - Une fois partitionné encore faut-il que le système fonctionne !
  - Outils d'aide à la conception via la **co-simulation**
- Environnements
  - POLIS (Berkeley) base de l 'outil VCC (Cadence)
  - CoWare (*Spin Off* de l'IMEC)
  - Tni-Valiosys outil Cosimate (à l 'origine TIMA Grenoble)
  - MCSE (IRESTE, Nantes)

Partitionnement (Manuel-2)

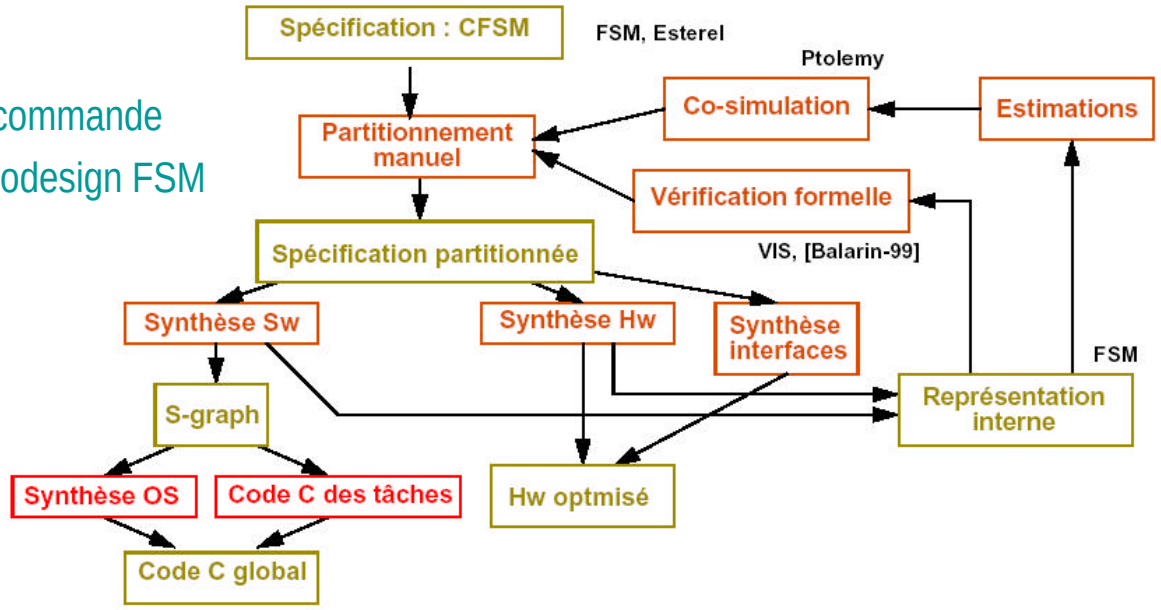
- Coware
  - Points forts
    - Synthèse automatique des interface
    - Cosimulation
    - Globalement asynchrone localement synchrone
    - Protocole générique (RDV)
    - Parallélisme traitement/transferts
  - Points faibles
    - duplication des données
    - connexion point à point

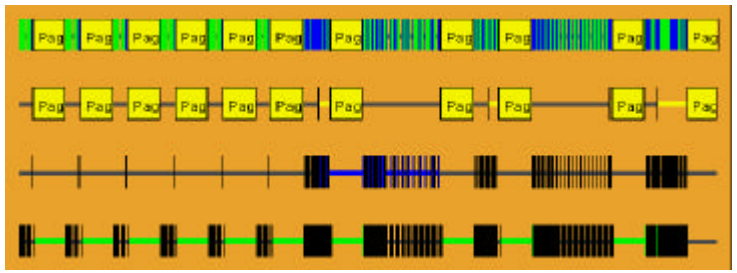




Partitionnement (Manuel-3)

- Polis (Vcc) [Berkeley/Cadence]
  - Post partitionnement
  - Environnement orienté contrôle -commande
  - Représentation avec le modèle Codesign FSM (CFSM)
    - **GALS**
  - Co-simulation : Ptolemy
  - Synthèse des interfaces
  - Synthèse FSM
  - Synthèse SW : génération du RTOS et simulation de l'ordonnancement RTOS
  - Modèle de bus (Ipbus, ..)





Partitionnement (Automatique-1)

- La cosimulation ok mais quid de l'exploration ?
  - **Devient critique dans le cadre des (R)SOC (choix IP, *Cores*, Hiérarchie mémo.)**
  - **Solution initiale et sens de la quête : SW → HW, HW → SW, mixte**
- 2 niveaux d 'exploration :
  - **Après la définition (générique) de la cible**
    - Bibliothèque de solutions HW, SW pour chaque fonction
    - Nombre de solutions par fonction limitée (ex. 2 SW, 2HW)
  - **Avant le choix de la plate-forme matériel (Estimation / exploration système)**

- Évolution des recherches
  - **(i) Multi-processeur**
  - **(ii) 1 processeur + Asic(s) : Pb des com**
  - **(iii) SOC : bus embarqué : 1 proc + HW**
  - **(iv-a) SOC + RTOS + HW**
  - **(iv-a) RSOC + HW**
  - **(v) DRSOC + ASIP + HW**

Board level

Chip level

2 familles :

- Orienté contrôle FSM
- Orienté Traitement (C)DFG

Partitionnement (Automatique-2)

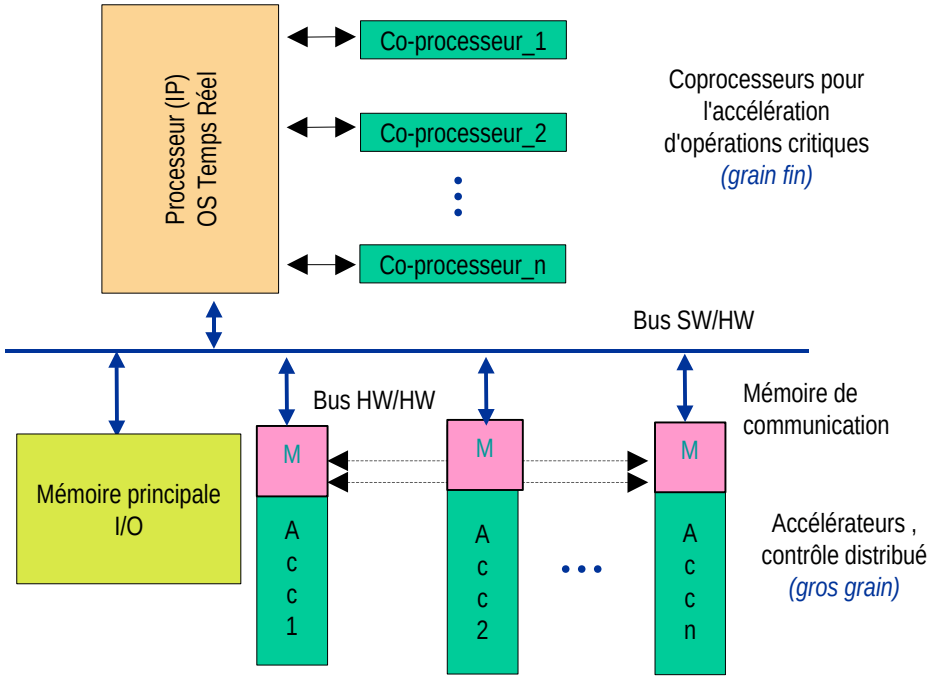
- Récapitulatif (non exhaustif)

Auteurs	Modèle	Granularité	Architecture	Objectif	Technique	Nb Solutions	Remarques
Cosyma [Ernst 93]	C étendu	Instruction	1 proc; 1 ASIC	Temps	Recuit simulé SW->HW	1	
Vulcan [De Micheli 94]	CDFG	Instruction	1 proc; 1 ASIC	Temps	Glouton HW -> SW	1	
GCLP [Kalavade 94]	DFG	Tâche	1 proc; 1 ASIC	Temps / Surface /IO	List Scheduling Modifié	1	
Lycos [Madsen 97]	CDFG	Instruction	1 proce; 1 ASIC	Temps	Prog. dynamique SW->HW	1	
SpecSyn [Vahid 97]	CDFG	Fonction	1 proc; 1 ASIC	Temps / Surface	Min Cut modifié / SER paradigm	Exploration	*Modèles COM fins*
[Teich 97]	CDFG	Fonction	SOC	Temps	Algo. Génétique	Exploration	
Cosyn [Dave 99]	GT	Tâche	Multi-proc	Temps / Conso	Clustering	1	*OS préemtif ou non, Multirate*
Crusade [Dave 99]	GT	Tâche	Multi-proc + FPGA	Temps / Conso	Clustering	1	*OS préemtif ou non + Reconfiguration*
Move [Karkowski 98]	CDFG	Boucles	SOC Multi-Asip	Temps / Nb proc	Enumération	Exploration	
Class [Kuchcinski 99]	DFG	Tâche	SOC	Temps / Surface	Prog. par contrainte et B&B	Exploration	
[LI 00]	CDFG	Boucles	1 proc; 1 FPGA	Temps	Clustering	1	*Reconfiguration*
Syndex [Sorel 94]	DFG	Tâche	Multi-proc	Temps	List Scheduling Modifié	1	*OS non préemtif, Synthèse RTOS*
[Wolf 95]	GT	Tâche	Multi-proc	Temps/ Surface	IDL Ordo + Gradient Search	1	*OS préemtif*
Codef [Auguin 00]	CDFG	Instruction	SOC	Temps/ Surface	Glouton + Exclusion mutuelles	Exploration	*Modèles COM fins*
[Azzedine 02]	GT	Tâche / Instructi	SOC	Temps / Surface	Exact Analysis Ordo + B&B	Exploration	*OS préemptif, multirate, Estimations dynamiques*

- Originalité visée dans *Design Trotter*
  - **Codesign Gros grain et grain fin**
  - **Prise en compte du RTOS dans le cadre Mono-proc SOC**
  - **Large exploration (courbe de compromis par fonction)**
  - **Multi-cadence, tâche périodique, apériodique, sporadique**

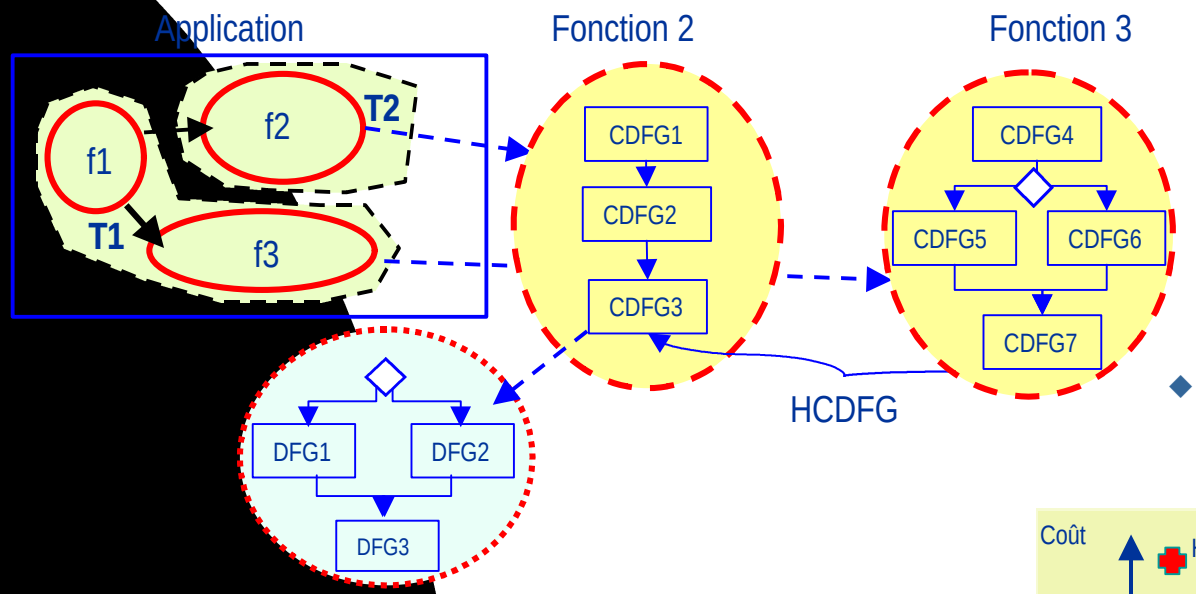
Partitionnement (Automatique-3)

- Projet Ordonnancement RTOS du *Design Trotter* (LESTER)
  - Modèle d 'architecture : SOC
    - **Processeur + RTOS**
    - **Coprocesseur (grain fin) + Accélérateur (gros grain)**
  - Contrainte :
    - **Temps réel**
  - Fonction de coût :
    - **Surface**
    - **puis Conso ..**

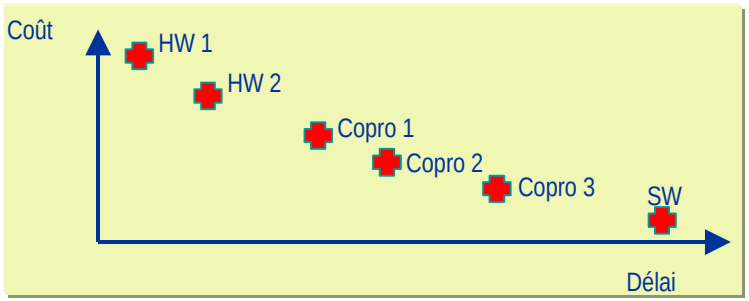


Partitionnement (Automatique-4)

- Modèle d 'application : Graphe de Tâche + HCDFG



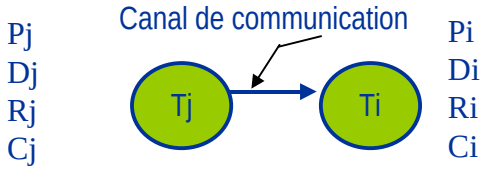
- Estimation Dynamique
  - **niveau CDFG**



- Fonction de coût
  - SW : taille code
  - HW : surface Si
  - Copro : surface Si (avec partage possible des ressources)+ taille du code
- Temps Com :
  - SW-SW ou HW-HW : nul (inclus dans Texec des tâches)
  - HW-SW : temps de lecture de la tâche SW; SW-HW : Temps écriture de la tâche SW

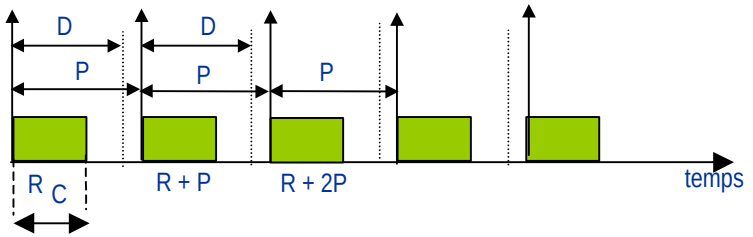
Partitionnement (Automatique-4)

- Technique d'ordonnancement
  - **Caractéristiques Tâche i :**
    - Période Pi
    - Deadline Di
    - Date de réveil Ri
    - Tps exec Ti



Contrainte de dépendance entre deux tâches Tj, Ti : Tj précède Ti

  - **Tâches Périodiques multi-rate**
    - Six cas
      - Pi=Pj ou Pi = nPj ou Pj = nPi
      - Di < Dj ou Dj > Di



  - **Tâches apériodiques**
    - Contrainte stricte : Di devient la pseudo-période (Pi Di)
    - Contrainte lâche : serveur de tâches

Partitionnement (Automatique-5)

- Technique d'ordonnancement
  - **Ordonnancement**
    - Ordonnancement statique préemptif HPF (*High priority first*)
    - Ordonnançabilité : Analyse exacte [Audsley 93]

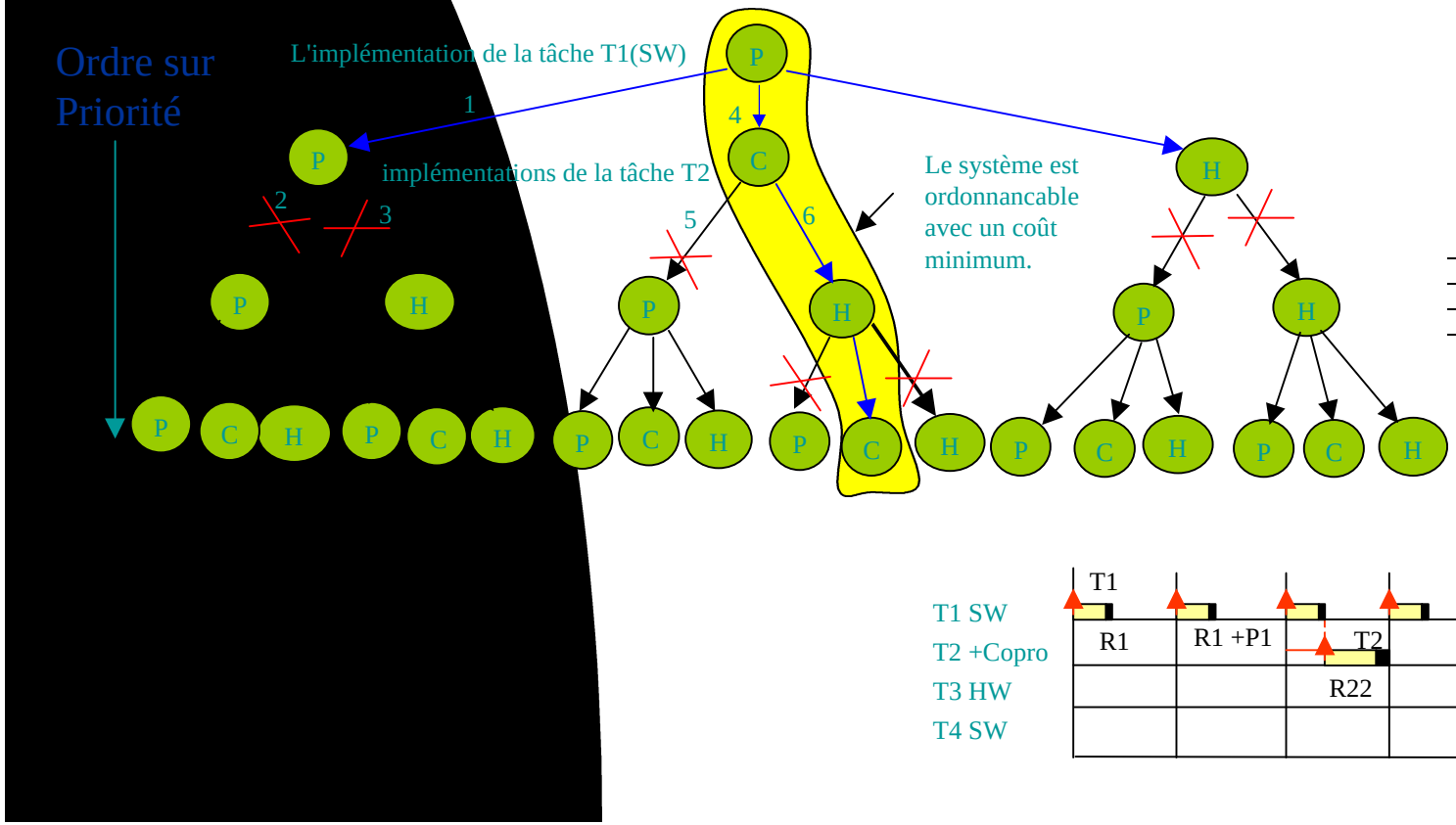
$$\forall j \in hp(i)\ \exists t \le Di \quad t = \sum_{j} \left\lceil \frac{t}{Pj} \right\rceil^{①} * (cj + Comj)^{②} + Bi^{③} + Comi$$

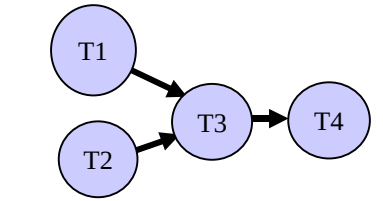
  - ① : nb de fois que la tâche j plus prioritaire que la tâche i est activée dans [0, t]
  - ② : temps de communication des données de la tâche j
  - ③ : temps de blocage par des tâches de priorité plus faible (partage de ressources)

- Méthodologie :
  - **Spécification**
  - **Estimation dynamique niveaux Système / Architecture**
  - **Calcul des priorités** (prise en compte des dépendances et criticité) → **Tâches indépendantes**
  - **Ordonnancement / Partitionnement**
  - **Calcul des dates de réveil (Ri) associées**

Partitionnement (Automatique-6)

- Partitionnement : *Branch & Bound*
  - *Arête gauche : tout SW*
  - *Pb des com : si Tn HW alors C*n-1 =Tcom(n-1)+ Cn-1 : SW → HW : remontée !*
  - *Version heuristique nécessaire*
- Exemple



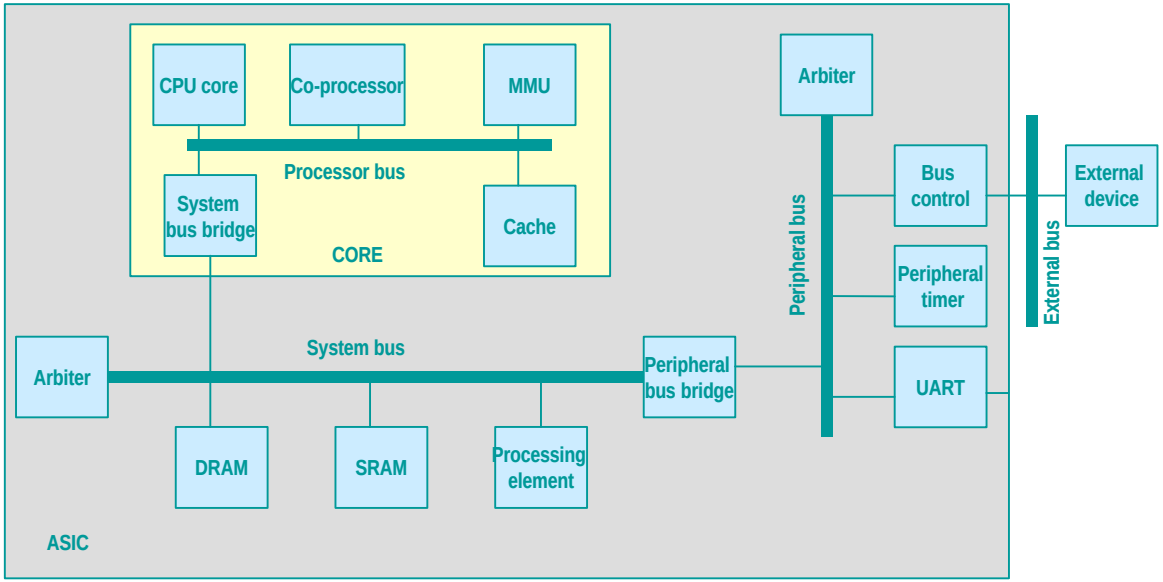


	T_1	T_2	T_3	T_4
Pi	300	900	1800	1800
D_i	200	500	700	1400

Synthèse des communications

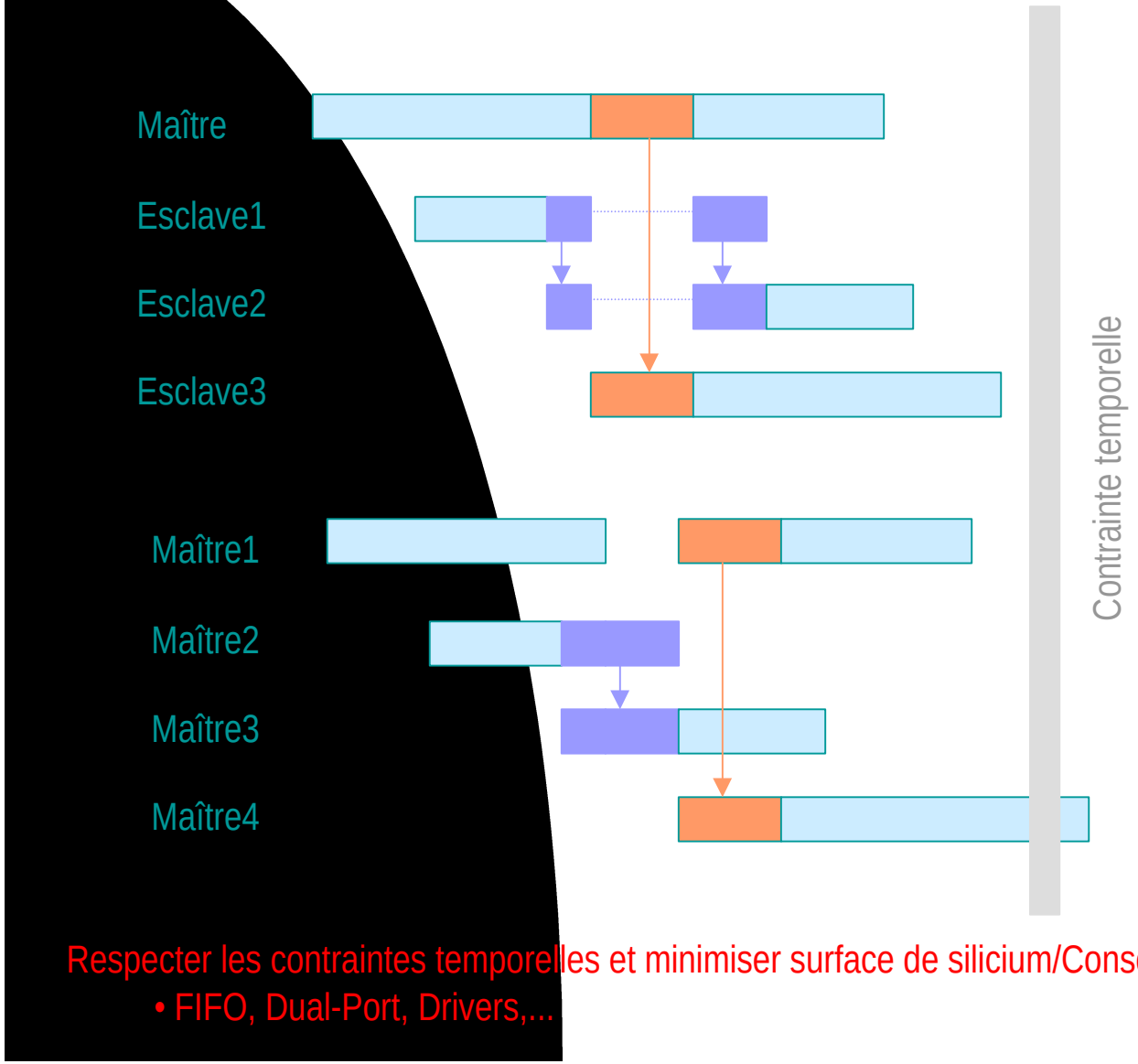
Synthèse des communications

La spécification d'une architecture spécialisée inclut l'interconnexion



- Automatisation de la synthèse d'interface (rendre compatible des protocoles d'unités hétérogènes)
- Déterminer une interconnexion implique :
  - Choisir les ressources de communication : types et nombres de bus, FIFO, Dual-Port, Arbitre de bus ...
  - Allouer les communications de l'application sur les ressources sélectionnées
  - Choisir des modes de transfert :
    - protocole de transfert : mode burst, pipeline, ...
    - transfert synchrone ou asynchrone
    - priorités allouées aux transferts
- Classiquement la synthèse des communications est réalisée :
  - Pendant le partitionnement (modèle simplifié utilisé)
  - Après partitionnement

La réalisation des communications conditionne le comportement global



Respecter les contraintes temporelles et minimiser surface de silicium/Consommation

- FIFO, Dual-Port, Drivers,...

Synthèse des communications

- Automatisation de la synthèse d'interface

Auteurs	*Modèle*	*Granularité*	*Interface*	*Architecture*	*Remarque*
Nestor et Thomas [Nestor86]	Chronogrammes	Netlist	logique + contrôleur	1 émetteur / 1 récepteur	Protocole synchrone
Boriello et Katz [Boriello87]	Graphe d'événements	Netlist	logique + contrôleur	1 émetteur / 1 récepteur	Protocole Synchrone / asynchrone
Chiodo et al. [Chiodo93]	Graphe d'événements	Netlist	logique + contrôleur	1 émetteur / n récepteurs	Sauvegarde données
Brodersen et Sun [Brodersen92]	Protocole de haut niveau (IDL)	RTL	logique + contrôleur	1 émetteur / 1 ou n récepteurs	Architecture complexe
Madsen et Hald [Madsen95]	Graphe d'événements	Netlist	logique + contrôleur	1 émetteur / 1 récepteur	Protocoles complexes
Narayan et Gajski [Narayan95]	Graphe d'événements	RTL	logique + contrôleur	1 émetteur / 1 récepteur	
Lin et Vercauterren [Lin94]	Réseau de pétri	RTL	logique + contrôleur	1 émetteur / 1 récepteur	

  - Permet de rendre compatible des protocoles d'unités hétérogènes a priori non compatibles
  - Plusieurs types de protocoles sont utilisés (point à point, broadcast) et leur définition s'effectue à des niveaux de granularités variables
  - Plus le niveau de granularité est fin et plus cela demande une connaissance fine des protocoles
  - Plusieurs outils permettent de gérer la génération des interfaces (Archimate, Coware)

Synthèse des communications

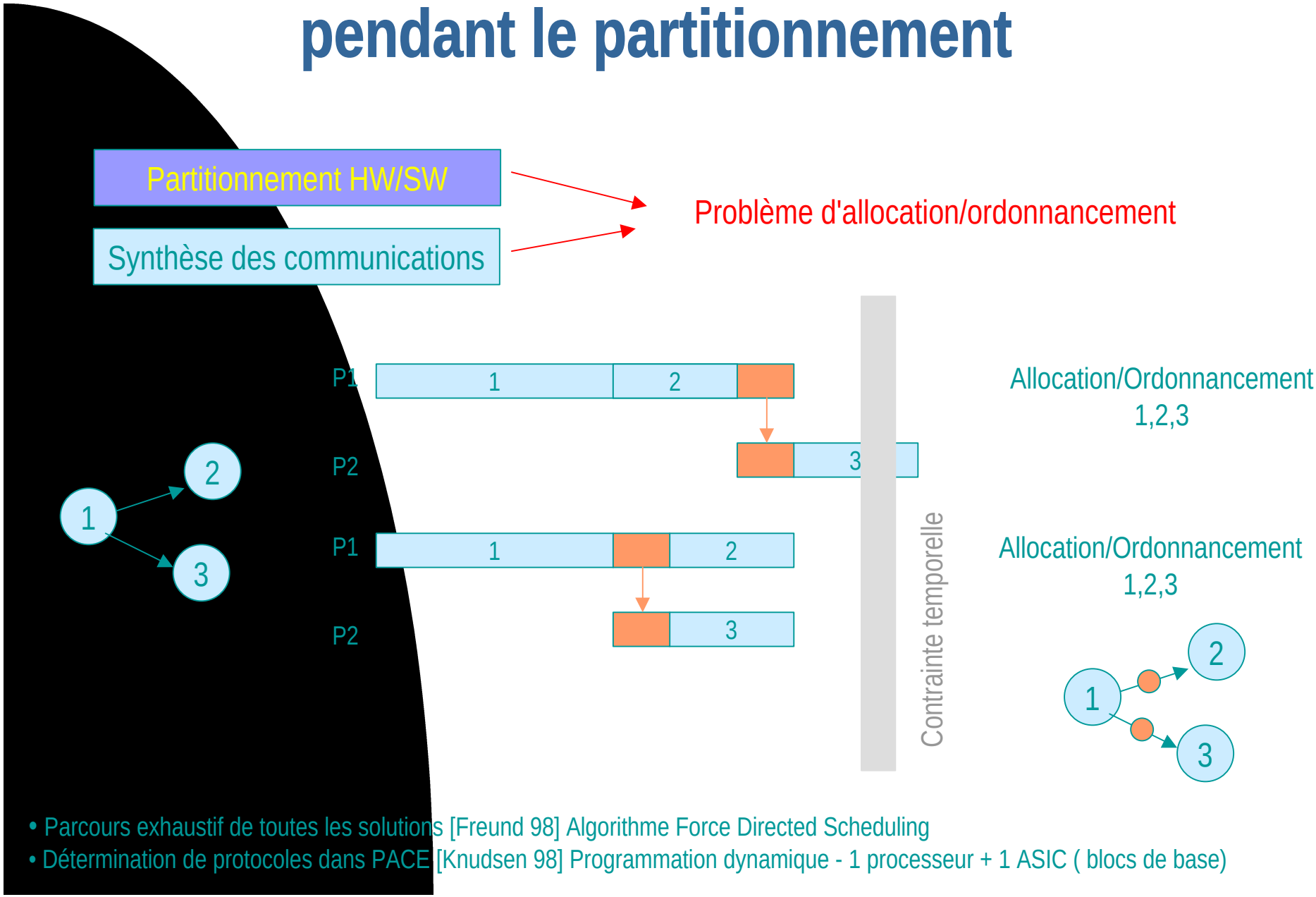
Synthèse des communications pendant le partitionnement

Auteurs	*Modèle*	*Méthode*	*Architecture*	*Remarque*
Henkel et al. [Henkel94]	Taille du code - Surface HW - Temps	Ordonnancement - Allocation	Processeur + Coprocesseur + Mémoire partagée	Optimisation locale, bus
Bender [Bender96]	Occupation des bus	MILP : temps d'exécution, nombre de processeurs, coût des communication	Processeurs + Coprocesseurs + ASICS	Pas de mode DMA, bus
Yen et Wolf [Yen95]	Durée, Mode, Support, Taille	RMS	Processeurs	DMA/CPU, Point à point/bus, buffer/mémoire partagée
Dave et al. [Dave99]	Durée, Mode, Support, Taille	Cluster	Processeurs	Optimation en puissance
Freund [Freund98]	Durée, Mode, Support, Taille	FDS	Processeurs + Coprocesseurs + ASICS	DMA/CPU, FIFO/bus, Synchrone/asynchrone
Knudsen et Madsen [Knudsen98]	Durée, Mode, Support, Taille, Burst, Débit, Overhead	Programmation dynamique	Processeur + ASIC	PCI/USB

Synthèse des communications après le partitionnement

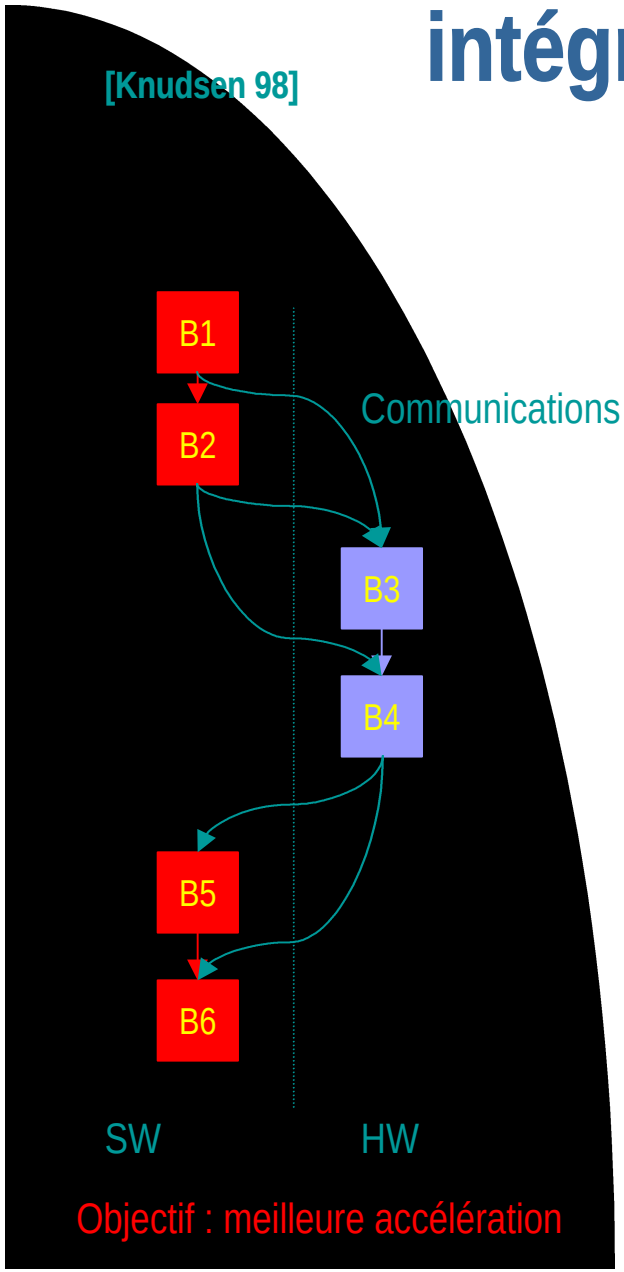
Auteurs	*Modèle*	*Méthode*	*Architecture*	*Remarque*
Daveau, Ben-Ismail et Jerraya [Daveau95]	Niveau de granularité élevé (Protocole, taux de communication moyen et crête) - Bibliothèque de supports de communication	Allocation de protocoles	Processeurs + ASIC + FPGA + Mémoire partagée	tni-valiosys (Archimate)
Narayan et Gajski [Narayan94]	Bus partagé - Maximiser l'utilisation du bus en distribuant temporellement les communications (respect de la bande passante, taille minimale des bus, taux de communication moyen et crête)	Déplacement des communications - fusion des bus	Processeurs + ASIC + Bus	Pas de prise en compte des optimisations du bus sur l'ordonnancement global
Gong, Gajski et Bakshi [Gong96]	Mémoire interne/externe, connexion point à point/bus partagé. Le temps de communication peut quadrupler entre données mémoire interne vs externe	Optimisation de la localisation des données (interne, externe)	Processeurs + ASIC + Bus/point à point + Mémoire	Génération automatique des interfaces
Filo, Ku, Coelho et DeMicheli [Filo93]	Graphe flôt de données, Réordonnancement des nœuds du graphe. Bus, mémoire	Maximiser les communications synchrones	Processeurs + ASIC	Horloge unique
Gogniat [Gogniat97]	Graphe flôt de données, Réordonnancement des nœuds du graphe. DMA/CPU, bloquant/non bloquant, FIFO/Bus	Allocation et Optimisation. Maximiser les communications synchrones	Processeurs + ASICS + Bus + FIFO	Architecture à base de bus
Cuesta [Cuesta01]	Extension [Gogniat97], communication par interruptions, Graphe incluant du contrôle	Allocation et Optimisation. Maximiser les communications synchrones et semi-synchrones	Processeurs + ASICS + Bus + Mémoires	CODEF (I3S - Philips Semiconductors Sophia)

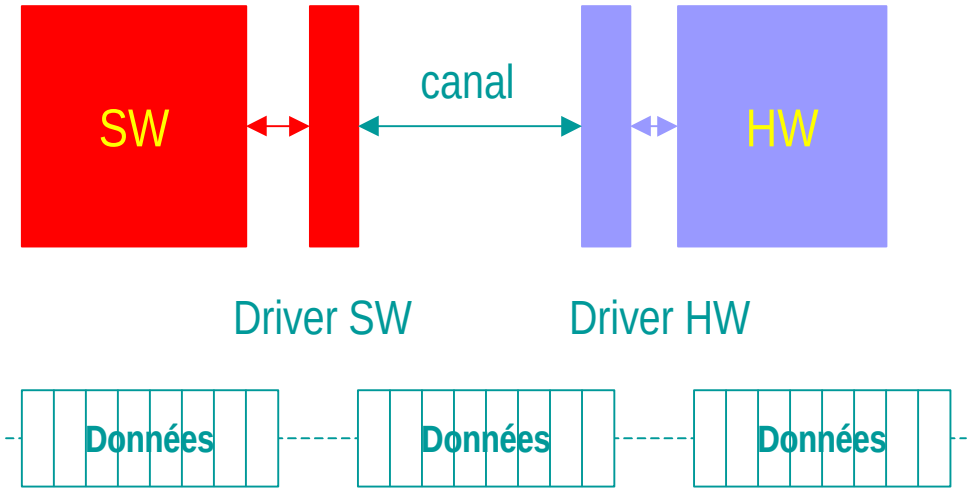
Synthèse des communications pendant le partitionnement



- Parcours exhaustif de toutes les solutions [Freund 98] Algorithme Force Directed Scheduling
- Détermination de protocoles dans PACE [Knudsen 98] Programmation dynamique - 1 processeur + 1 ASIC (blocs de base)

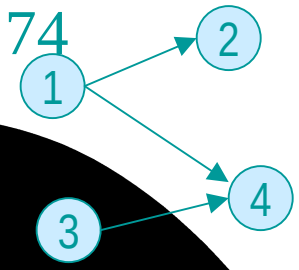
Synthèse des communications intégrée au partitionnement





- Prise en compte des communications :
  - Nombre de burst
  - Nombre de données par burst
  - Débit sur le canal
  - Horloge des drivers
  - Overhead d'appel des drivers
  - Cycles d'overhead du contrôle des transferts
  - Taille des mots transférés
  - Largeur du canal

Synthèse des communications aprés partitionnement (CODEF)

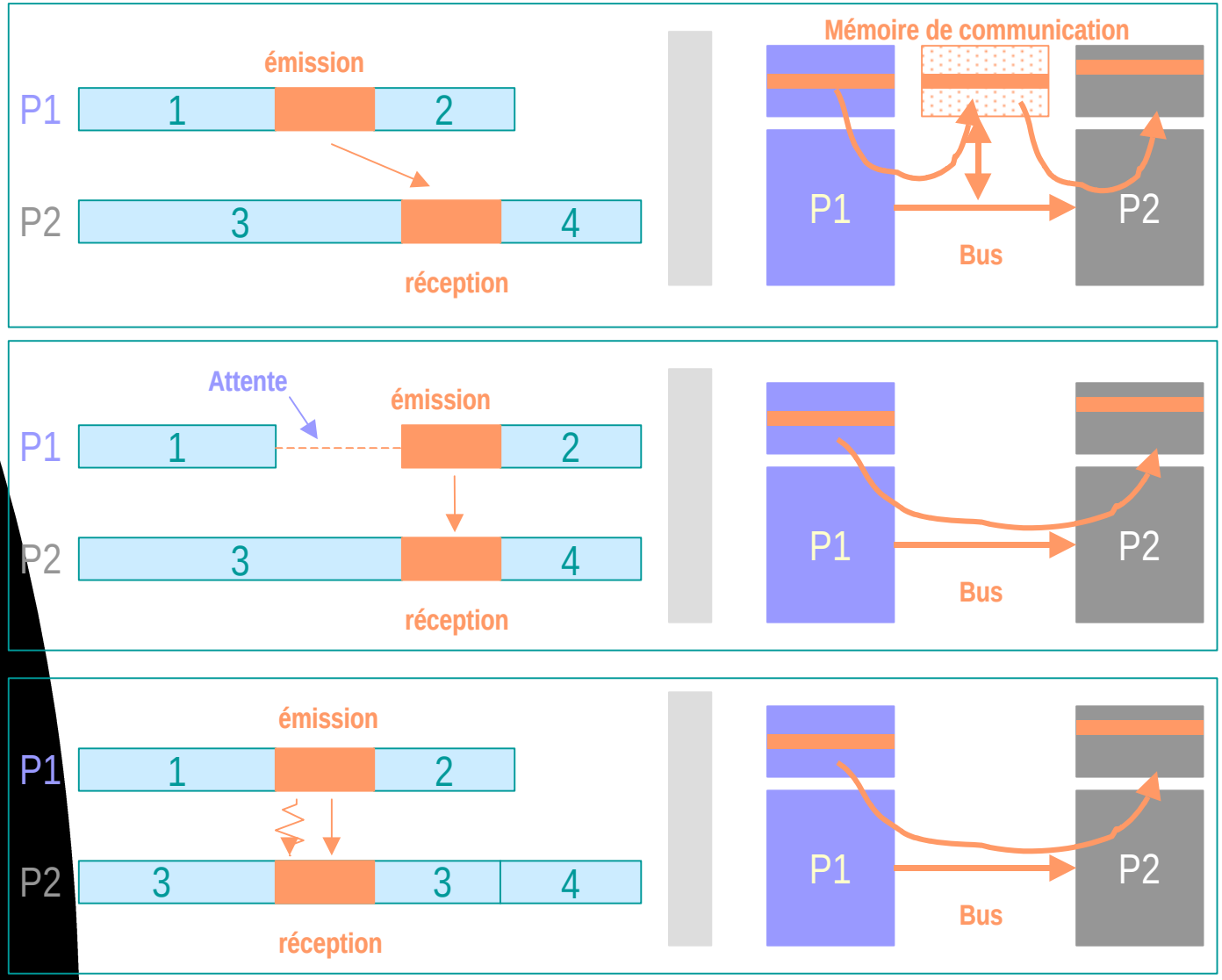


Transfert Asynchrone (A)
Le plus coûteux
surface, consommation

Transfert Synchrone (B)
Le moins coûteux

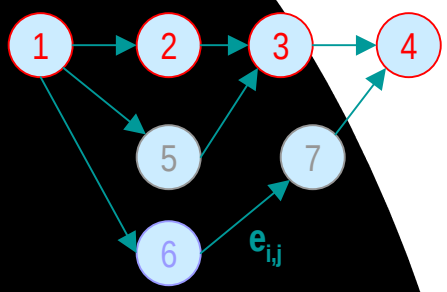
Transfert Synchrone
par interruption (C)

• Privilégier B puis C puis A

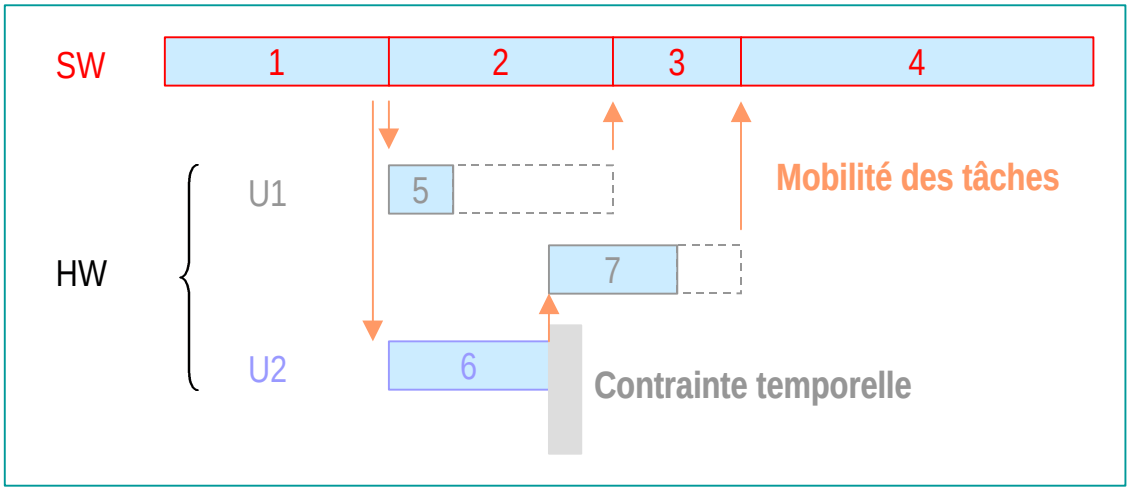


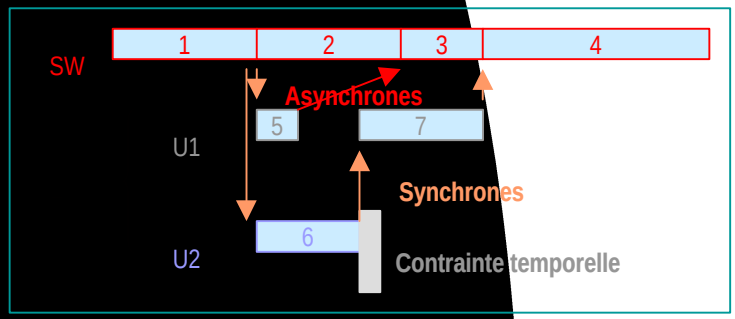
[Cuesta00]

Synchronisation par utilisation des mobilités

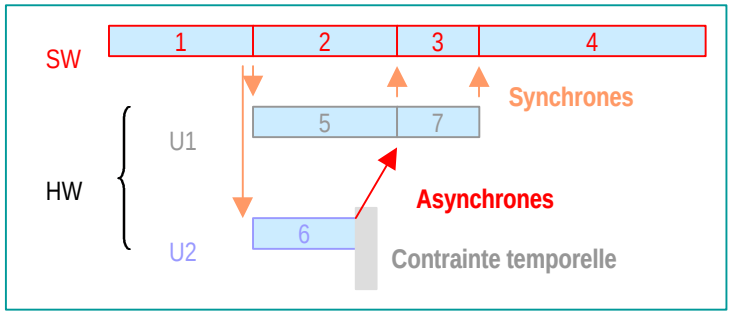


Après partitionnement





Synchronisation 7->4 et 6->7, Transfert 5->3 asynchrone

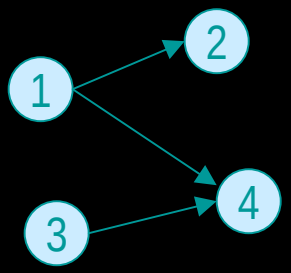


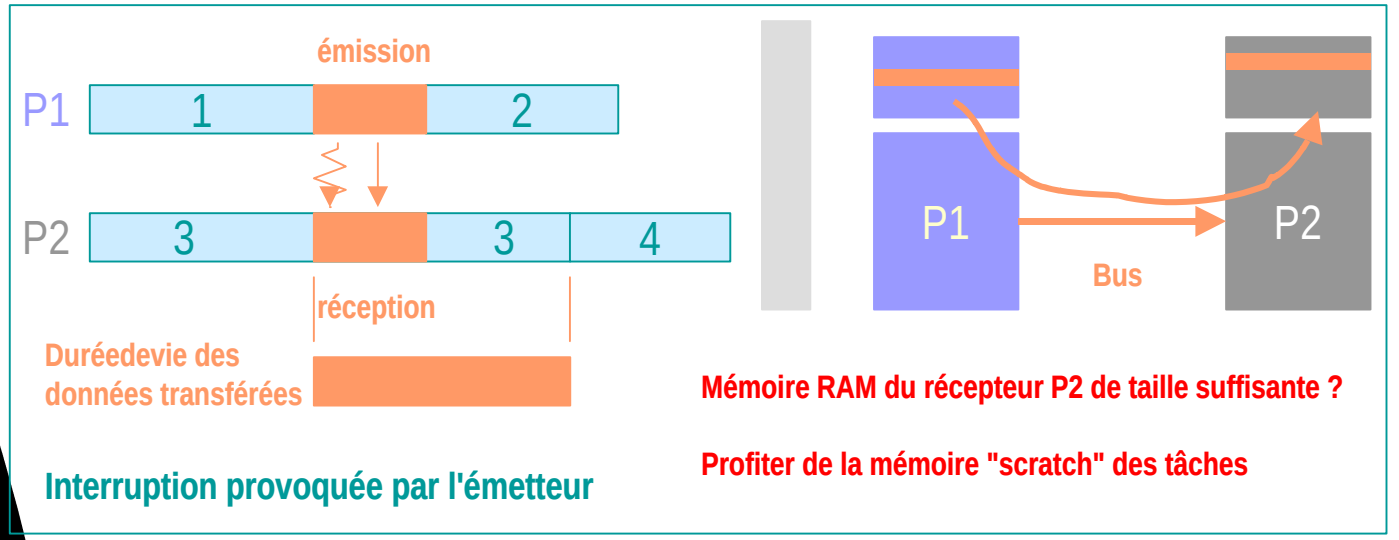
Synchronisation 5->3 et 7->4, Transfert 6->7 asynchrone

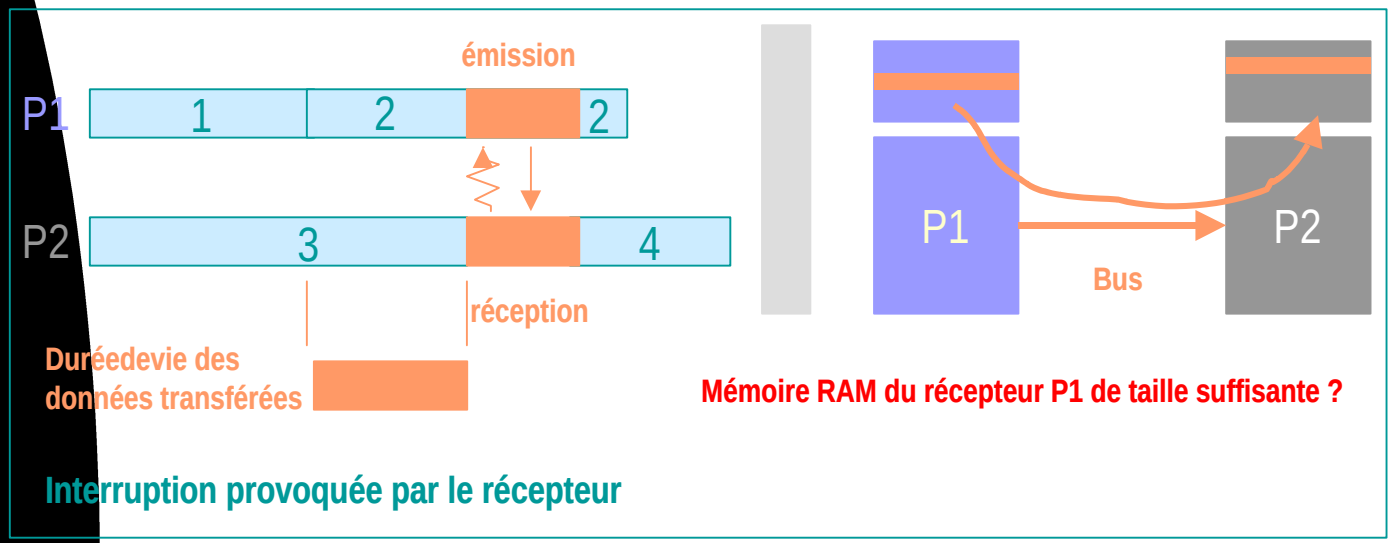
1) Synchronisation des arcs qui ne provoquent pas de transferts asynchrones (Volume données transférées $e_{i,j}$ / Mobilité $e_{i,j}$)
2) Synchronisation des arcs (Volume données arcs asynchrones / Volume données arcs synchrones

[Gogniat97]

Synchronisation par interruptions







[Cuesta00]

Synthése des supports de communication