Conception faible consommation

Nathalie JULIEN
LESTER UBS-Lorient
Nathalie.Julien@univ-ubs.fr

ARCHI '03, Roscoff, le 01/04/03

Plan

I. Généralités sur la consommation

L.E.S.T.E.R
Laboratoire d'Electronique des Systèmes TEmps Réel

I. Généralités sur la consommation

I.1. Pourquoi ?

I.2. Quoi ?

I.3. Comment ?

I.4. Qui consomme ?

I.5. Niveaux d 'action

I. Généralités sur la consommation

I.1. Pourquoi ?

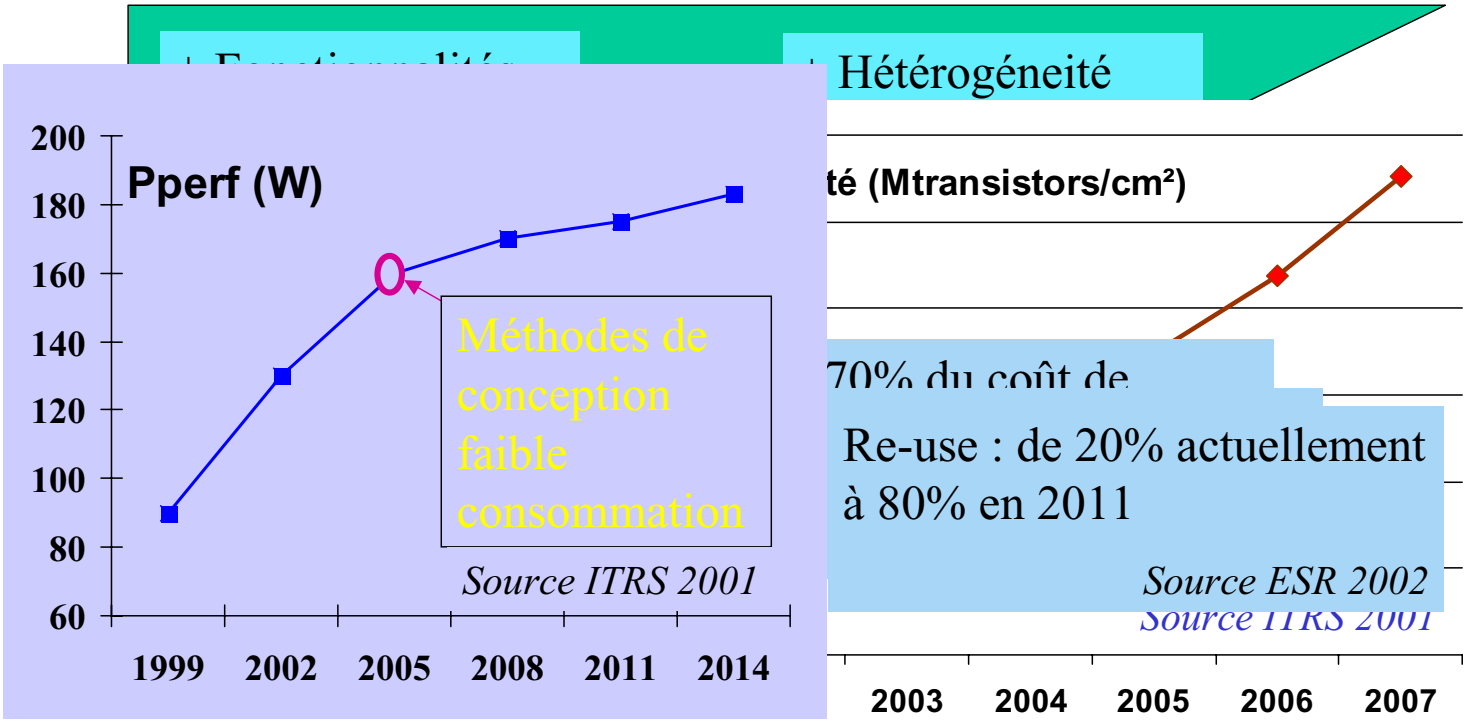
I.2. Quoi ?

I.3. Comment ?

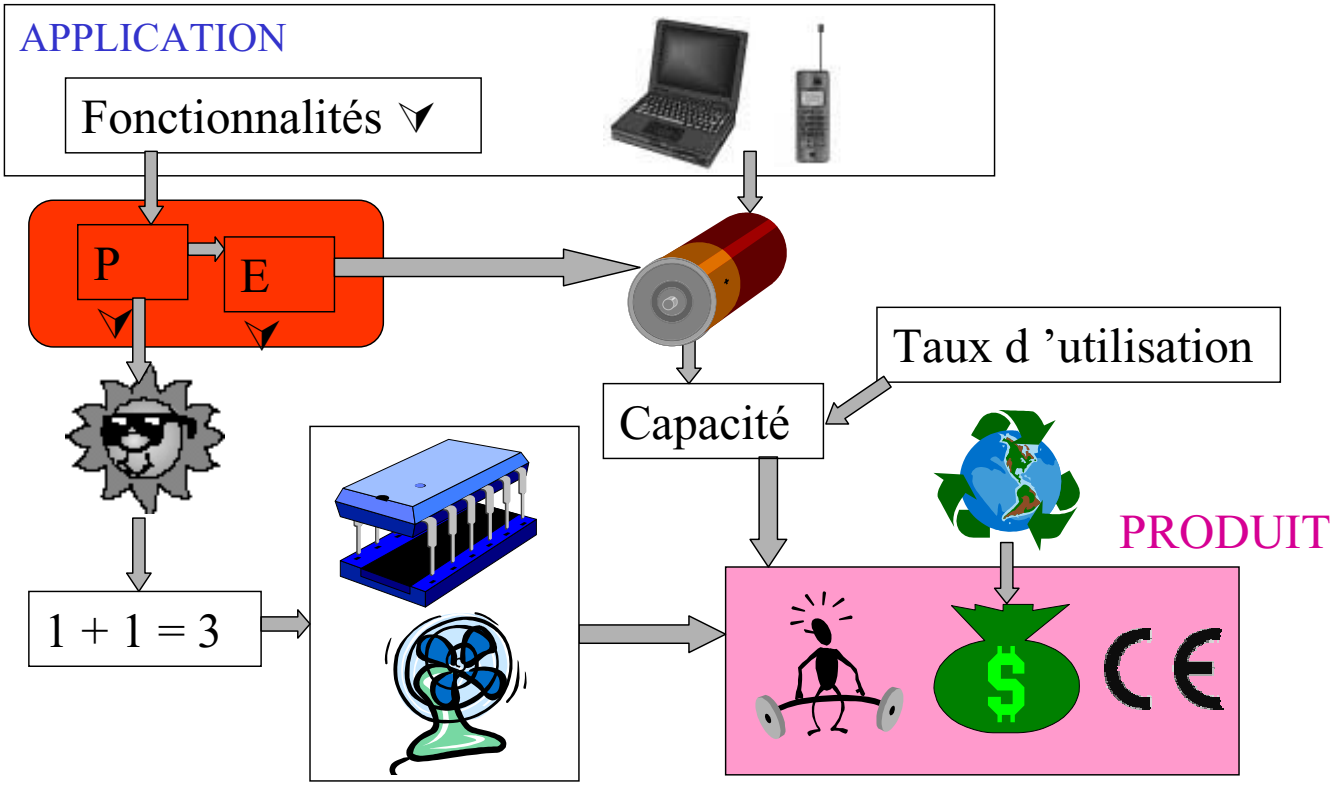
I.4. Qui consomme ?

I.5. Niveaux d 'action

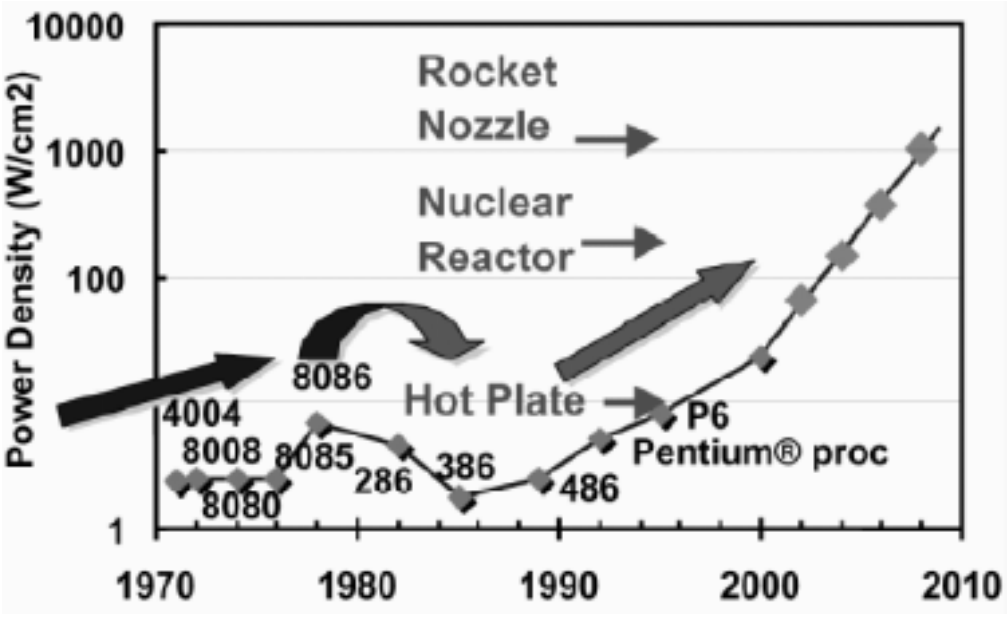
I.1. Pourquoi ?



I.1. Pourquoi ?



I.1. Pourquoi ?



Source Tutorial Date '02

I.1. Pourquoi ?

Améliorer la consommation d'une application, c'est aussi améliorer ses performances !

I. Généralités sur la consommation

I.2. Quoi ?

- Critères
  - Puissance (Watt) : $P_{crête}$
  - Energie (Joule) : P_{moyen} x Temps
  - Energie/opération ou MIPS/W
  - Energie x Délai

I.2. Quoi ?

$$P(CMOS) = P_{statique} + P_{dynamique}$$

$$P_{statique} = I_{leakage}\ V_{dd}$$

négligeable ?

$$P_{dynamique} = P_{court\text{-}circuit} + P_{commutation}$$

$$P_{commutation} = \alpha\ C_L\ V_{dd}^2\ F$$

I. Généralités sur la consommation

I.1. Pourquoi ?

I.2. Quoi ?

I.3. Comment ?

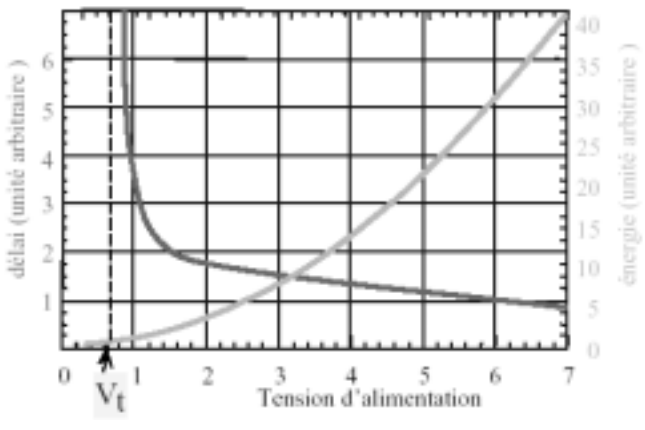
I.4. Qui consomme ?

I.5. Niveaux d 'action

I.3. Comment ?

$$P_d = \alpha C_L V_{dd}^2 F$$

- ↘ $t_{pd} \alpha C_L V_{dd}/(V_{dd} - V_t)^2$
- si Vt ➚ I leakage ↘
- ➚ chemin critique
- ↘ taille et complexité
- ➚ marge de bruit
- Compromis : Voltage scaling
- Energie x Délai optimum pour Vdd #3Vt



I.3. Comment ?

$$P_d = \alpha C_L V_{dd}^2 F$$

- 2 composantes : circuit et interconnexions
  - ➢ taille des circuits
    - ➢ taille des transistors ; mais t_{pd} ➘
    - *transistor sizing*, *gate sizing*
  - ➢ interconnexions : de + en + important
    - ➢ nombre : *placement and routing*
    - ➢ C avec conductivité + grande, section + petite, permittivité + petite
- ➢ coût d'une transition logique : autres logiques
- partage des ressources
- substitution d'opérateurs par transformations

L.E.S.T.E.R
Laboratoire d'Electronique des Systèmes TEmps Réel

I.3. Comment ?

$$P_d = \alpha C_L V_{dd}^2 F$$

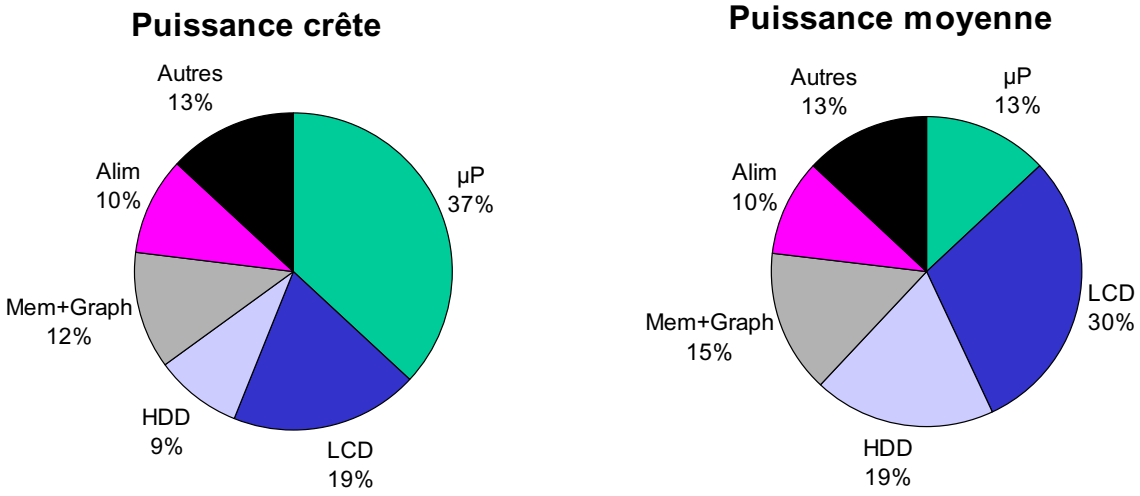
- α : activité des données : nombre de transitions de 0 à 1 par cycle d 'horloge
- αF : activité de commutation
- codage et format de codage des données
- limiter les commutations de fonctions
- limiter le nombre d'opérations réalisées
- modes de veille
- partitionnement
- ➢ commutations parasites (glitch)

I.3. Comment ?

- Conclusions
  - Compromis A-T-P
  - adaptation du circuit à l 'environnement ou aux données : changements dynamiques du circuit (power management…)
  - éviter les pertes (glitch, clock…)
  - exploiter la localité (partitionnement mémoire…)

I. Généralités sur la consommation

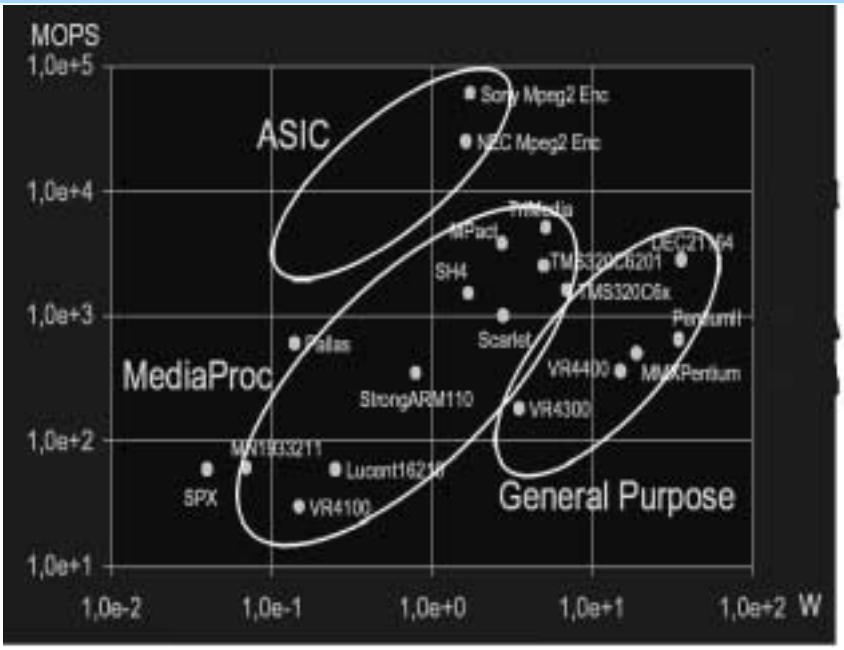
I.4. Qui consomme ?



Mesures sur PC portable (*tutorial Date'02*)

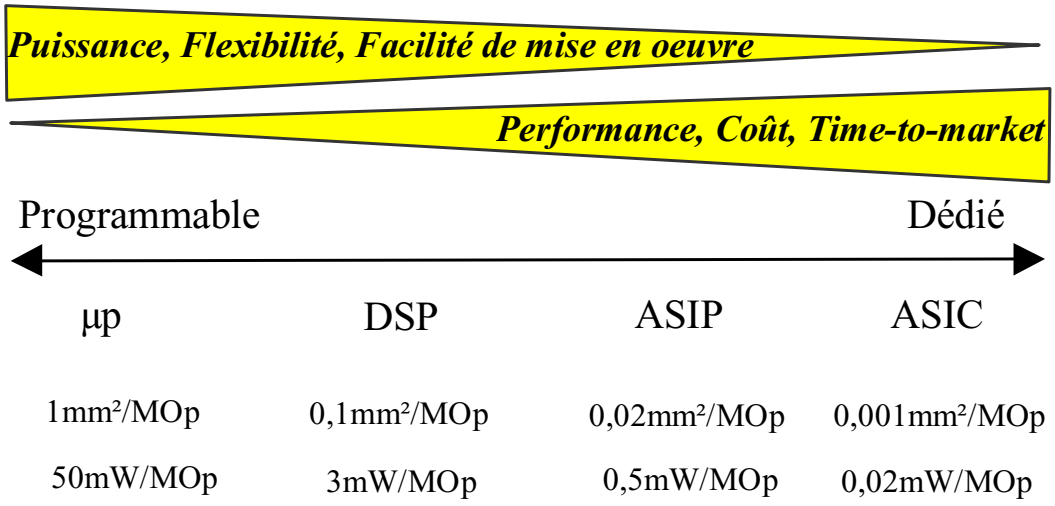
I.4. Qui consomme ?

Espace de conception pour une application MPEG-2 **[DeMic00]**

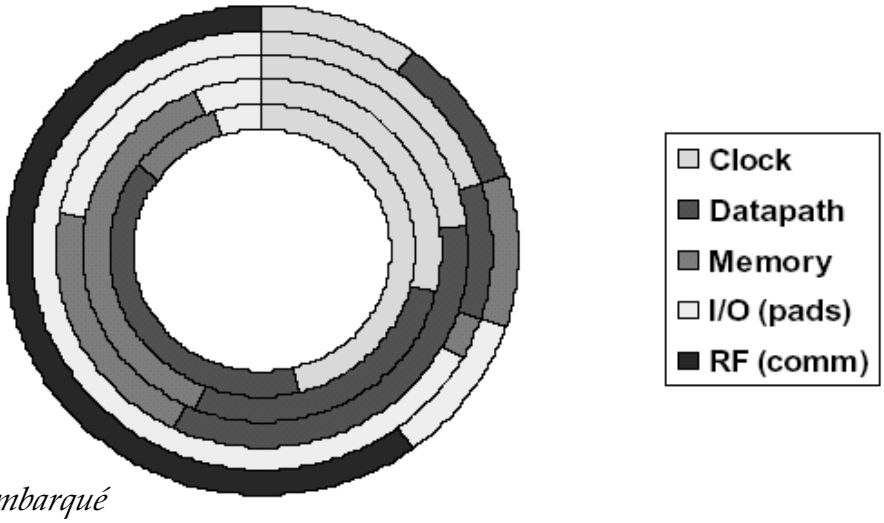


I.4. Qui consomme ?

Performances comparées des circuits en technologie 0,5 µm.



I.4. Qui consomme ?



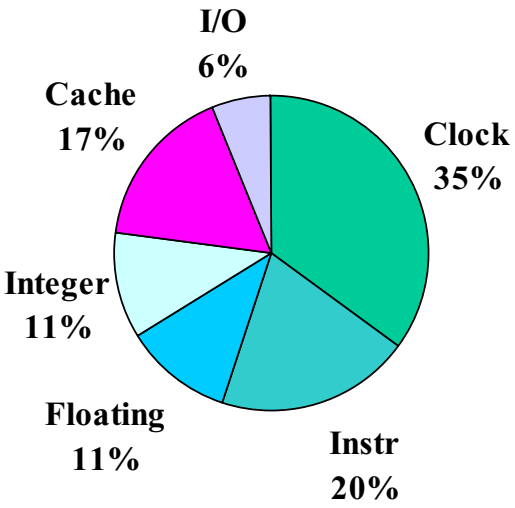
Cercle intérieur : microprocesseur embarqué
2ème cercle : CPU + cache on-chip
3ème cercle : ASIC décodeur MPEG2
4ème cercle : ASIC commutateur ATM
Cercle extérieur : PDA sans fil

source M. J. Irwin Univ. Penn State Univ.

I.4. Qui consomme ?

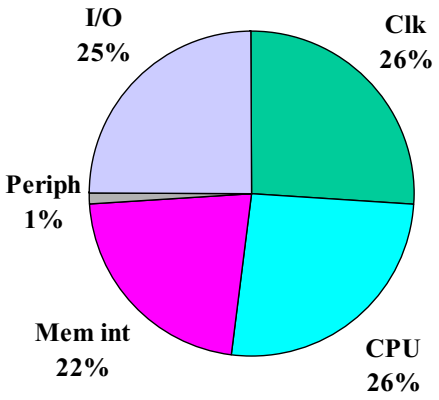
DEC Alpha 21264

Source MIT



TMSC6201

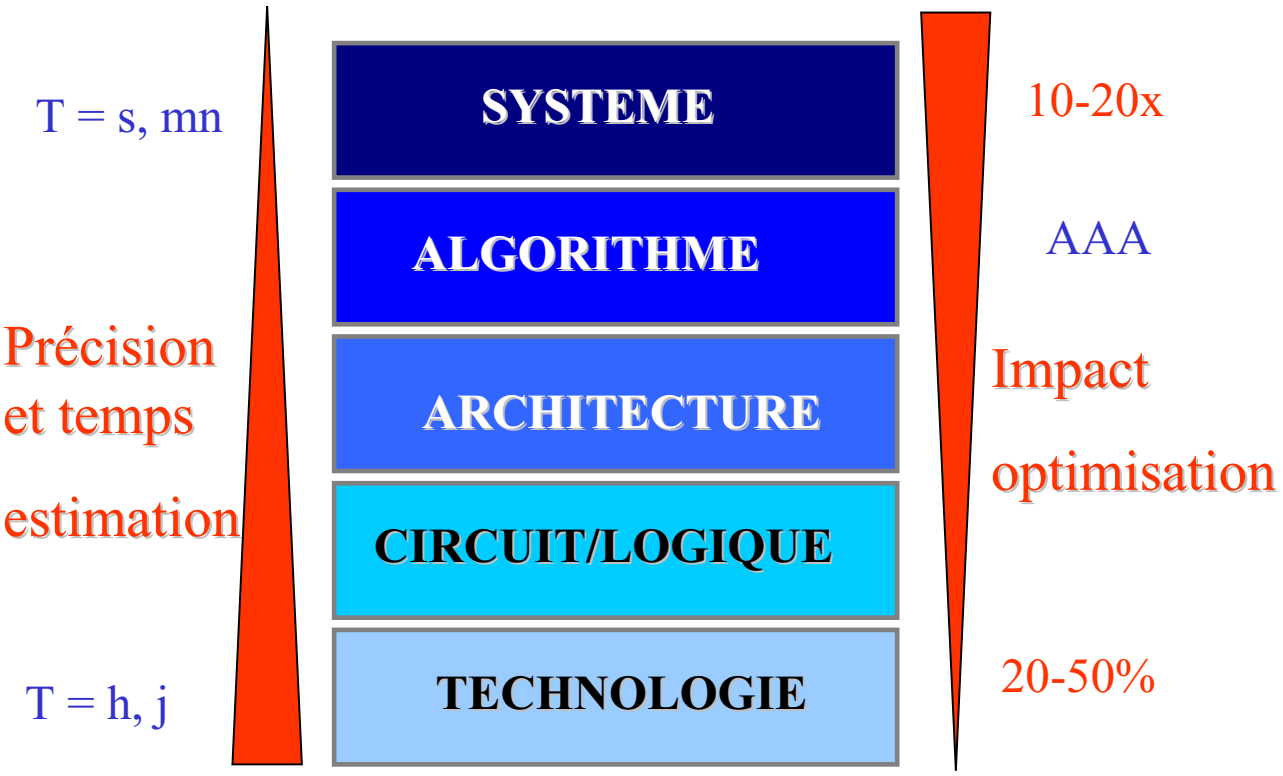
Source TI



Total 1.76 W

I. Généralités sur la consommation

I.5. Niveaux d 'action



I.5. Niveaux d 'action

- Pour maîtriser la consommation, il faut :
  - agir à chaque niveau de la hiérarchie de conception
    chaque % gagné est intéressant !!! (Intel)
  - développer des outils d 'estimation
    - évaluer le budget en consommation
    - détecter les points chauds du système/circuit
    - évaluer l 'efficacité des différentes optimisations
  - développer des méthodes d 'optimisation
    - intégrer la consommation comme critère de conception dans les outils de synthèse de haut niveau

II. Méthodes d 'optimisation

L.E.S.T.E.R
Laboratoire d'Electronique des Systèmes TEmps Réel

II. Méthodes d 'optimisation

II. Méthodes d 'optimisation

II.1. Niveau technologique
II.2. Niveau circuit/logique
II.3. Niveau architectural
II.4. Niveau algorithmique
II.5. Niveau système
II.6. Outils

II.1. Niveau technologique

- Domaine
  - Sources de consommation clairement identifiées
    - modèles physiques des transistors
  - Technologie de plus en plus adaptée
  - Domaine bien exploré
- Outils d 'estimation
  - Très bonne précision
  - Temps de simulation très important

II.1. Niveau technologique

- transistor sizing : ajustement de la taille des transistors
- actions sur les seuils
- SOI : Silicium On Insulator
  - pas de courant de fuite de jonction
  - ➢ C parasites 2 à 3 x
- actions sur Vdd
  - alimenter avec Vdd basse les parties lentes
  - séparer les tensions d 'alimentation des circuits d 'E/S et du cœur du circuit.

II. Méthodes d 'optimisation

II.2. Niveau circuit/logique

- Domaine
  - Modèle de consommation moyenne au cours du temps
    - Bibliothèques des composants
  - nature des signaux d'entrées influence fortement la consommation du circuit ➔ Choix des vecteurs de simulation des opérateurs logiques
  - Nombreuses études réalisées
- Outils d 'estimation
  - Temps important
  - Bonne précision
  - Méthodes probabilistes/statistiques

II.2. Niveau circuit/logique

- Dimensionnement des portes : *gate sizing*
- différents types de logique : préchargement, adiabatique, pass-transistor, statique/dynamique, synchrone/asynchrone, à faible excursion...
- optimisations logiques : precomputation (précalcul)
- extraction des sous-fonctions communes
- partage des ressources
- format de codage
- choix des opérateurs (sélection)
- action sur les horloges : clock gating

II. Méthodes d 'optimisation

II.1. Niveau technologique
II.2. Niveau circuit/logique
II.3. Niveau architectural
II.4. Niveau algorithmique
II.5. Niveau système
II.6. Outils

II.3. Niveau architectural

- Domaine
  - sources de consommation à peu près maîtrisées
  - implantation réelle inconnue (avant P/R)
    - ➔ estimation de paramètres
  - optimisations expérimentales liées à une application
  - peu de travaux prennent en compte les mémoires
- Outils d 'estimation
  - bonne rapidité
  - précision raisonnable

II.3. Niveau Architectural

- Optimisations
  - Diminuer Vdd : *voltage scaling*
  - Parallélisme/Pipeline
  - Actions sur les bus
  - opérations avec entrées corrélées sur les mêmes opérateurs : diminue l 'activité
  - partage des ressources exploitant la localité spatiale
  - éteindre les modules inutilisés : *clock gating*
  - compression de code

II.3. Pipeline/Parallélisme

Impact du pipeline et du parallélisme (*source Rabaey*)

Datapath	Vdd (V)	Surface	Puissance
Simple	5	1	1
Pipeline	2,9	1,3	0,37
Parallèle	2,9	3,4	0,34
Pipeline/parallèle	2	3,7	0,18

Fonction sup(A+B, C)

II.3. Actions sur les bus

- $C_{parasite}$ entre boîtiers ≈ 100 x $C_{parasite}$ interne
- E/S ≈ 50% P d 'un circuit
- 15 à 20% de P dans interchips drivers
- encodage de bus
- compresser l 'information des lignes d 'adresses
- Placement
  - *Torino 02 :* ➢ C : placement des lignes non équi-distantes ; E(bus)➢ 30%
  - *NEC Lab* : placement des données sur les bus

II.3. Bus encoding

- Principe : coder pour diminuer le taux d 'activité
- Ne transmettre que les transitions : intéressant quand forte corrélation temporelle des données (images)
- T0 code : pour les bus d'adresse
  - + INC = 1 pour 2 adresses consécutives
  - sinon INC = 0 et les lignes d'adresses utilisées normalement
- Codage de bus inversé *Bus Invert Code*
- Utilisation d 'un « livre de codes »
  - on transmet le code d 'une ancienne valeur la plus proche et la différence
  - NEC 02 : 18% à 40% d 'économie, S supplémentaire non négligeable

II.3. Bus encoding

Séquence	Gray	Bus Invert	T0 code
0 0 0	0 0 0	0 0 0 0	0 0 0 0
0 0 1	0 0 1	0 0 0 1	1 0 0 0
0 1 0	0 1 1	1 1 0 1	1 0 0 0
0 1 1	0 1 0	1 1 0 0	1 0 0 0
1 1 1	1 0 0	1 0 0 0	0 1 1 1
1 0 0	1 1 0	0 1 0 0	0 1 0 0
1 1 0	1 0 1	0 1 1 0	0 1 1 0
1 1 1	1 0 0	0 1 1 1	1 1 1 0
0 0 0	0 0 0	1 1 1 1	1 1 1 0
12 transitions	**10 transitions**	**10 transitions**	**9 transitions**

Gray : minimise les transitions sans ligne supplémentaire
T0 code : intéressant pour les bus instructions
Bus Invert code : intéressant pour les bus données (larges)

II. Méthodes d 'optimisation

II.4. Niveau algorithmique

- Domaine
  - sources de consommation mal maîtrisées
  - architecture inconnue
  - comparaison d'algorithmes par estimation
  - travailler à ce niveau a un impact énorme sur la consommation
- Outils d 'estimation
  - estimateur basé sur un modèle architectural spécifique (ASIC,DSP)
  - modèle bruit blanc : ignorer les statistiques des données
  - utilisation de métriques particulières
    - localité, régularité, complexité : nombre d'opérations, concurrence...
    - chemin critique/fréquence de fonctionnement

L.E.S.T.E.R
Laboratoire d'Electronique des Systèmes TEmps Réel

II.4. Niveau algorithmique

- 2 composantes
  - dissipation inhérente à l 'algorithme
    - unités d 'exécution (nombre d 'opérations)
    - mémoire (nombre d 'accès)
  - implementation overhead (lié à l 'architecture)
    - contrôle, interconnexions, registres
    - importance comparable à l 'autre composante
    - régularité, concurrence, localités : spatiale (partionnement en clusters) et temporelle (durée de vie moyenne des variables)

II.4. Niveau algorithmique

- ➢ dissipation inhérente à l 'algorithme
  - ➘ vitesse pour ➢ Vdd par :
    - pipeline fonctionnel
    - retiming
    - mise en ligne des fonctions
    - transformations algébriques : associativité, commutativité, distributivité
    - transformations de boucles : déroulage de boucle, réordonnancement, fusion...
    - compromis ATP

L.E.S.T.E.R
Laboratoire d'Electronique des Systèmes TEmps Réel

II.4. Niveau algorithmique

- ➢ activité de l 'algorithme :
  - ➢ taille
  - ➢ # opérations : élimination des sous-expressions communes, élimination du code mort, propagation des constantes
  - *strength reduction* : remplacer une opération par une + simple (* par cste = additions + décalages)
- ➢ implementation overhead
  - la localité spatiale guide le partitionnement
  - algos réguliers : - de contrôle et interconnexions
  - ➢ S pour ➢ C_{bus}

II. Méthodes d 'optimisation

II.1. Niveau technologique
II.2. Niveau circuit/logique
II.3. Niveau architectural
II.4. Niveau algorithmique
II.5. Niveau système
II.6. Outils

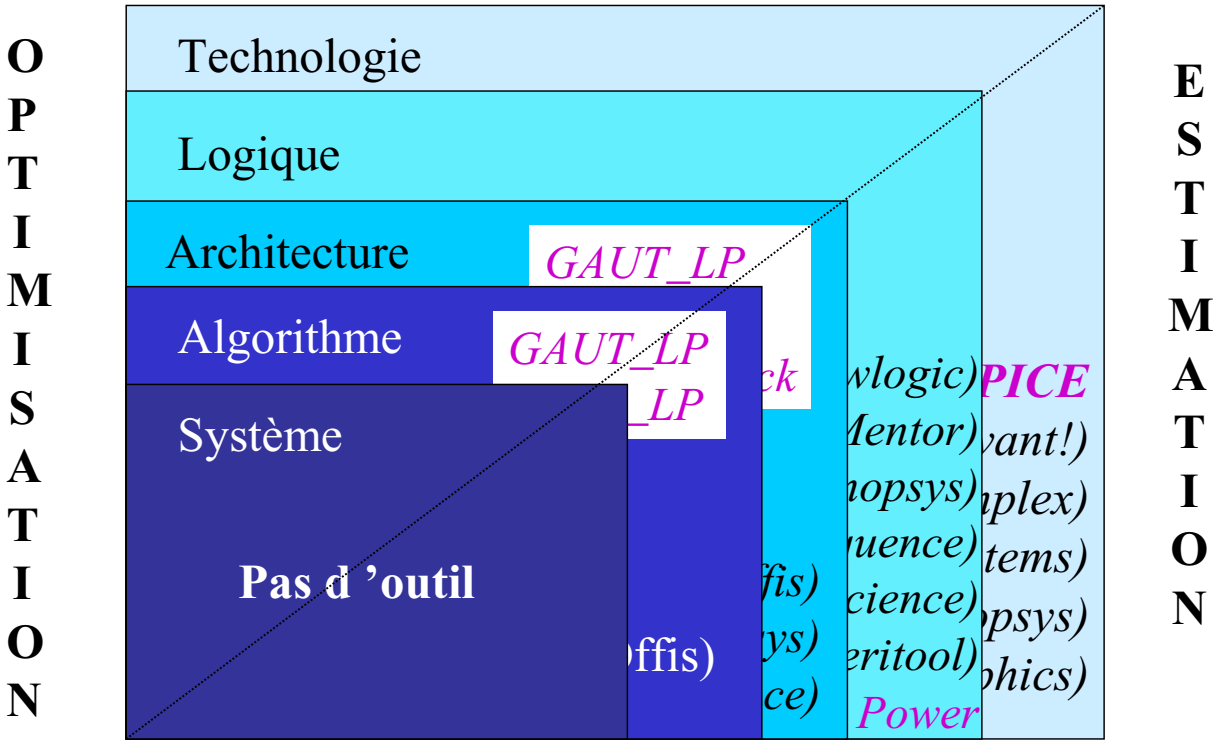
II.5. Niveau système

- Contrôle d 'activité
- Partitionnement
- Voltage scaling
- Mises en veille
- Power management (ACPI)
- peu d'outils et d'études

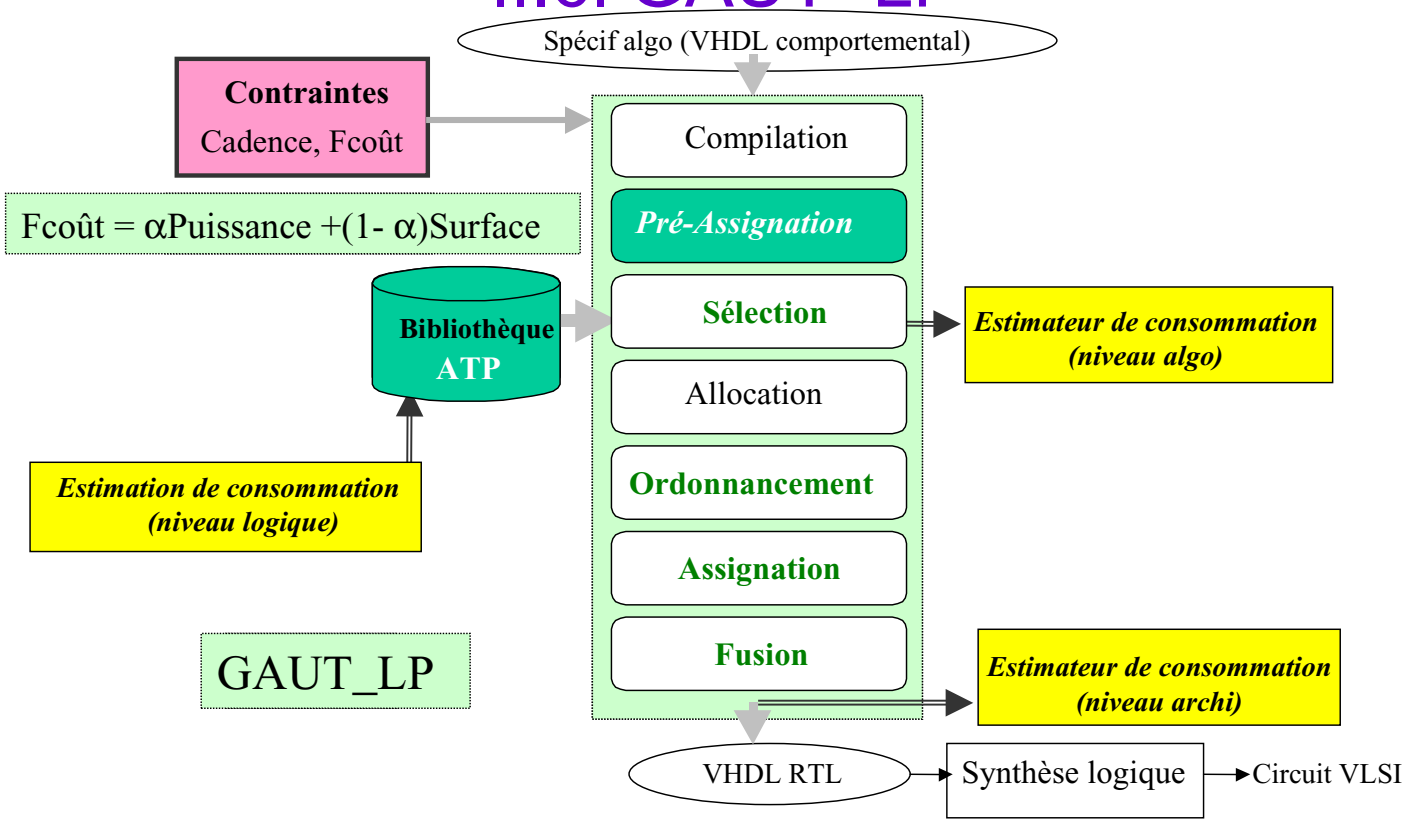
II. Méthodes d 'optimisation

II.1. Niveau technologique
II.2. Niveau circuit/logique
II.3. Niveau architectural
II.4. Niveau algorithmique
II.5. Niveau système
II.6. Outils

II.6. Outils

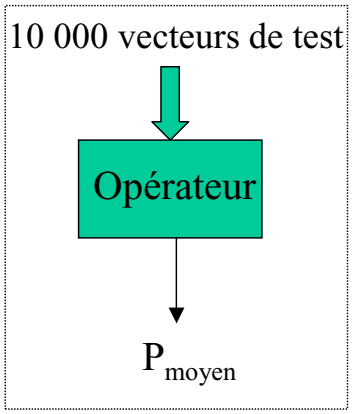


II.6. GAUT_LP



II.6. GAUT_LP

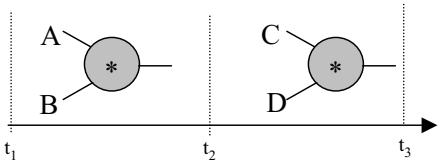
5 caractérisations statistiques par estimateur logique



Opérateurs 16 bits	T (ns)	S (10^{-3} mm²)	P (µW) à 1 MHz	
			Modèle 1	Modèle 2
Multiplieur (DataPath)	32.3	1337	18827	9865
Multiplieur Pipeline (DataPath)	22	1337	18821	9502
Multiplieur (VHDL)	53.3	1023	11820	5956
Multiplieur rapide (VHDL)	33.2	1270	16231	8512

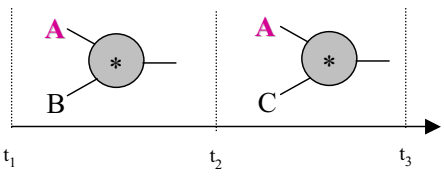
- 50%

Modèle 1



Deux variables non-corrélées

Modèle 2



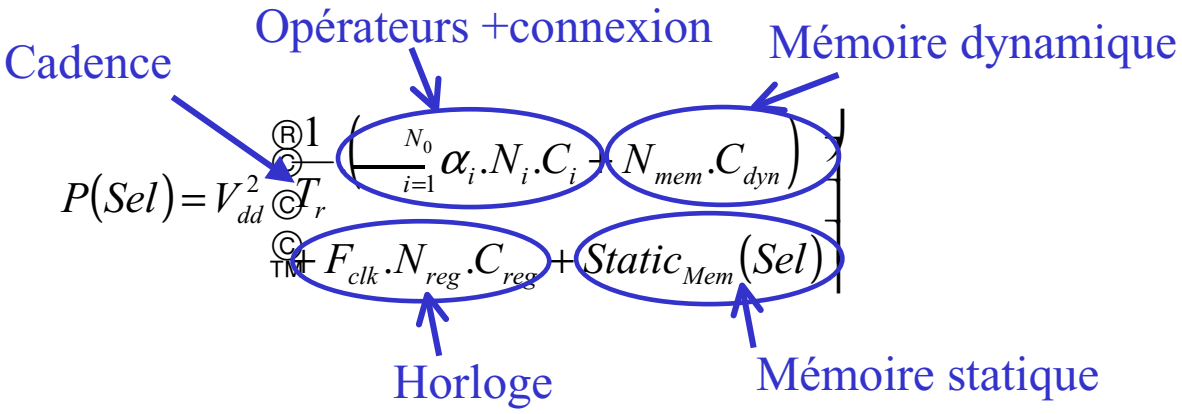
Une variable non-corrélée et une corrélée (stable)

→ Erreur < 1%

intervalle de confiance 95%

II.6. GAUT_LP

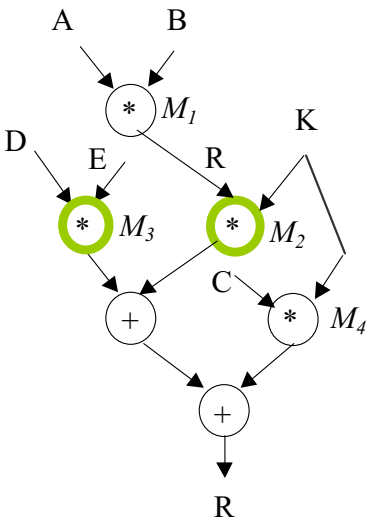
$$P(Sel) = P_{opérateurs} + P_{mémoire} + P_{bus} + P_{horloge} + P_{contrôle}$$



- Après synthèse archi : ressources et accès définis → P = ΣPi
- Estimateur de haut niveau : après la sélection, Sel et V_{dd} connus, N_{reg} et N_{mem} obtenus par estimation probabiliste

II.6. GAUT_LP

Assignation/ordonnancement : Stabilisation des données à l 'entrée des opérateurs



Si un seul multiplieur

- List Scheduling

  {M1, M2, M3, M4}

  4 * Modèle 1

- Power Scheduling : favoriser la stabilité des variables

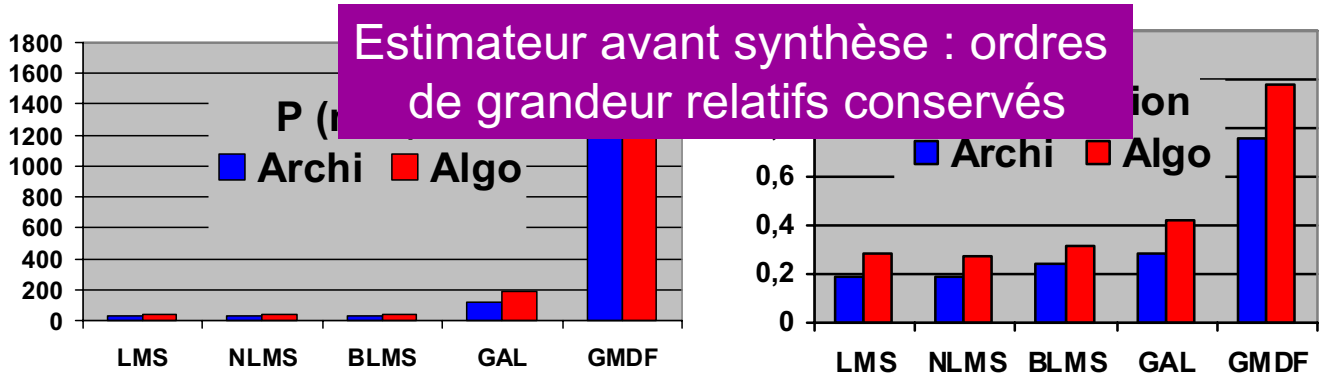
  {M1, M3, M2, M4}

  3 * Modèle 1, 1 * Modèle 2

II.6. GAUT_LP

Niveau de l'estimateur	Δ/estimateur logique	T_{CPU}
Logique (Compass)	-	1 h
Architectural	10%	10 mn
Algorithmique	20 à 30%	10 s

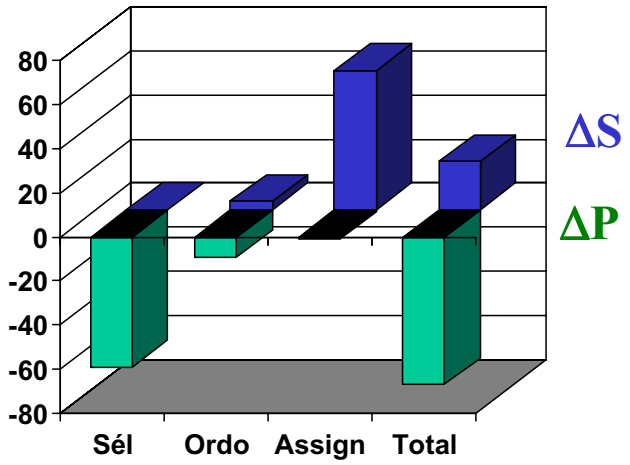
Estimateur avant synthèse : ordres de grandeur relatifs conservés



II.6. GAUT_LP

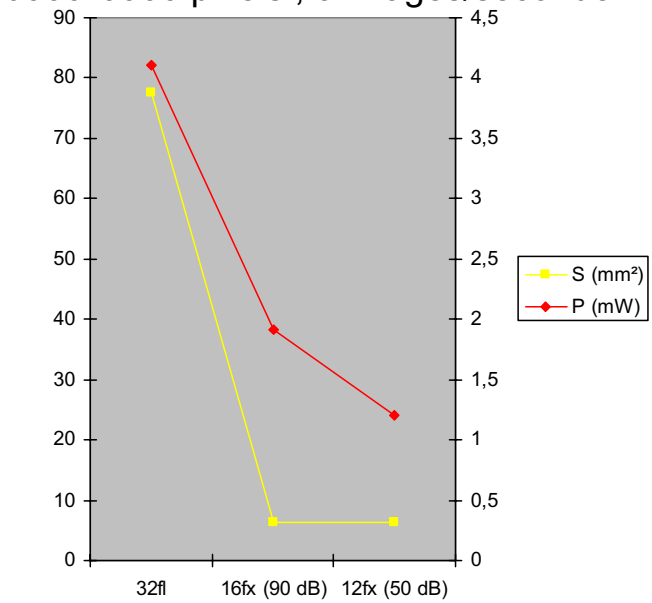
- GMDFα

128 points à 8 kHz



- TO : Format de données

6000*6000 pixels ; 9 images/seconde



II. Conclusion

- Liste non exhaustive : De nombreuses techniques d ’optimisation à différents niveaux
- Les stratégies doivent avoir peu d ’impact sur le flot de conception traditionnel
- Domaine de recherches très actif
- Besoins :
  - Outils et méthodes d ’estimation et d ’optimisation à haut niveau
  - Points chauds :
    - logiciel
    - mémoire

L.E.S.T.E.R
Laboratoire d'Electronique des Systèmes TEmps Réel

III. Le logiciel

III. Impact du logiciel

- 70% du coût de dévt des systèmes complexes
- La taille du code double tous les 2 ans
- 2 codes peuvent avoir les mêmes performances mais des consommations différentes
- Le plus efficace en E est d'améliorer la performance du code (ré-utilisation de nombreux travaux)
- Augmenter le parallélisme augmente les performances et la puissance mais diminue l'énergie
- les "power compilateurs" sont peu nombreux

III. Le logiciel

III.1. Estimation

III.1.a. Etat de l 'art

III.1.b. SoftExplorer

III.2. Optimisations

III. Le logiciel

III.1. Estimation

III.1.a. Etat de l 'art

III.1.b. SoftExplorer

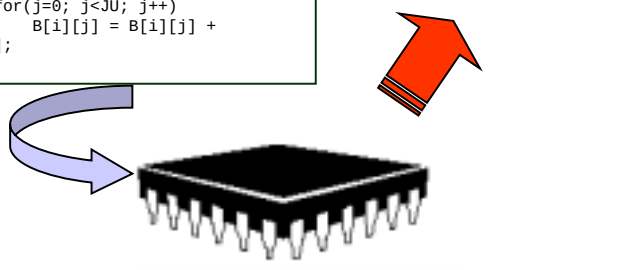
III.2. Optimisations

III.1.a. Problème

- Estimer la consommation d 'un programme sur un processeur

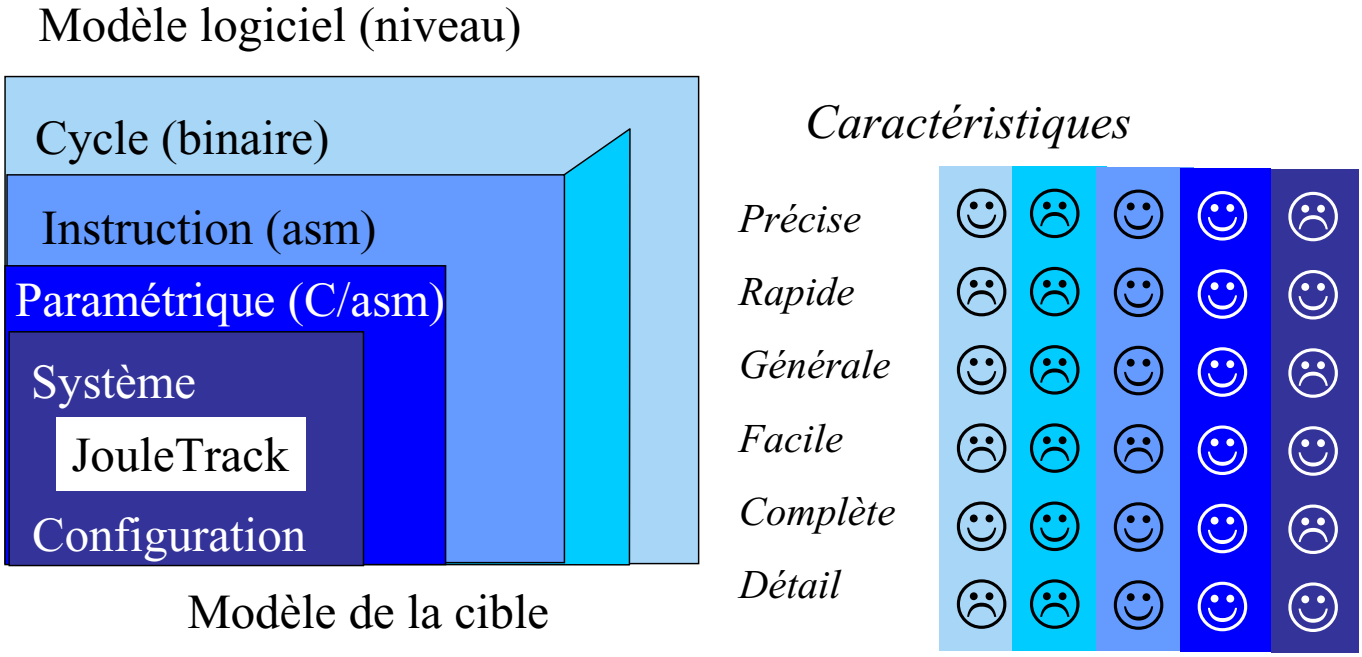
MODELE LOGICIEL

```
test1(int IU, int JU, int KU)
{
    int i, j, k;
    for(i=0; i<IU; i++)
        for(j=0; j<JU; j++) {
            for(k=8; k<KU; k++)
                A[k] = A[k-8];
            B[i][j] = B[i+1][j] + A[i];
        }
    for(i=0; i<IU; i++)
        for(j=0; j<JU; j++)
            B[i][j] = B[i][j] +
B[i+1][j];
}
```



MODELE CIBLE

III.1.a. Méthodes et outils



III. Le logiciel

III.1. Estimation

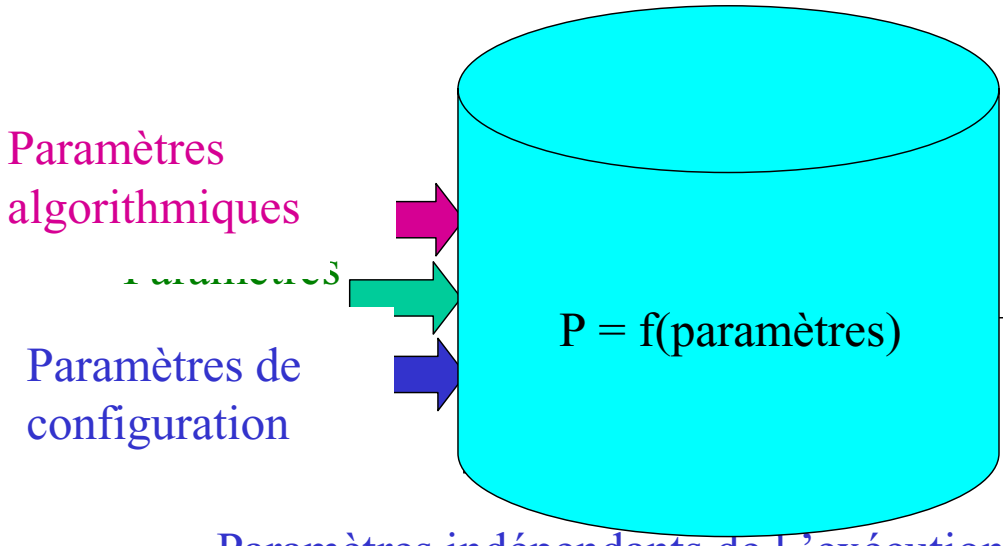
III.1.a. Etat de l 'art

III.1.b. SoftExplorer

III.2. Optimisations

III.1.b. Modèle du processeur

Etape 3 : détermination des lois de consommation



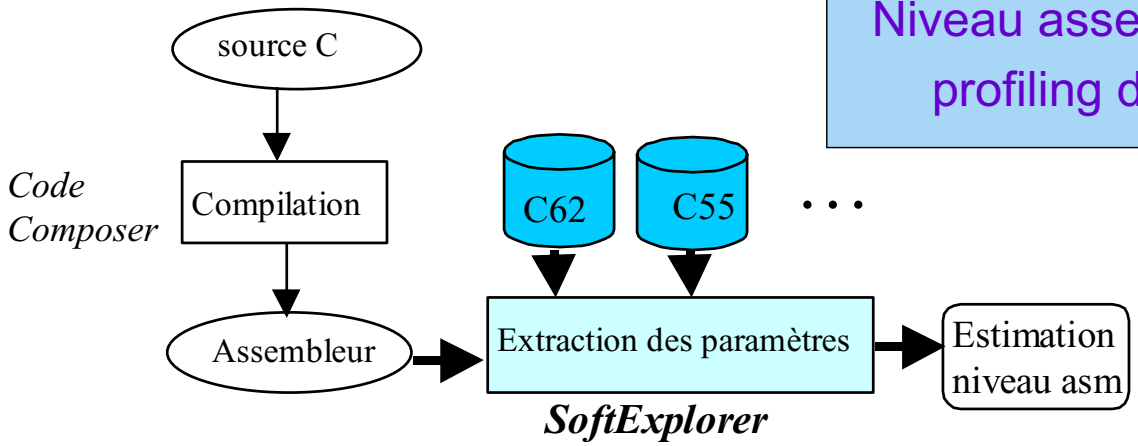
Paramètres indépendants de l'exécution

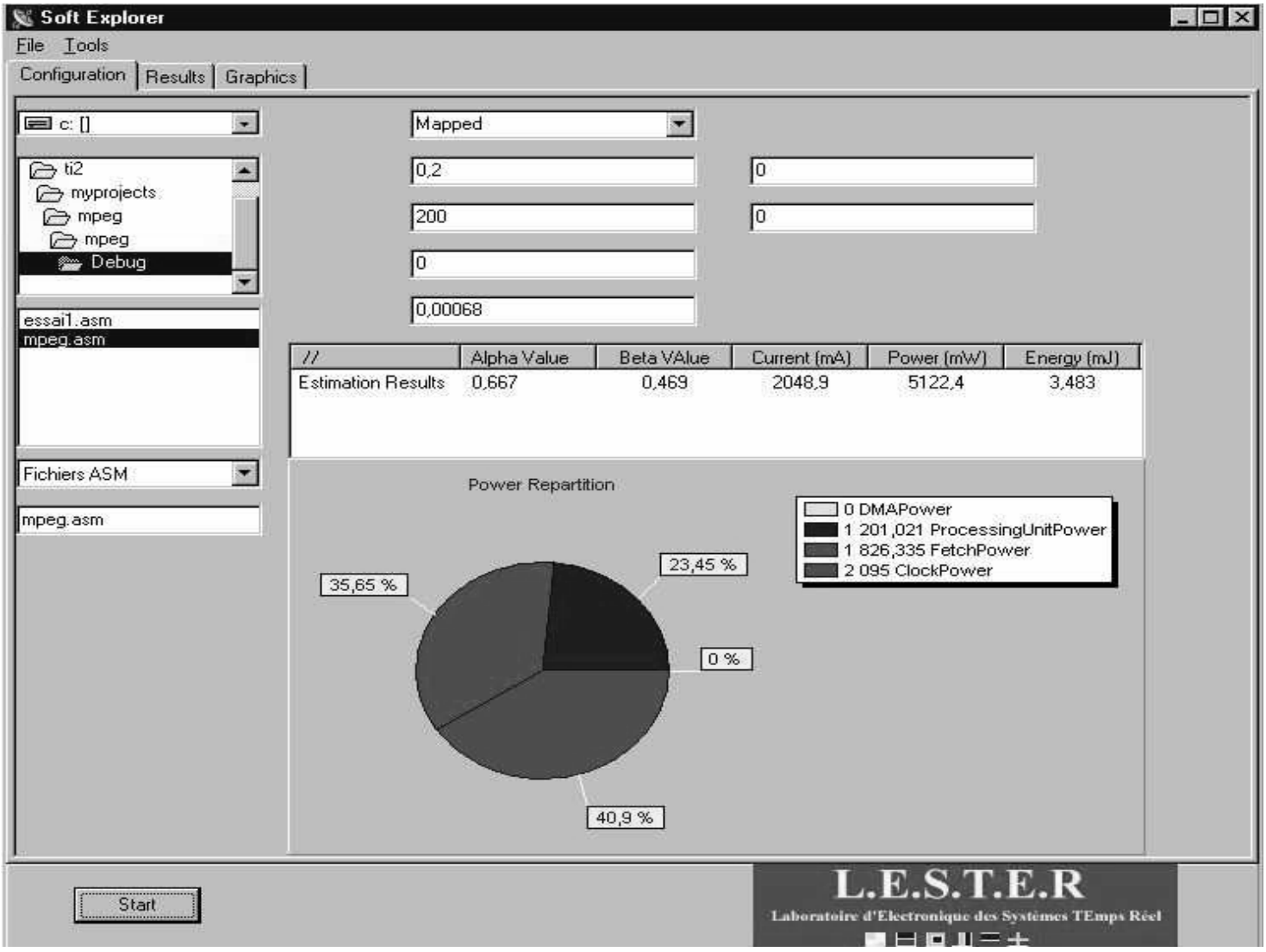
Modèle du processeur défini : valeurs des paramètres → P

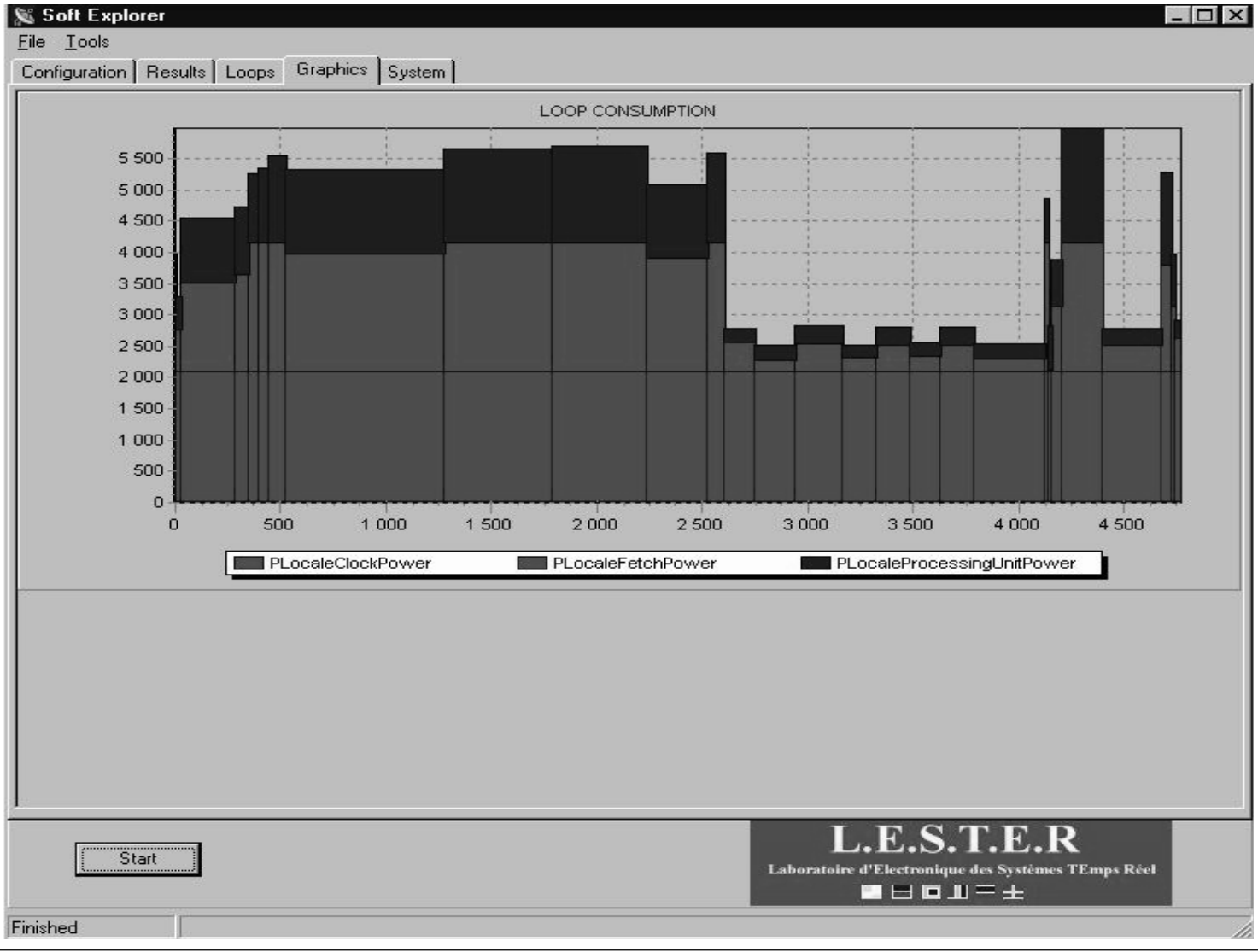
III.1.b. Estimation au niveau assembleur

- Paramètres de configuration : F, MM, DM, PM
  - connus avec l'application
- Paramètres algorithmiques : α, β, τ, γ, ε, PSR
  - valeurs à déterminer

Niveau assembleur
profiling du code




Soft Explorer
File Tools
Configuration Results Graphics
c: []
Mapped
0,2
0
200
0
0
0,00068
ti2
myprojects
mpeg
mpeg
Debug
essai1.asm
mpeg.asm
Fichiers ASM
mpeg.asm
// Alpha Value Beta VAlue Current (mA) Power (mW) Energy (mJ)
Estimation Results 0,667 0,469 2048,9 5122,4 3,483
Power Repartition
0 DMAPower
1 201,021 ProcessingUnitPower
1 826,335 FetchPower
2 095 ClockPower
35,65 %
23,45 %
0 %
40,9 %
Start
L.E.S.T.E.R
Laboratoire d'Electronique des Systèmes TEmps Réel

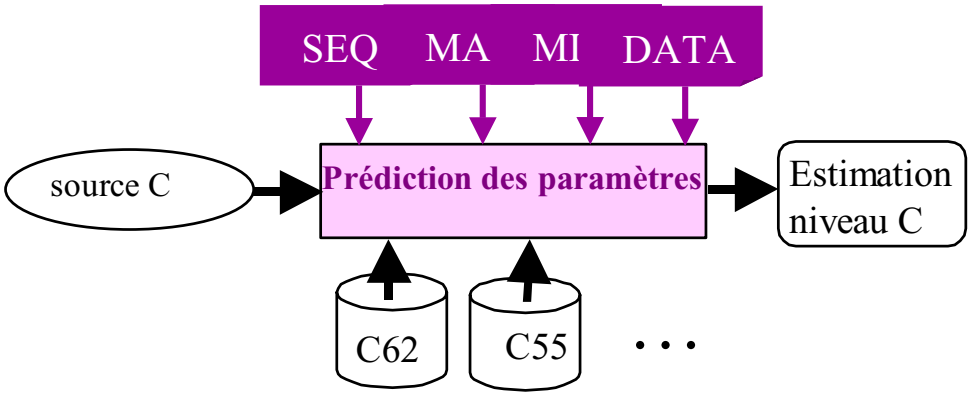

Soft Explorer
File Tools
Configuration Results Loops Graphics System
LOOP CONSUMPTION
5 500
5 000
4 500
4 000
3 500
3 000
2 500
2 000
1 500
1 000
500
0
0 500 1 000 1 500 2 000 2 500 3 000 3 500 4 000 4 500
PLocaleClockPower
PLocaleFetchPower
PLocaleProcessingUnitPower
Start
L.E.S.T.E.R
Laboratoire d'Electronique des Systèmes TEmps Réel
Finished

III.1.b. Estimation au niveau C

- But : donner rapidement au programmeur les caractéristiques de consommation de son algorithme
- Pas d 'analyse de code assembleur à réaliser.
- Pas d 'étape de compilation
- Pas d 'environnement à acheter (compilateur)
- Adéquation-Algorithme-Architecture
  - Choix du meilleur algorithme sur une cible donnée pour une contraintes de consommation
  - Choix de la meilleure cible pour une application donnée (bibliothèque de composants)

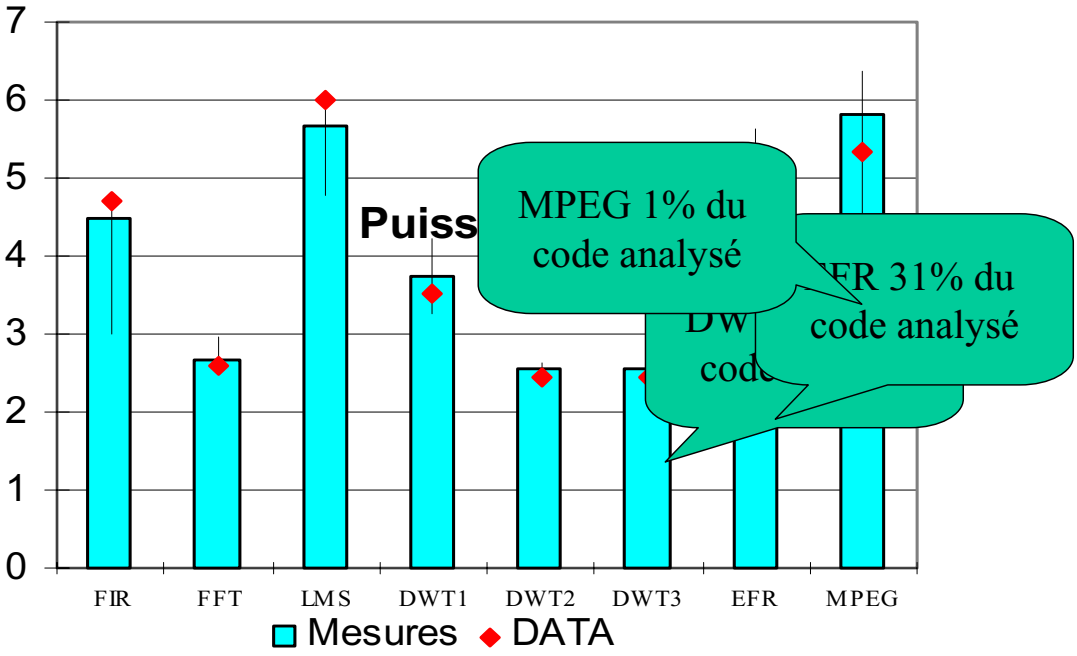
III.1.b. Estimation au niveau C

- Paramètres algorithmiques : valeurs à déterminer
  - Niveau C sans compilation
    - même modèle du processeur
    - Modèles de prédiction de l 'efficacité du compilateur



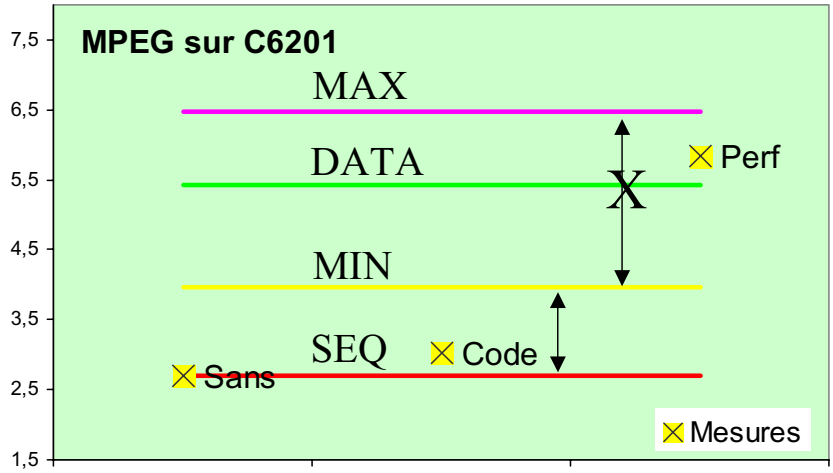
III.1.b. Niveau C

- Validation de la précision : Erreur max = 8% /mesures



III.1.b. Options de compilation

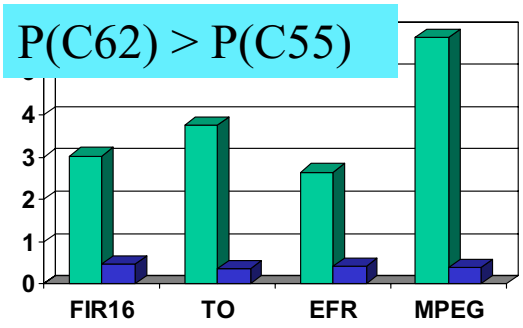
- Applis TDSI - erreur max/mesures :
  - niveau assembleur : 4%
  - niveau C : 8%
- Modèles / options de compilations

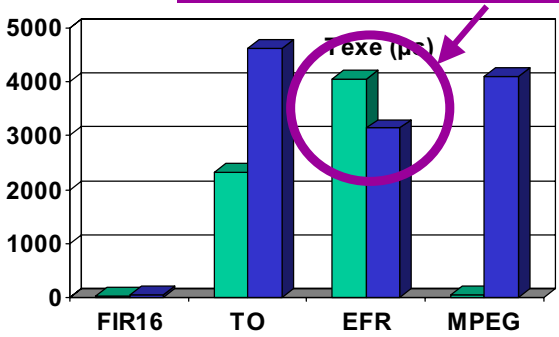


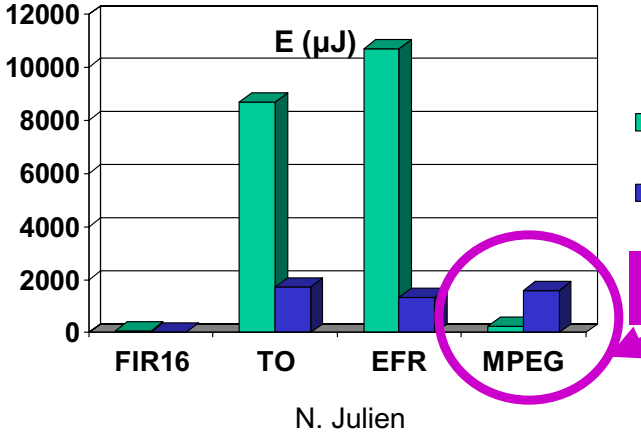
III.1.b. Utilisation

T(C62) > T(C55) !

P(C62) > P(C55)



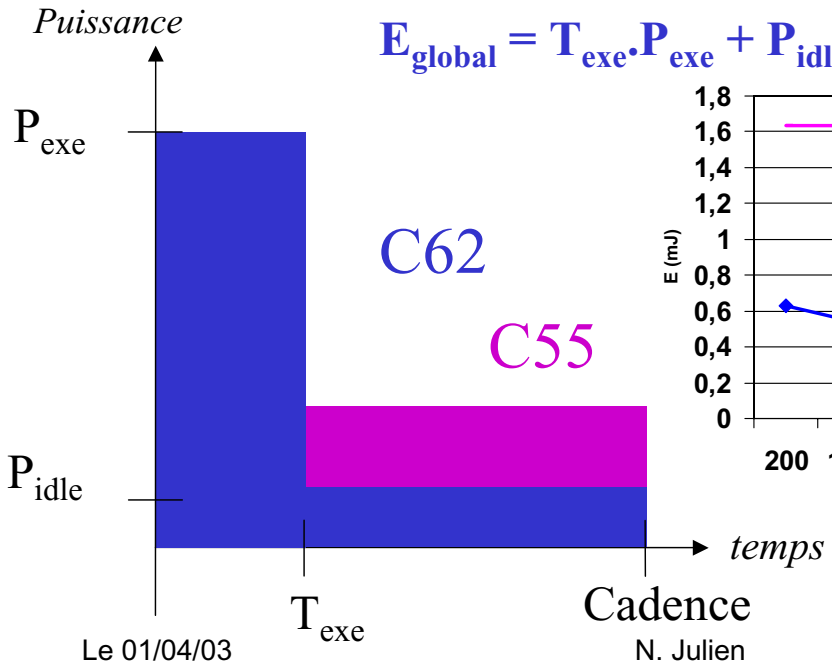




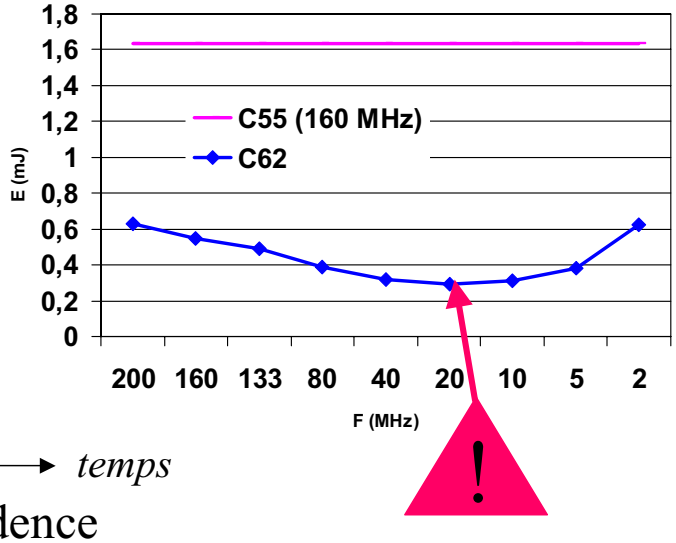
E(C62) < E(C55) ?

III.1.b. Utilisation

- MPEG cadencé



$$E_{global} = T_{exe}.P_{exe} + P_{idle}.(Cadence - T_{exe})$$



III. Le logiciel

III.1. Estimation

III.1.a. Etat de l 'art

III.1.b. SoftExplorer

III.2. Optimisations

III.2. Optimisations logicielles

- Choisir des algorithmes réguliers
- Diminuer l'activité
  - en diminuant la taille (élimination des sous-expressions communes, compression de code)
  - ou le nombre d'opérations et les accès mémoire
  - ou le coût des opérations (transfo algébriques, réduction des LD/ST, sélection de code)
- Augmenter la vitesse de traitement pour diminuer Vdd
  - pipeline fonctionnel,
  - mise en ligne des fonctions,
  - transformations de boucles,...
- Améliorer la gestion des données
  - réduire les accès mémoire Data reuse (*IMEC, Pennsylvania*)
  - améliorer l 'utilisation des registres Data forwarding (*Milano*)
  - réduire la commutation des lignes d'adresse
  - exploiter la localité des données (mémoire)

III.2. Optimisations logicielles

- Effets sur E et P parfois difficiles à prédire et liés à l'architecture (*IMEC, Irvine, Univ. Georgia, Univ. Austin*)
- améliore perf et E : réduction des loads et stores, déroulage de boucle (augmente P), mise en ligne des procédures, transformations de boucles, élimination des sous-expressions communes.
- réduit E : ordonnancement, register pipelining, code selection (remplace une instruction dissipative)
- Transformations de haut niveau :
  - échange/inversion de boucles (augmente la localité des données)
  - déroulage de boucle (lié à l 'archi du processeur)
  - loop tiling : ajuste à la taille du cache
  - fusion/fission de boucles
  - extraction des invariants de boucles

IV. Mémoires

IV. Les mémoires

IV.1. Consommation des mémoires

IV.2. Mémoires spécifiques

IV.3. Optimisations mémoire

L.E.S.T.E.R
Laboratoire d'Electronique des Systèmes TEmps Réel

IV. Les mémoires

IV.1. Consommation des mémoires

IV.2. Mémoires spécifiques

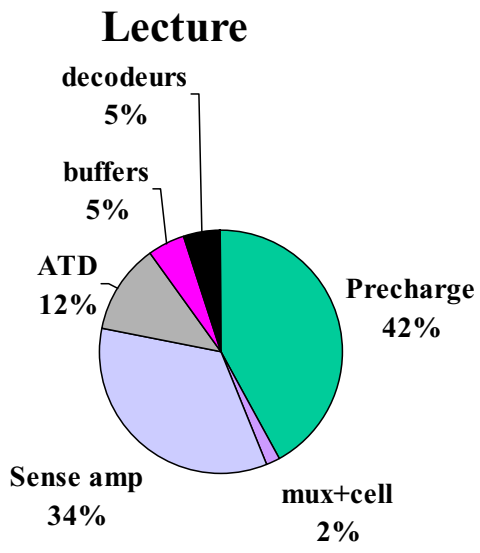
IV.3. Optimisations mémoire

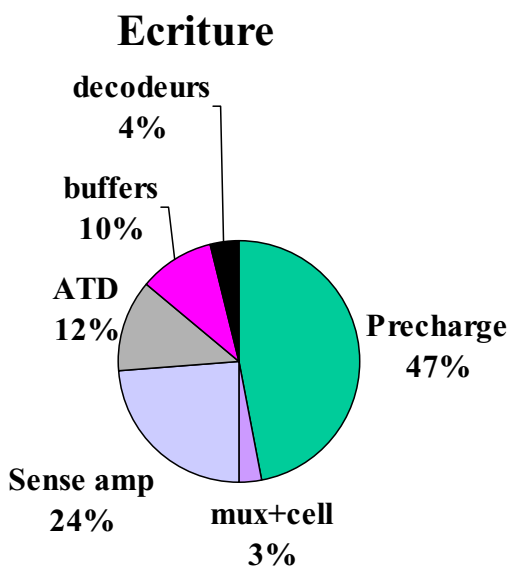
IV.1. Consommation des mémoires

- Applis TDSI : 20 à 80% de la consommation globale
- On-chip/Off-chip sur ARM instruction Load 32-bit I/ 3 ; E/ 7
- TO sur TIC6201
  - E(données en mémoire externe) = 3,6 x E(données en mémoire interne)
- P_{dyn} = f (#accès mémoire, vitesse, taille)
  - réduite par partionnement
- $P_{statique}$ liée à $I_{leakage}$ = f (taille)
  - importante pour les SRAM rapides en techno faible
  - réduite par mise en veille

IV.1. Consommation des mémoires

- SRAM embarquée : Répartition d 'énergie





IV.1. Consommation des mémoires

- Choix de mémoire
  - métriques pour évaluer les performances des mémoires
  - MOSYS : pour différents types de mémoires
  - Memory Technology Coefficient =Fmax (MHz) x densité (Mbits/mm²)/ P normalisée (mW/MHz/Mbit)
  - Dolphin : pour un même type de mémoire
    - Figure of Merit

$$FoM = \frac{\frac{1}{temps\_d'accès(ns)} x \frac{\Delta V}{puissance(mA/MHz)}}{\frac{surface(mm^2)}{capacité(Mbits)}}$$

  - Validité des comparaisons
    - se placer dans les même conditions pour toutes les mémoires.
    - métriques pondérées par process

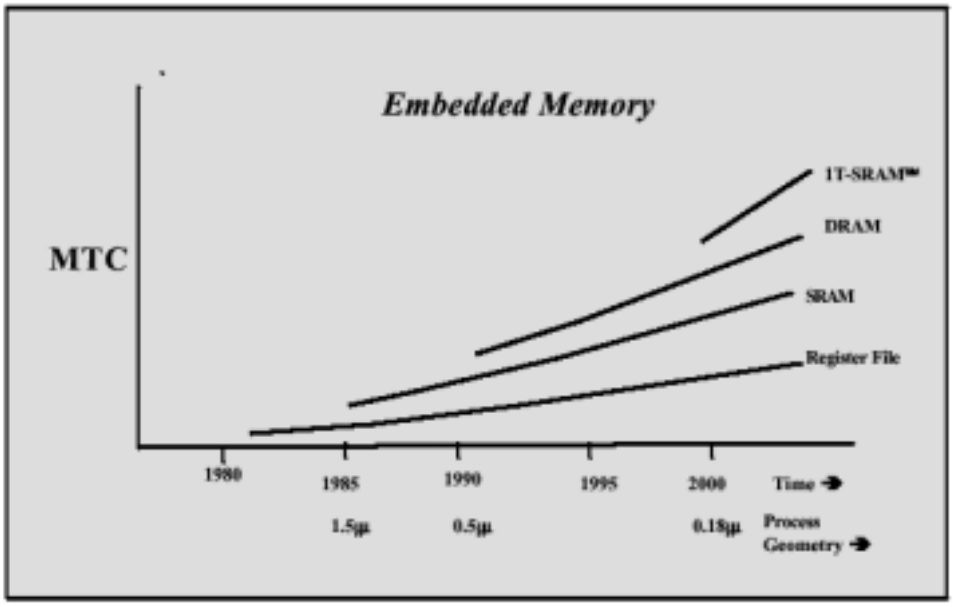
IV.1. Consommation des mémoires

Puissance normalisée des mémoires

		Taille Mb	V_{dd} V	P W /*standby*	F MHz	µW/(Mb MHz)
SRAM	Mitsubishi	2.25	3.3	1.2	350	1524
	Intel	18	1.5	8.1 m	750	0.6
	IBM	8	2.5	1.9	470	505
	NTT	1	1	35m	200	175
	Hitachi	1	1.8	4.2	1800	2333
	Fujitsu	0.288	1.8	280m	700	1389
SRAM low power	Hynix	1	3	60m/6µ	18	3333
	Cypress	1	3	150m/30 µ	20	7500
	NEC	1	3	210m/60µ	14	15000
	Toshiba	1	3	195m/75µ	14	13929
SDRAM	Hynix	256		0.7	133	20.6
	Micron	256		1	133	29.4
	Elpida	256		1	133	29.7

IV.1. Consommation des mémoires

Evaluation des mémoires embarquées



IV. Les mémoires

IV. 2. Mémoires spécifiques

- mémoires caches : flexibles mais consommatrices
- gestion du cache : diminution des accès externes (T_{exe} et E)
- scratch-pad : SRAM localisée sur laquelle on mappe les variables les plus fréquemment accédées
- placement statique ou dynamique
- effet : diminuer le nombre d 'accès en mémoire externe et diminuer les conflits avec cache
- problème : quel segment (tableau, boucle…) placer ?
- placement statique ou dynamique

IV. 2. Mémoires spécifiques

- Placement statique :
  - ARM Cache/scratchpad : +2,5 à 4 fois en E/accés (*Dortmund*)
  - scratchpad/cache : 12% à 43% de réduction E
  - (*Bologna Torino)* sur cœur ARM : -12% à -68% en E
- Placement dynamique (LESTER) :
  - Transformée de Hadamard : multiplication de matrices de taille supérieure à la mémoire interne données
  - DSP Texas Intrument C6201B ; Ucoeur=2,5 V Upériph =3,3 V
  - SDRAM externe @100MHz ; I lecture/écriture: 120mA

	Pcoeur W	Texe ms	Emem-ext mJ	Etotale mJ	ΔE
externe	3,19	1,88	3,97	9,97	+ 60%
mixte	3, 23	1,31	3,08	7,31	+ 27%
optimisé	3, 26	1,11	2,20	5,77	-

IV. Les mémoires

IV.1. Consommation des mémoires

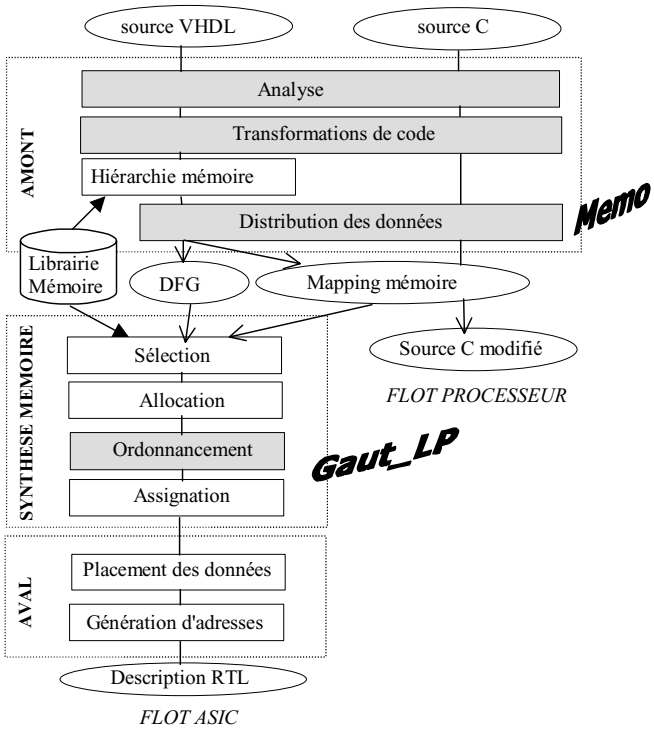
IV.2. Mémoires spécifiques

IV.3. Optimisations mémoire

IV.3. Optimisations mémoire

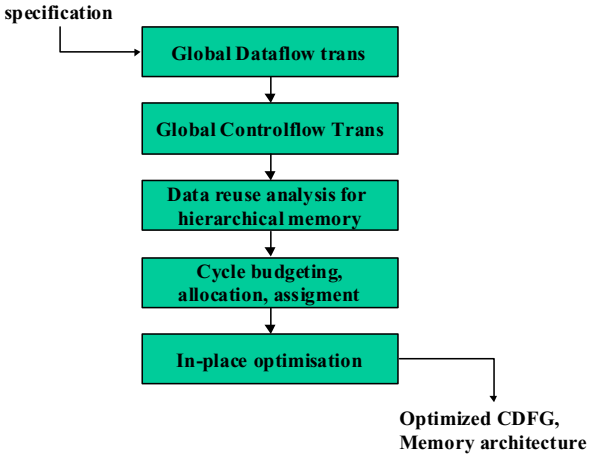
- La réduction de la consommation de la mémoire s 'appuie sur :
  - L 'exploitation de la localité spatiale et temporelle.
  - La réduction des transferts.
  - La réduction des transitions sur les bus d 'adresse.
- Les différentes méthodologies développées :
  - Optimisation de l 'architecture mémoire pour une séquence d 'accès mémoire donnée (*Irvine, Torino*)
  - Optimisation des accès mémoire, architecture mémoire connue (*Dortmund, Bologna, LESTER*)
  - Optimisation conjointe des accès mémoire et de l 'architecture (*IMEC, LESTER*)

IV.3. Approche LESTER



IV.3. Approche IMEC

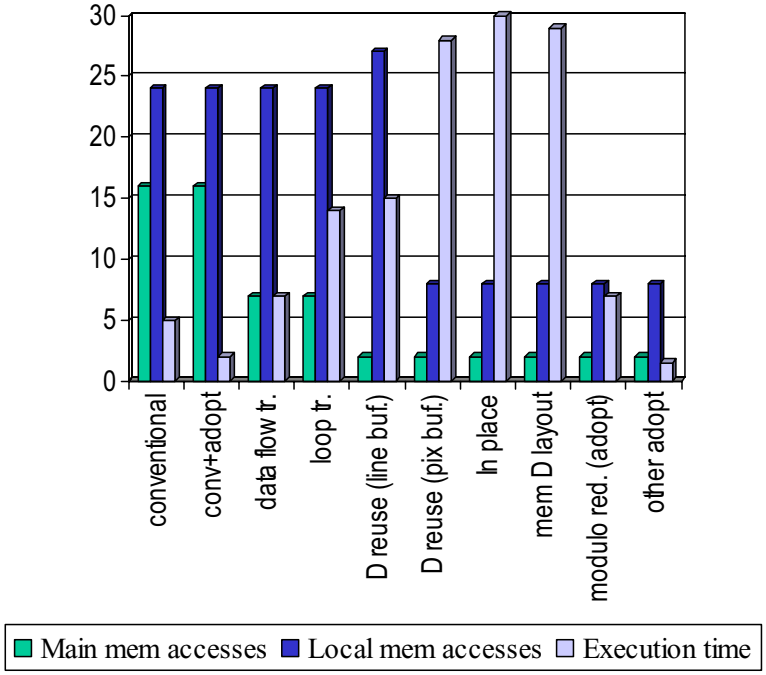
- DTSE Data Transfer & Storage Exploration IMEC
  - Loop transformations : améliorer localité et régularité
  - Data reuse exploration : réécriture de code
    - hiérarchie mémoire
  - Storage Cycle Budget Distribution pour containte temps-réel
  - Memory Allocation and Assignment
  - Data layout organization optimisation de l'organisation mémoire
  - Address optimization



IV.3. Approche IMEC

- Results on cavity detection (Execution time on pentium-mmx)

After DTSE:
local accesses / 3
memory size /5
power /5
system bus load /12
before ADOPT:
performances x6



L.E.S.T.E.R
Laboratoire d'Electronique des Systèmes TEmps Réel
UBS

IV.3. Difficultés

- Détermination de la hiérarchie et architecture mémoire
- Caractérisation des mémoires
- Nombreux paramètres interdépendants
  - Technologiques (type, process…)
  - Architecturaux (taille, nombre de bancs…)
  - Algorithmiques (nombre d'accès en lecture, écriture…)

Bibliographie

- T. Mudge « Power: A First-Class Architectural Design Constraint » *Computer vol. 34 n°4, April 2001, pp52-57*
- J. M. Rabaey, M. Pedram « Low Power Design Methodologies » *Kluwer Academic Publishers 1996*
- A. Raguhathan, N.K. Jha, S. Dey « High-level Power Analysis and Optimization » *Kluwer Academic Publishers 1998*
- A. Guyot, S. Abou-Samra « Conception pour la faible consommation » *cours DEA Microélectronique*

  http://tima-cmp.image.fr/~guyot/Cours/Basseconso/
- F. Parain, M. Banâtre, G. Cabillic, T. Higuera, V. Issarny, J.P. Lesot « Techniques de réduction de la consommation dans les systèmes embarqués temps-réel » *publication interne IRISA N°1332 Mai 2000*
- *travaux du LESTER* http://lester.univ-ubs.fr:8080/
- et de nombreux articles...