LEAT
LABORATOIRE D'ELECTRONIQUE ANTENNES ET TELECOMMUNICATIONS

Archi09

Gestion système de la consommation énergétique dans un SoC : systèmes monoprocesseur, multiprocesseur

C. Belleudy (belleudy@unice.fr)

Avec la collaboration de : M. Auguin, S. Bilavarn, K. Bhatti, A. Castagnetti

cnrs dépasser les frontières

Université Nice SOPHIA ANTIPOLIS

PLAN

Système basse consommation ?

- Nombre de piles (par an) mises sur le marché européen : **160 millions de tonnes**
- *Ces piles contiennent des métaux qui sont polluants pour l'environnement en fin de leur cycle vie.*

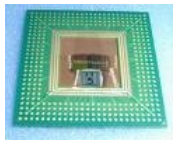
→ *Contrainte énergétique et de puissance*

↙ ↘

Budget énergie par scénario applicatif => puissance moyenne

Système basse consommation

- Packaging : -20% on P =>-30% (le packaging représente jusqu'à 1/3 du coût de l'IC)
- Alimentation à découpage,
- Système complet : boîtier à moindre coût.



(Source THOMSON)

Contrainte sur la puissance

↙ Pic de puissance

↘ Puissance moyenne

Ordonnancement temps réel

Système temps réel : Application => ensemble de tâches ti caractérisées par :

C_i : Worst Case Execution Time (WCET), T_i : Period of the task i,

A_i : Actual (real) execution time of the task i,

Problème : Ordonnancer (définition des dates d'exécution) afin de satisfaire les contraintes temporelles + minimiser la consommation.

Technique Hors ligne (Statique)

Technique En ligne (dynamic)

Contrainte temporelle Hard, soft, best effort

Téléphone mobile : lecture vidéo

⇒ Audio : 96 kbps

⇒ vidéo : 30 fps (QVGA)

Vidéo : Tv : 1/30 s, Cv : 20 ms
Audio : Ta : 1/44100 s, Ca : 7 µs
Accrochage réseau :
Tr : 1/21 s, Cr : 577 µs

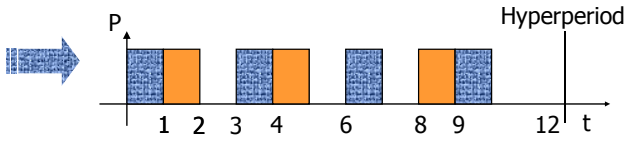
Exemple de politique d'ordonnancement : EDF

Principe de la politique EDF (Earliest Deadline first) [Gr02] :
Priorité à la tâche dont l'échéance est la plus proche.

$$\text{Test d'ordonnancement pour N tâches : } U = \sum_{i=1}^{N} \frac{Ci}{Ti} \leq 1$$

Hors ligne

Task	WCET	Periode
T1	1	3
T2	1	4

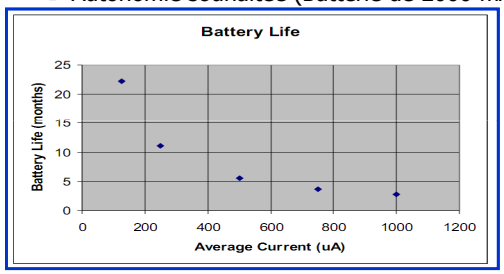


$$U = \frac{1}{3} + \frac{1}{4} = \frac{7}{12} < 1$$

Cas multiprocesseur : <m

Ordonnancement + Contraintes énergétiques

- Contrainte d'énergie et/ou de puissance :
  Exemple : Réseau de capteur sans fil
  - Autonomie souhaitée (Batterie de 2000 mAh)



Cellule solaire :
- plein soleil : 4 mW/cm²
- à l'intérieur : 3 µW/cm²

Exemple : Téléphone 3G : batterie 900mA, 9h en vidéo !!!

  - Par scénario applicatif => E, P
- Packaging, … : Pmoyen, Ppeak à ne pas dépasser
- Aucune contrainte => déterminer la solution qui consomme le moins pour une application ou un jeu d'application donné.

Problématique : ordonnancement basse consommation

Système temps réel:

Ensemble de tâches ti caractérisées par :

Ci : Worst Case Execution Time (WCET) :

Ai : Actual (real) execution time of the task i,

Ti : Period of the task i,

Problème : Ordonnancer afin de satisfaire les contraintes temporelles et de minimiser la consommation en sélectionnant :

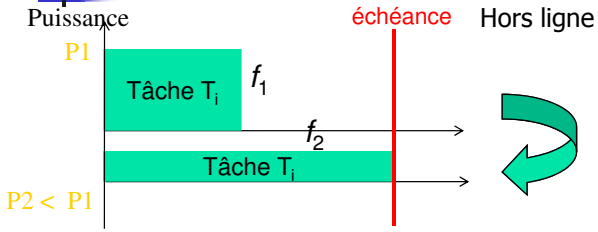
- la vitesse de fonctionnement adéquate ou
- un mode repos du processeur.

Vitesse globale

Vitesse Locale

Modes repos

VFS (Voltage and Frequency Scaling): principe



$f_2 < f_1$ ⇒ $V_2 < V_1$

Facteur de ralentissement : $S = f_1 / f_2$

$$C_i' = C_i . S$$

$$\text{VFS global} : \sum_{i=1}^{N} \frac{Ci'}{Ti} = S * \sum_{i=1}^{N} \frac{Ci}{Ti} = S.U \le 1 \Rightarrow S \le \frac{1}{U}.$$

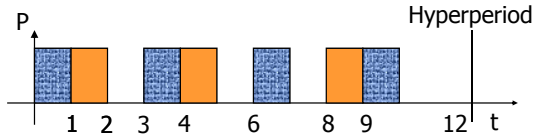
Même principe en local

$$\text{En ligne Ci remplacé par Ai} : U = \sum_{i=1}^{K} \frac{Ai}{Ti} + \sum_{i=k+1}^{N} \frac{Ci}{Ti} \le 1$$

⇒ Estimer les gains sur E, P
P= F(paramètres applicatifs, V, F)

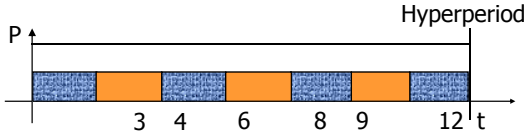
Ordonnancement EDF + VFS

Task	WCET	Periode
T1	1	3
T2	1	4



$$U = \frac{1}{3} + \frac{1}{4} = \frac{7}{12} < 1$$

$$S \le \frac{1}{0,583} \cong 1.7$$

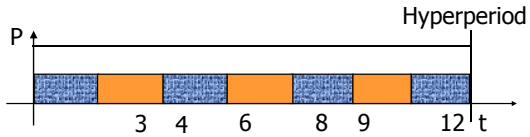


⇒ Gain ?

Ordonnancement EDF + DVFS

Task	WCET	Periode
T_1	1	3
T_2	1	4

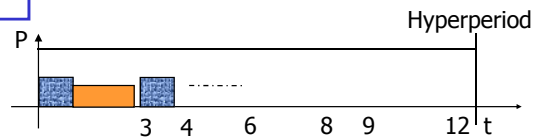
S=1,7



$$U = \frac{0,75}{3} + \frac{1}{4} = \frac{6}{12}$$

A_{T1}=0,75

$$S \le \frac{1}{0.5} \cong 2$$

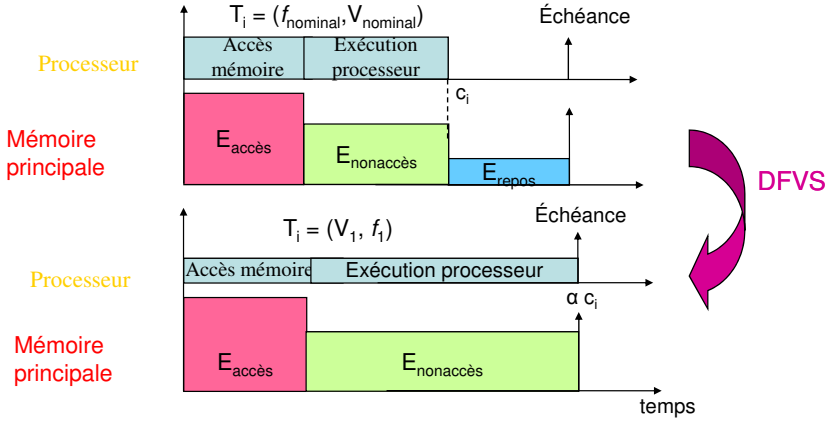


⇒ Gain ?

Evolution du WCET en fonction de F

Temps d'exécution x Fnominal/Fnew

⇒ accès mémoire ??



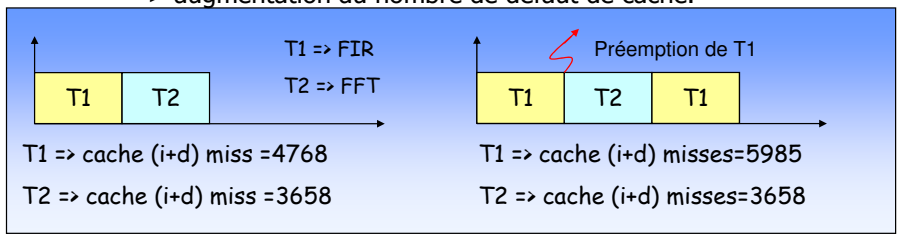
DVFS et préemption ?

Prolonger le temps d'exécution augmente le nombre de préemption

Cout d'une préemption :

- Changement de contexte : sauvegarde des registres et mise à jour du TCB (Task Control Block)
- Exemple : sous linux => 2 à 30 *µsec pour un ordinateur personnel
- Cache et TLB (Translation Look-aside Buffers : translation d'adresses virtuelles en adresses physiques) => pertes des références locales

=> augmentation du nombre de défaut de cache.



DVFS, DPM : principe et modèles

- Mode repos des processeurs :

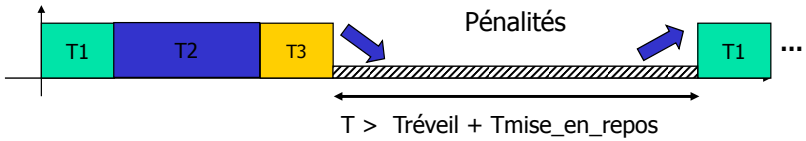
  On coupe la tension d'alimentation de certaines parties du processeur

  Exemple pour le XScale :

  Idle : les données dans le cache sont sauvegardées,

  Sleep : les données dans le cache sont perdues.

- Pénalités de réveil

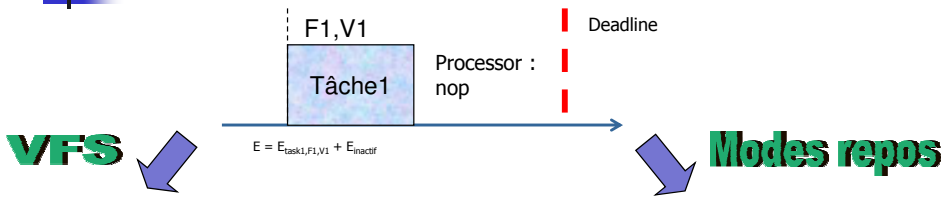


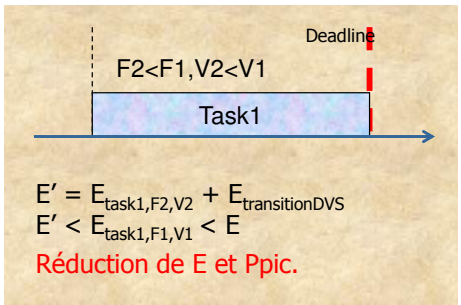
T > Tréveil + Tmise_en_repos

Données dans le cache ?

$$\text{VFS Global + DPM : } S * \sum_{i=1}^{N} \frac{Ci}{Ti} + \sum_{j=1}^{m} C_{lp\_\text{mod}ej} \leq 1.$$

VFS ou low power mode ?

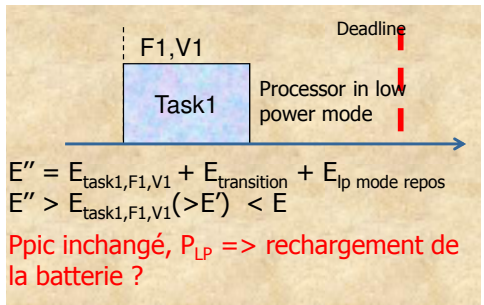




$E' = E_{task1,F2,V2} + E_{transitionDVS}$

$E' < E_{task1,F1,V1} < E$

Réduction de E et Ppic.



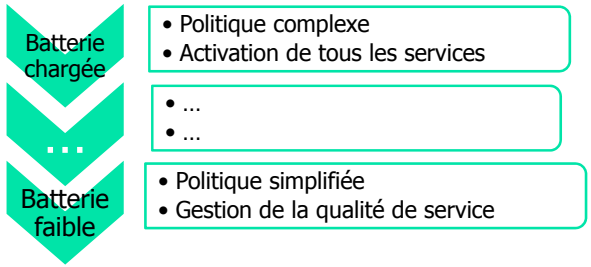
$E'' = E_{task1,F1,V1} + E_{transition} + E_{lp\ mode\ repos}$

$E'' > E_{task1,F1,V1} (>E') < E$

Ppic inchangé, P_{LP} => rechargement de la batterie ?

Batterie

- Modèle de décharge de batterie
  - loi de peukert $C = I^n.T$ avec I courant de décharge de la batterie, n : constante spécifique à la batterie, T : temps de décharge,
- Monitoring de la batterie,
- Adaptation de la politique en fonction de la charge de la batterie, modification de la plage de variation des paramètres processeurs (exemple : limitation de la fréquence, …), …



PLAN

1. Problématique
2. Systèmes monoprocesseur
   1. Modèle énergétique de processeur
   2. Consommation de la mémoire
   3. Stratégies basse consommation
3. Systèmes multiprocesseur
   1. Exemple de modèle de consommation
   2. Ordonnancement basse consommation
   3. Optimisation de la consommation mémoire

Objectifs

Ordonnancement basse consommation :
Nécessite des Modèles : puissance, temps, énergie,

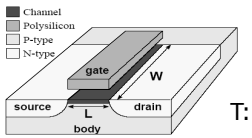
Processeur + Cache :
- **En fonction du couple tension/fréquence**
- **modes basse consommation (repos)** : idle, sleep …
- **Pénalités de changement de mode : T_{entry}, T_{exit}, P**

Mémoire principale
- **Mode actif**
- **Modes basse consommation.**
- **Pénalités de changement de modes**

VFS => Gain en consommation ? (E, P)

$$P = P_{static} + P_{dynamic}$$

$P_{static} \sim V_{dd} \cdot I_{leak}$
avec
$I_{leak} \sim (W/L)\, e^{(-V_{TH}/T)}$



T: température

$P_{dynamic} \sim C_l \cdot V_{dd}^2 \cdot F$

Changement de couple (V,F) :
Gain en $(V_1/V_2)^2.(f_1/f_2)$ sur la puissance
Et en $(V_1/V_2)^2$ sur l'énergie

Sur Pstatic : Gain en V_1/V_2
Sur l'énergie ?

VFS => Gain en consommation ? (E, P)

- Exemple de couple V,F :
  - 1.55 V, 624 Mhz
  - 0.9 V, 104 Mhz
- Gain attendus :
  - $P_{dynamic}$: facteur de réduction $(1.55/0.9)^2 x6 = 17.73$
  - P_{static} : Facteur de réduction $(1.55/0.9) = 1.72$
  - $E_{dynamic}$: Facteur de réduction : $(1.55/0.9)^2 = 2.96$
  - E_{static} : facteur d'augmentation : $(0.9/1.55)x6 = 3.48$

=> Gain sur E tant que $P_{dynamic}/P_{static} > 3,75$.
(=> attention : cas idéal, processeur seul)

Quelques modèles

- Pour un couple V, F [Ib08][Softexplorer][…]:
  - I = F(taux de cache miss, de rupture de pipeline, taux de parallélisme, …)
  - fonction généralement linéaire
- En tenant compte de V, F :
  - On évalue les constantes pour différents couples V,F [Gi05]

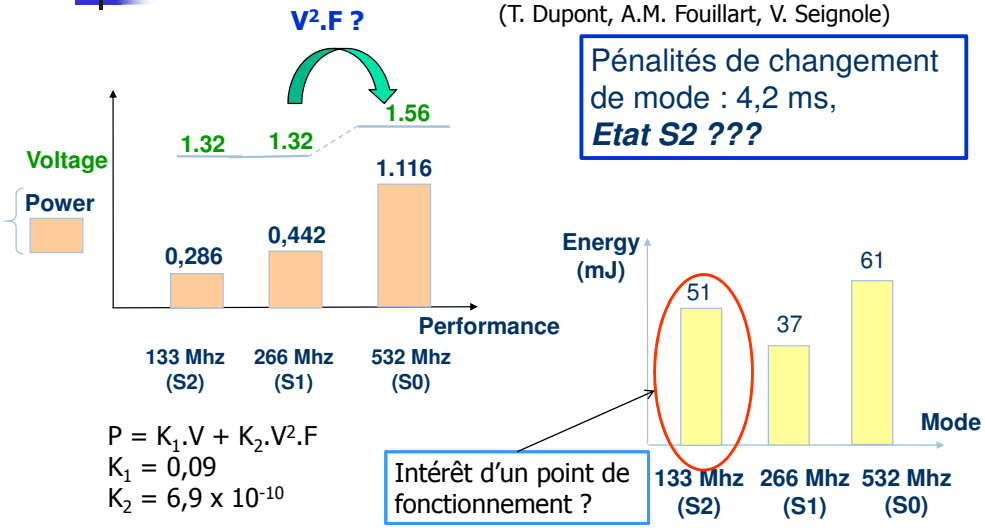
$$Power_{cpu} = \alpha_1(IFetch_{miss}) + \alpha_2(DataDep) + \alpha_3(DataTLB_{miss}) + \alpha_4(InstTLB_{miss}) + \alpha_5(InstExec) + K_{cpu} \quad (1)$$

Parameter	Core Voltage/frequency		
	1.0V 200Mhz	1.1V 300Mhz	1.3V 400Mhz
α_1	1.12×10^{-6}	1.62×10^{-6}	2.30×10^{-6}
α_2	1.11×10^{-8}	1.50×10^{-8}	2.53×10^{-8}
α_3	2.37×10^{-8}	2.42×10^{-7}	1.34×10^{-5}
α_4	4.78×10^{-7}	8.98×10^{-7}	2.11×10^{-6}
α_5	3.83×10^{-8}	5.38×10^{-8}	8.17×10^{-8}
β_1	2.21×10^{-6}	1.72×10^{-6}	2.14×10^{-6}
β_2	2.19×10^{-8}	1.53×10^{-8}	1.33×10^{-8}
K_{cpu}	0.08	0.115	0.165
K_{mem}	0.055	0.055	0.055

Processeur : PXA255

ARM1136 Processor (IMX31) - THALES

(T. Dupont, A.M. Fouillart, V. Seignole)



Pénalités de changement de mode : 4,2 ms, ***Etat S2 ???***

$P = K_1.V + K_2.V^2.F$
$K_1 = 0,09$
$K_2 = 6,9 \times 10^{-10}$

Intérêt d'un point de fonctionnement ?

Processeur RISC 32 bits (exemple ARM1176)

Plage de variation de :

- 0,95V à 1,45V => 256 valeurs,
- 50 Mhz à 400Mhz =>16 valeurs.

⇒ Définition de points de fonctionnement ? Combien ?
⇒ Complexité OS

Quelques valeurs :

(1,32V, 300 MHz) => P= 633 mW
1,412 ↓ 1,52
(1,20V, 250 MHz) => P= 415 mW
1,54 ↓ 1,54
(1,08V, 200 MHz) => P= 270 mW
1,72 ↓ 1,69
(0,95V, 150 MHz) => P= 160 mW

$P_0 = K_{arm1}.V^2.F$

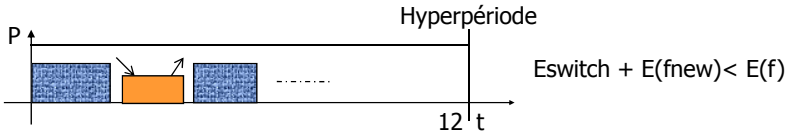
Mode repos des processeurs

Mode	Puissance	Pénalités temporelles
Active	116 mW (104 Mhz, 0.9V) à 926 mW (624 Mhz, 1.55V)	
Idle	64 mw (104 Mhz) à 260 mW (624 MHz)	Tentry=Texit = 1µs
Standby	1.7 mW	Tentry= 0.34 ms Texit= 11.28 ms
Sleep	0.16 mW	Tentry: 2.5 ms Texit: 136 ms
Deep Sleep	0.101 mW	Tentry : 2.66ms Texit =261ms

Processeur PXA 270 (Données constructeur)

Changement de couple V,F ?

- Tenir compte des pénalités de changement de fréquence
  - constant,
  - Variable. $O_i = C + K\left|f_{i+1} - f_i\right|$
- Changement de fréquence bénéfique si :



De même pour la mise en veille d'un processeur (ou d'une mémoire)

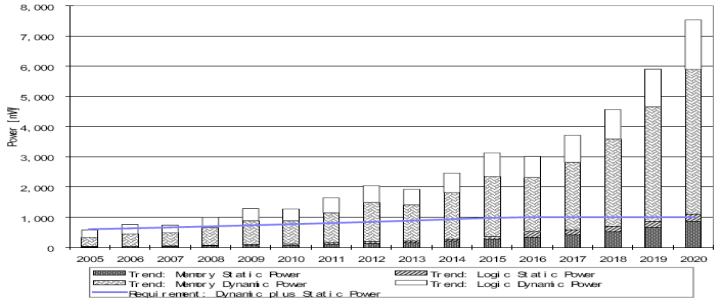
Consommation mémoire

• Quantité de données importante pour les nouvelles applications (multimédia, traitement d'image,…)

• Surface dédiée à la mémoire => augmentation

Samsung: « Un téléphone mobile qui n'avait pratiquement pas de mémoire DRAM en 2002, possède une mémoire dépassant les 20 MB en 2006»

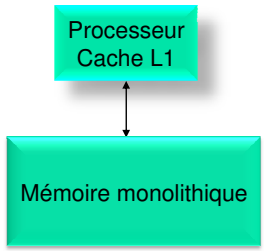
ITRS Roadmap 2005

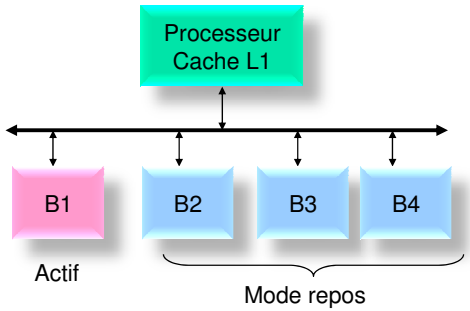


PLAN

1. Problématique
2. Systèmes monoprocesseur
   1. Modèle énergétique du processeur
   2. Modèle énergétique de la mémoire
   3. Stratégies basse consommation
3. Systèmes multiprocesseur
   1. Exemple de modèle de consommation
   2. Ordonnancement basse consommation
   3. Optimisation de la consommation mémoire

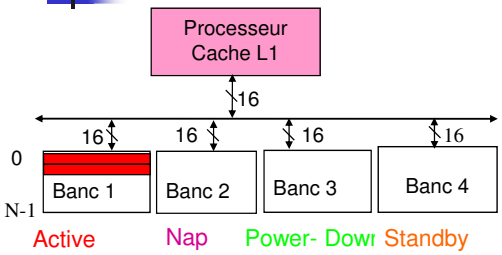
Architecture mémoire



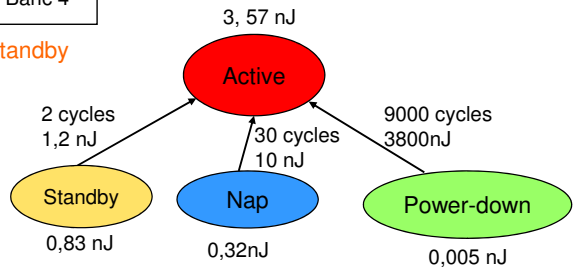


Cas idéal

Exemple de mémoire Multi-Bancs

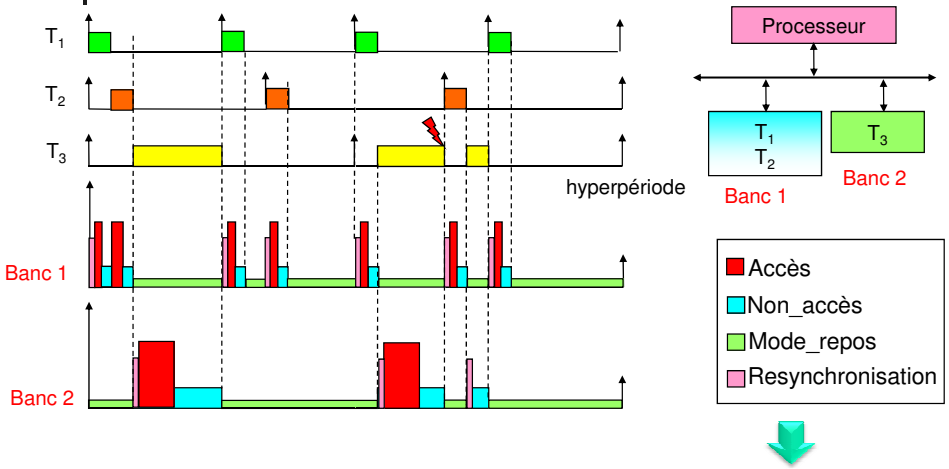


Rambus DRAM



- Mobile RAM d'Infineon, Mobile SDRAM de Micron,…

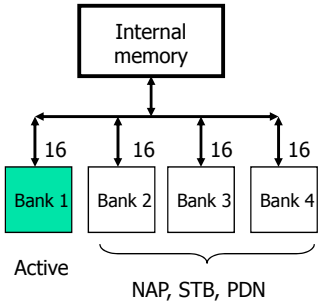
Modèle mémoire



Composantes de l'énergie

Paramètres mémoire + applicatifs

Paramètres mémoire
Access Time (Taccess)
P_{access} (par banc)
$P_{no\text{-}access}$ (par banc)
Nombre de banc
Taille des bancs
Pour chaque mode repos (par banc) : - T_{entry}, -T_{exit}, - P_{LP_mode} - P_{mode_switch}



Tâche Ti
WCET (Ci)
Periode (Ti)
Taille mémoire
Nombre d'accès à la mémoire externe

Composantes de l'énergie

$$E_{memory} = E_{access} + E_{noaccess} + E_{LP\_mode} + E_{mode\_switch} + E_{preemption}$$

$(\varphi(T_i)) = (b_j)$: fonction d' affectation de la tâche Ti au banc Bj

$$E_{access} = \sum_{b_j / j=1}^{k} \left(\sum_{T_i / \varphi(T_i) = b_j}^{N} N_{memory\_access\_T_i} \times E_{access} \right)$$

$$E_{noaccess} = \sum_{b_j / j=1}^{k} \left(\sum_{T_i / \varphi(T_i) = b_j}^{N} (C_i - T_{memory\_access\_T_i}) \times P_{noaccess} \right)$$

DVFS => Ci augmente

$$E_{LP\_Mode} = \sum_{b_j / j=1}^{k} T_{LP\_Mode\_b_j} \times P_{Mode\_repos}$$

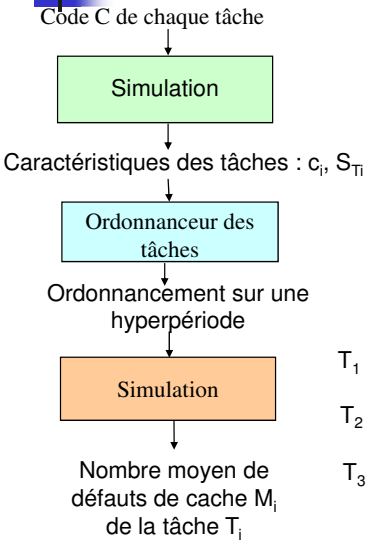
$E_{preemption}$ => estimation du coût moyen

Caractérisation de la mémoire

E_{access}, $P_{noaccess}$, P_{LPmode}

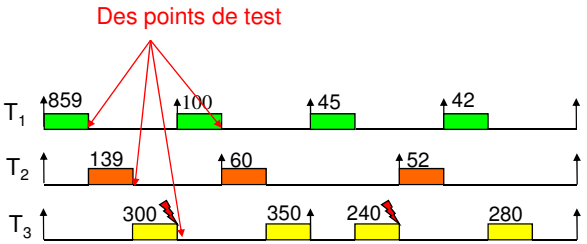
- Données constructeur
- Modèle théorique => $E_{access} = E_{bus} + E_{DRAM} + E_{switch}$
  - ⇒ nombre de bancs,
  - ⇒ taille des bancs …
  - ⇒ [CACTI]
  - ⇒ rmq : Difficile d'avoir la caractérisation des modes repos

Caractérisation des tâches : Ci + Taccess



Dans un contexte multi-tâches :

- Défauts de cache de démarrage (*cold miss*)
- Défauts de cache intrinsèques,
- Défauts de cache extrinsèques.



PLAN

1. Problématique
2. Systèmes monoprocesseur
   1. Modèle énergétique du processeur
   2. Modèle énergétique de la mémoire
   3. Stratégies basse consommation
3. Systèmes multiprocesseur
   1. Exemple de modèle de consommation
   2. Ordonnancement basse consommation
   3. Optimisation de la consommation mémoire

Les principales techniques de VFS

1- Recherche d'une Vitesse minimale statique globale ou locale

2 - Approche par chemin

V et F déterminées à partir d'un chemin d'exécution de référence (WCEP). Si le système dévie de ce chemin de référence, le couple (V, F) est ajusté.

Chemin plus long => V, F augmentent

Chemin plus court => V et F diminuent

L'analyse des différents chemins de contrôle d'une tâche est effectuée par profiling par exemple.

3 - Approche stochastique

- Approche classique : Fmax puis F réduite
- Approche stochastique : Fréduite => Faugmentée si besoin, objectif éviter d'aller à Fmax.

Approche basée sur des modèles stochastiques des temps d'exécution des tâches pour déterminer les fréquences de fonctionnement du processeur.

Les principales techniques de VFS

4 - Extension du temps d'exécution jusqu'au NTA (Arrival time of the Next Task)

Généralement AET<WCET => temps d'inactivité.

Temps d'inactivité si le NTA de la tâche suivante se situe après le WCET de la tâche active => ajustement de la fréquence de telle façon que la tâche active termine son exécution juste avant le prochain NTA.

5 - Distribution de slack Time basée sur la priorité

AET< WCET, les tâches de priorités moins élevées peuvent utiliser l'intervalle de temps (WCET – AET) des tâches plus prioritaires afin de réduire leur vitesse d'exécution (CC-EDF, Look-Ahead EDF, …).

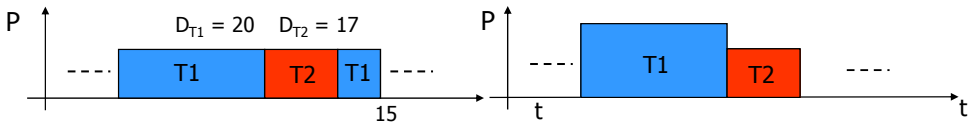
6 - Mise à jour du taux d'utilisation du processeur

Mise à jour du taux d'utilisation du processeur en certain point de l'ordonnancement (généralement au début et la fin d'exécution d'une tâche) avec le temps d'exécution AET des taches qui ont terminé leurs exécutions. Ainsi, une fois le taux d'utilisation mis a jour, la vitesse du processeur est ajustée. Simplicité d' implémentation.

DVFS, Préemption, Remèdes [Ki04]

- Accélérer une tâche pour finir avant l'arrivée d'une tâche plus prioritaire [Ki04],

  => Diminue le nombre de préemption mais certaines tâches peuvent avoir une fréquence plus élevée que celle f_{DVS}.

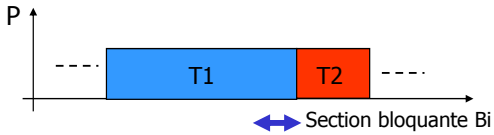


⇒ Changement de couple tension/fréquence en augmentation

- La tâche la plus prioritaire est reportée afin que la tâche de priorité inférieure termine son exécution sans préemption (PTS).

  => section bloquante

$$\forall\, i = 1,...n, \quad \frac{Bj}{Tj} + \frac{1}{s} * \sum_{\substack{i=1 \\ i \neq j}}^{N} \frac{Ci}{Ti} \leq 1$$



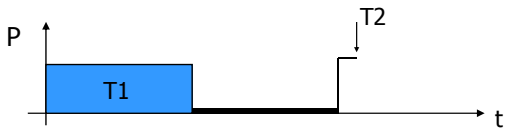
Gestion des Modes repos

- Méthode probabiliste [Be02] : prédiction du temps de mise au repos => valeur moyenne de T entre l'exécution des tâches

  => choix du mode repos,

  => réveil du processeur si forte probabilité d'arrivée d'une tâche

  => temps réel souple.



t

DVFS + Modes repos

- LC (Leakage Control)-EDF [LR03] : Modification de l'ordonnancement EDF.

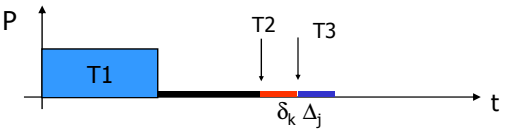
Si le processeur est en mode repos, lorsqu'une tâche t_k arrive, on allonge le mode repos d'un intervalle Δ_k tel que :

$$\sum_{i=1et\, i\neq k}^{N} \frac{Ci}{Ti} + \frac{C_k + \Delta_k}{T_k} < 1$$

Si une autre tâche Tj arrive et le processeur est encore en mode repos, deux cas :

1. Deadline t_j < deadline t_k → EDF
2. Deadline t_j > deadline t_k → retarder t_j par Δj

$$\frac{(C_k + \delta_k)}{T_k} + \frac{C_j + \Delta_j}{T_j} + \sum_{\substack{i=1 \\ i\neq k,j}}^{n} \frac{C_i}{T_i} = 1 \quad \ldots$$



- DVS + LC-EDF [LR03] : calcul d'un facteur de ralentissement statique, puis calcul des intervalles maximum de mise en repos selon le LC-EDF.

Optimisation de la consommation processeur + Mémoire [CS04] pour une application

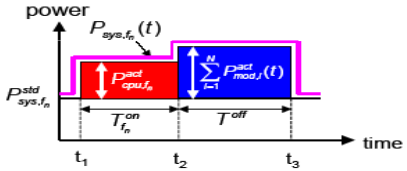
- Ton : cycles processeur utilisés pour exécuter des instructions dans le processeur

- Toff : nombre de cycles d'accès à la mémoire externe.

- Cette décomposition est réalisée en ligne basée sur des statistiques données par le PMU (Performance Monitoring Unit)

$$T = T_{on} + T_{off} = \frac{N \times CPI_{on}^{avg}}{f^{cpu}} + \frac{M \times CPI_{off}^{avg}}{f^{off}}$$

$$f_{targ\,et}^{cpu} = \frac{f_{max}^{cpu}}{1 + PF_{loss} \cdot \left[1 + \left(\frac{T^{off}}{T^{on}}\right) \cdot \left(\frac{f_{max}^{CPU}}{f^{CPU}}\right)\right]}$$



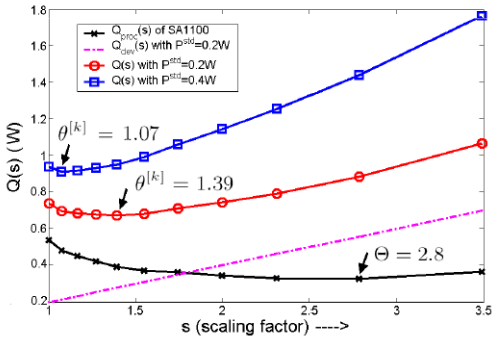
- On s'accorde une dégradation des performances : PF_{loss}.

- pour une dégradation de performance 20 % : gain pouvant aller de 20% (pour des applications à faible accès mémoire), à 80% (pour des applications à fort accès mémoire).

Ordonnancement faible consommation processeur + Mémoire [Zh05]

Le meilleur facteur de ralentissement (DVFS) doit prendre en compte le temps d'activation des ressources autres que le processeur, par exemple, les mémoires externes
(SDRAM + mémoire flash).

Par tâche, calcul d'un facteur de ralentissement idéal :

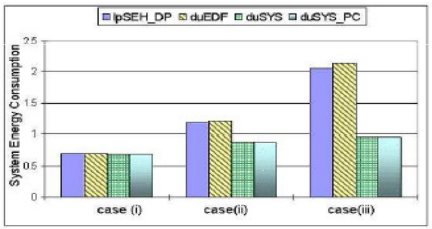


Processeur SA1100

Ordonnancement faible consommation processeur + Mémoire [Zh05]

Algorithme d'ordonnancement :

- Hypothèse : tâches périodiques, Deadline = Période, ordonnancement EDF
- En ligne, calcul d'un facteur de ralentissement optimal en fonction de la charge de travail.
- Expérimentation pour différentes puissance de repos des mémoires :



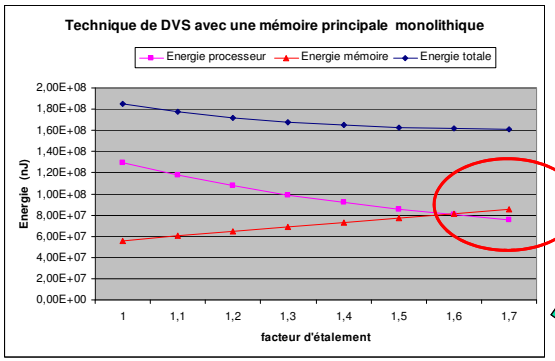
=> n'exploite pas les modes repos ni du processeur ni de la mémoire.

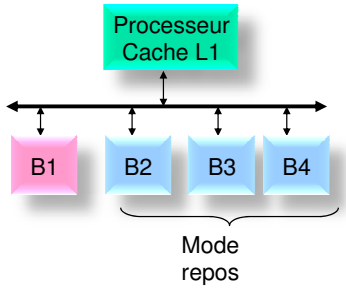
Cas 1 : Psdramstby = 20 mW, Pflashstby = 40 mW

Cas 2 : Psdramstby = 200 mW, Pflashstby = 400 mW (cas typique)

Cas 3 : Psdramstby = 2 W, Pflashstby = 4 W

VFS et consommation mémoire

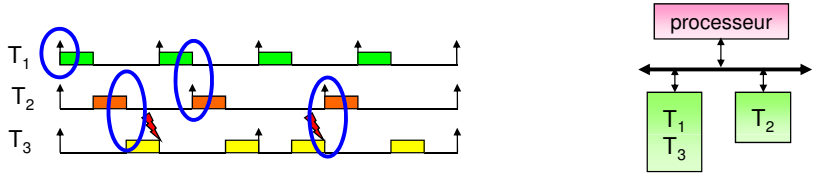




Allocation des tâches aux bancs mémoires afin de réduire la consommation de la mémoire externe

Paramètres : successivité et préemption

$E_{memory} = E_{access} + E_{noaccess} + E_{LP\_mode} + E_{mode\_switch} + E_{preemption}$



$N_{resynchronisation\_b1} = N_{exeT1} + N_{exeT3} - \sigma_{13} = 4$; $N_{resynchronisation\_b2} = N_{exeT2} = 3$.

$$N_{resynchronisation\_b_j} = \sum_{T_i / \varphi(T_i)=b_j} N_{exeTi} - \sum_{T_i,T_j / (\varphi(T_i),\varphi(T_j))=(b_j,b_j)} \sigma_{ij}$$

Ordonnancement statique => σ_{ij}

Prise en compte de la variabilité du temps d'exécution donc de l'ordonnancement => $P\sigma_{ij}$ probabilité de successivité

Optimisation de la consommation mémoire

Algorithme exhaustive

- Configuration optimal
- Espace d'exploration augmente exponentiellement avec le nombre de tâches

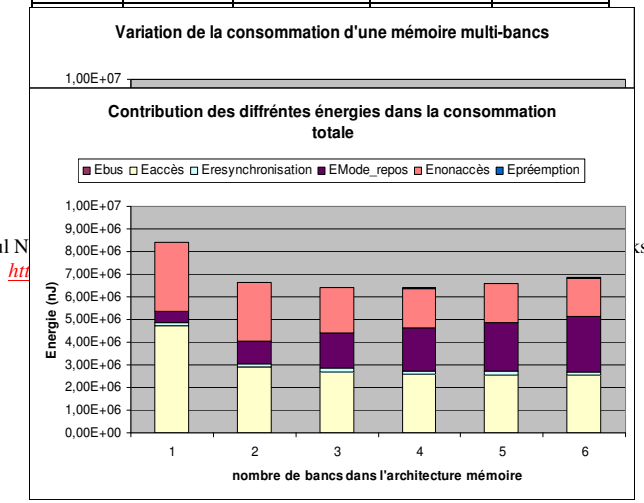
  $B_4 = 15$; $B_8 = 4140$,
  $B_{10} = 115975\ldots$ $B_{20} = 51724158235372$.
- Applicable uniquement hors ligne pendant la phase de conception.

Algorithme avec Heuristique

- Configuration proche de l'optimal
- Complexité de $O(N^3)$,
- Contexte dynamique.

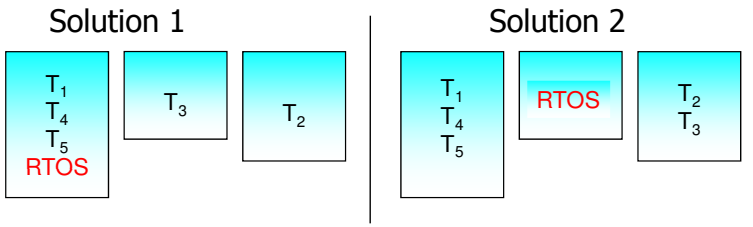
Consommation et nombre de bancs

Tâche	P_i (cycles)	c_i (cycles)	S_{Ti} (kbytes)	M_i

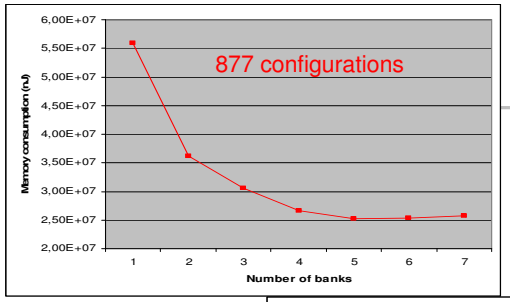


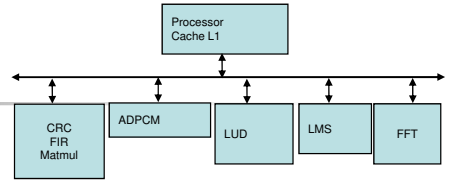
Seoul N… …ks.
htt…

Place du RTOS ?



Gain de 10 % à 45 % en fonction des applications (taux d'accès mémoire)

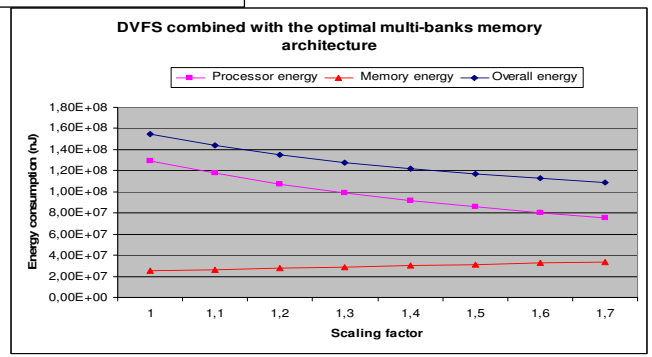




Solution Optimale

- energy savings is about 54%.

DVFS + configuration mémoire optimale



de15 à 35% de gain en énergie totale, dépend du facteur d'étalement

PLAN

1. Problématique
2. Systèmes monoprocesseur
   1. Modèle énergétique du processeur
   2. Modèle énergétique de la mémoire
   3. Stratégies basse consommation
3. Systèmes multiprocesseur
   1. Exemple de modèle de consommation
   2. Ordonnancement basse consommation
   3. Optimisation de la consommation mémoire

Architecture SMP : mesure de Puissance

P en MW	0,95V 150 MHz	1,08V 200Mhz	1,2v 250 Mhz	1,35V 300 Mhz
No activity	93	156	241	361
1 core	160	270	415	633
2 cores	225	380	587	885
3 cores	276	462	701	1034
4 cores	335	561	841	1256

$$P = K_{arm1}.V^2.F + K_{arm2}.V^2.F .Nproc$$

Consommation du Processeur P0

Processeurs activés en plus de P0

Erreur < 10%

Architecture SMP , décodeur H264

➢ Expérimentation avec un décodeur H264:

- 4 CPUs 150MHz/0.95V:
  - 13.4 fps
  - 335 mW
  - 7.51mJ

- 2 CPUs 300MHz/1.32V:
  - 13.5 fps
  - 880 mW
  - 18.80mJ

	Configuration			
1.32V/300MHz	8 threads	4 threads	2 threads	1 thread
1 core	27270934	26618776	25898597	25689278
2 core	19214328	18753565	18761297	24922293
3 core	17106270	18352958	18087990	24859718
4 core	15396686	14447570	18401966	25218340
1.20V/250MHz	8 threads	4 threads	2 threads	1 thread
1 core	20556629	20651264	20106985	19947207
2 core	14998485	14476599	14364500	19108370
3 core	13340077	14084984	14240079	19434282
4 core	12309593	11776155	14229021	19250358
1.08V/200MHz	8 threads	4 threads	2 threads	1 thread
1 core	16427499	16385310	15964827	15830034
2 core	11879384	11541959	11487365	15440986
3 core	10224445	11125531	11317221	15324991
4 core	9782451	9222689	11527336	15387828
0.95V/150MHz	8 threads	4 threads	2 threads	1 thread
1 core	13047007	12728265	12474501	12326764
2 core	9169221	9108118	9043768	11933668
3 core	8221745	8448172	9047181	12042604
4 core	7569195	7505936	8878216	12042511

TABLE IV

ENERGY CONSUMPTION ($10^{-9} J$) FOR SEVERAL CONFIGURATIONS AND WORKLOAD (FOREMAN)

Autre Modèle [Ma07]

$$Ptotale = Pdyn\ \max * \sum_{t=1}^{TN} \frac{WCET\ (t)}{D} * F_1(f_{jit}) +$$
$$PN * Pstat\ \max * F_2(f_{jit}) +$$
$$P\text{standbye} * F_3(f_{jit}) * (PN\ \max - PN)$$

PN : processor number
TN : task number
Fjit : Just In Time Frequency

Décomposition :
- P_{cpus}
- $P_{communication}$
- $P_{mémoire}$

PLAN

Ordonnancement multiprocesseur

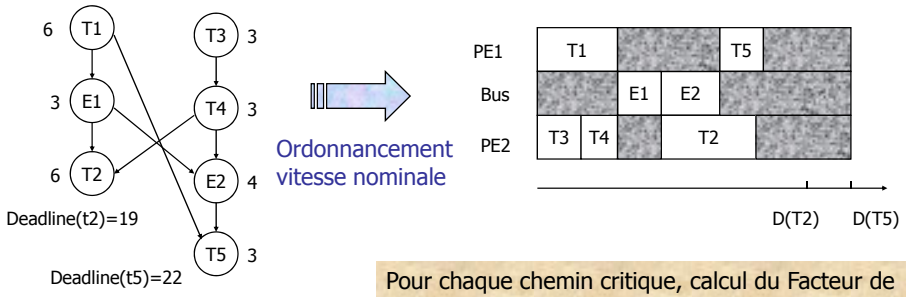
- Ordonnancement global
- Allocation + ordonnancement local
- Systèmes homogènes, hétérogènes
- Vitesse unique par classe de processeur ou par processeur
- Tâches indépendantes :
  - Borne min de processeurs donnée par U
  - Borne max donnée par la plateforme

=> Condition d'ordonnancabilité : U<m+1/2 !

Multiprocesseurs et basse consommation

- **Approche statique :**
  - Ajustement des vitesses,
  - Réduction du pic de puissance,
  - Ordonnancement statique,
  - Allocation + ordonnancement statique
- Approche dynamique :
  - Ordonnancement dynamique
  - Allocation, ordonnancement dynamique locale
  - Allocation, ordonnancement dynamique (migration de tâches)

Ajustement des vitesses processeur [Luo02]

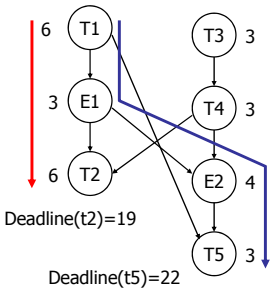


Ti : tâche i
Ei : Communication entre unités

Pour chaque chemin critique, calcul du Facteur de Ralentissement :

FRchemin critique = ((Tfin - Tdébut – Σ_i WCETi)/ Σ_i sur processeur WCETi) + 1

Ajustement des vitesses processeur [Luo02]



Ti : tâche i
Ei : Communication entre unités

Deux chemins critiques :

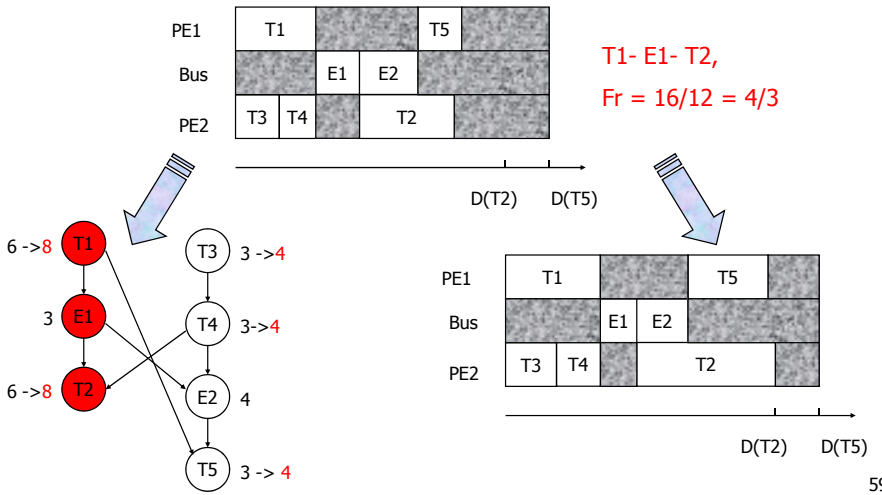
1 - T1- E1- T2, Fr = 16/12

2 - T1- E1- E2 -T5, Fr= 5/3 (---)

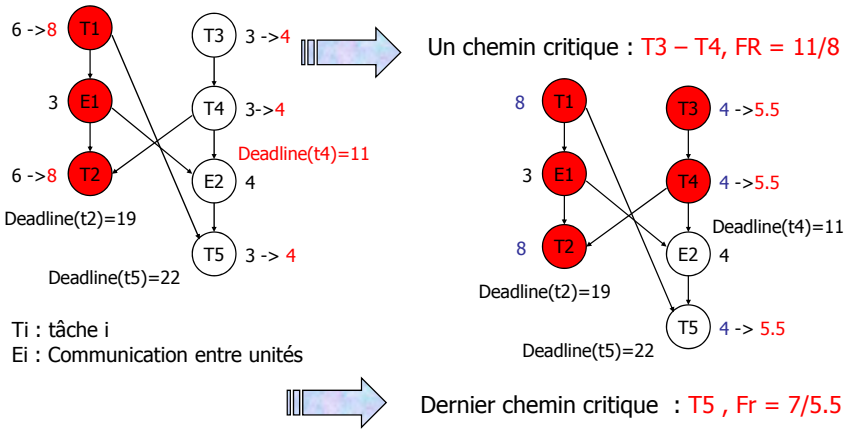
On sélectionne FRmin.

Toutes les tâches se voit affectées de nouvelles dates d'exécution. Celles appartenant au chemin sont ôtées du graphe. On itère cet algorithme.

Ajustement des vitesses processeur [Luo02]



Ajustement des vitesses processeurs [Luo02]

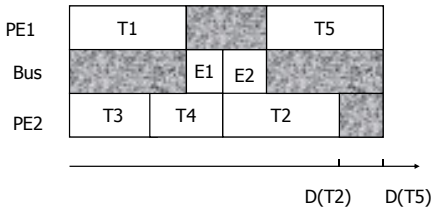


Ti : tâche i
Ei : Communication entre unités

Dernier chemin critique : T5 , Fr = 7/5.5

Ajustement des vitesses processeur [Luo02]

Au final :



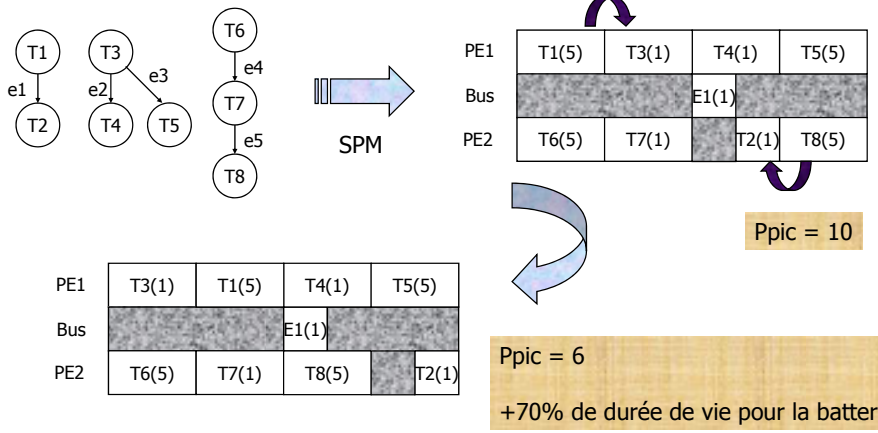
	T1	T2	T3	T4	T5
FR	8/6	8/6	5.5/3	5.5/3	7/3

Autres stratégies possibles :

Par charge processeur, …..

Résultats : -30% sur E

Réduction du pic de puissance Ppicmin [Luo01]



Autre approche : Denis Dupont, IBISC

Allocation, ordonnancement + VFS [Chen05] (1)

- Processeurs pouvant travailler à des vitesses différents, Tâches indépendantes
- $P = C.Vdd^2.S$ avec $S=K(Vdd-Vt)^2/Vdd$ => $P(s) = h.s^{\alpha}$ (avec $2 < \alpha < 3$)
- Changement de vitesse : pas de pénalité
- Pas de migration, tous les jobs d'une tâche sont sur le même processeur avec la même vitesse
- Deux phases (M processeurs) :
  - Phase de relaxation (programmation convexe) :
    - Minimize Σ Ei(ti), Subject to Σ ti/pi = M, avec 0 < ti ≤ pi (ti=ci*s)
    - the Karush-Kuhn-Tucker optimality condition in O(|T| log |T|).

    => À l'issu de cette phase, on déduit le temps d'exécution ti (s=ci/ti) de chaque tâche Ti qui minimise l'énergie
  - Phase de rounding :
    - algorithme LEUF (Largest-Estimated Utilization First) : les tâches sont affectées au processeur ayant le plus faible taux d'utilisation (Um), on commence par les tâches ayant le plus grand taux d'utilisation (U_{Ti} = ti/pi).
    - ordonnancement EDF par processeur à la vitesse correspondant à ti' = ti x 1/Um
- RMQ : pas de gestion des modes repos, communication non pris en compte

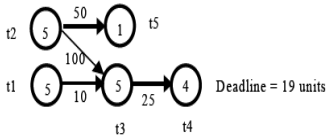
Allocation, ordonnancement + VFS [Ian06] (2)

- Approche prenant en compte la consommation statique
- Tâches dépendantes
- Choix optimal du nombre de processeurs + voltage scaling
  - Borne min (Mlwb) et max (Mupb) du nombre de processeur
  - Pour M =(Mlwb + Mupb)/2 processeurs, ordonnancement de type list scheduling qui utilise les priorités de l'EDF => si l'ordonnancement est possible, exploration de Mlwb à M, sinon de M à Mupb.
  - Recherche du nombre de processeur Mmin qui satisfait les échéances => Mmin
  - Ensuite pour Mmin, Mmin+1, …, recherche du nombre de processeur et d'une fréquence de travaille qui minimise l'énergie (ordonnancement List scheduling + ralentissement global pour tous les processeurs). On arrête la recherche dès que l'énergie ne décroit plus.
- Rmq : n'utilise pas les modes repos sur les processeurs actifs (temps de réveil trop long).

Allocation et ordonnancement + VFS [Var03] + prise en compte des communications (1)

Approche statique

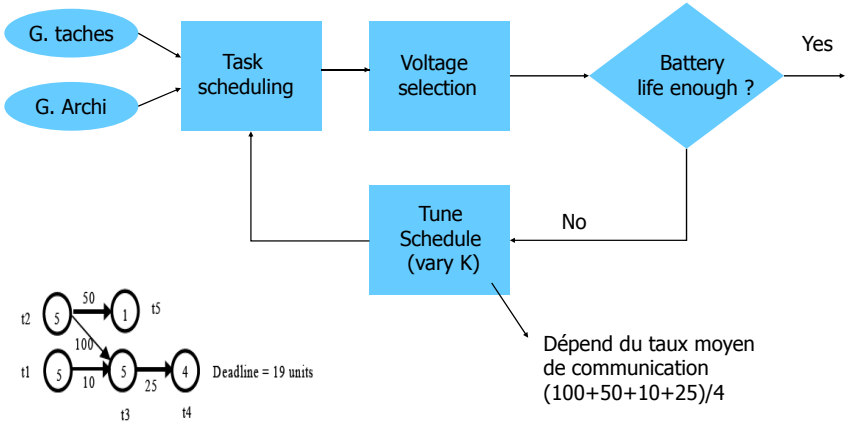
- Taches dépendantes,
- Plateforme hétérogène classe de processeur (RISC, DSP, ASIC …)
- Principe : Réduction du nombre de communication, DVS
- Caractéristiques d'entrée :
  - Graphe de Tâches =>
    - Nœud = tâches : WCET et E(V) par classe de processeur
    - Arc : nombre de bits à transférer,
    - Deadline,
  - Graphe de l'architecture :
    - Nœud : processeur (classe d'appartenance),
    - Arc : communication entre les processeurs, vitesse, énergie d'une communication



Allocation et ordonnancement + VFS [Var03] + prise en compte des communications (2)

Approche statique

List scheduling + heuristique pour réduire le nombre de communications

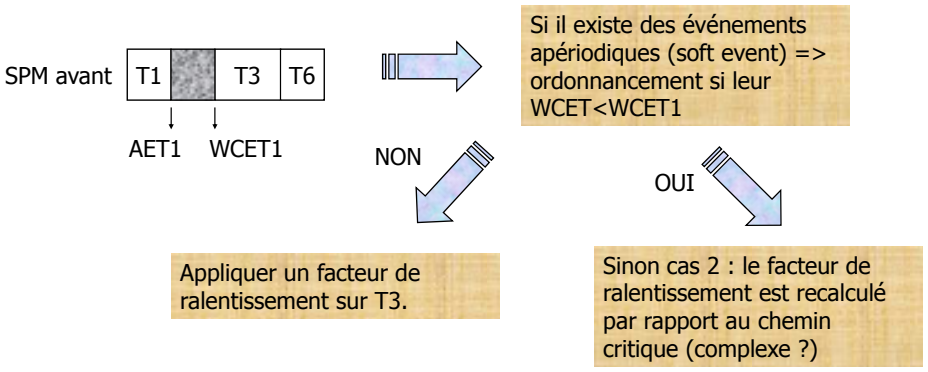


Multiprocesseurs et basse consommation

- Approche statique :
  - Ajustement des vitesses,
  - Réduction du pic de puissance,
  - Ordonnancement statique,
  - Allocation, ordonnancement statique
- **Approche dynamique :**
  - Ordonnancement dynamique
  - Allocation, ordonnancement dynamique locale
  - Allocation, ordonnancement dynamique (migration de tâches)

Approche dynamique

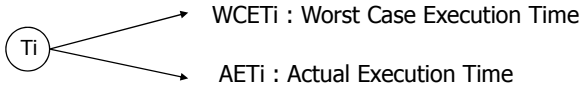
Ordonnancement dynamique + DVFS [Luo02]



Allocation et ordonnancement dynamique + DVFS [Zh03]

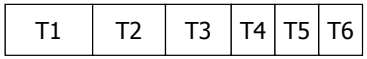
Cas 1 : ensemble de tâches indépendantes et un deadline D.

Hypothèse : on change de couples V,F sans pénalités autant de fois que nécessaire.



Exemple : T1(5,2), T2(4,4), T3(3,3), T4(2,2), T5(2,2), T6(2,2).

File : (globale)

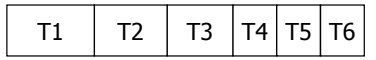


(WCET décroissant)

La file des tâches exécutables est donnée par l'algorithme du list scheduling.

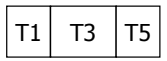
Allocation et ordonnancement dynamique + DVFS [Zh03]

File :

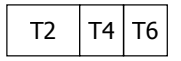


(WCET décroissant)

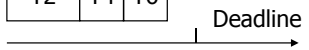
P1



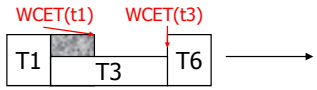
P2 :



avec AET sans DPM

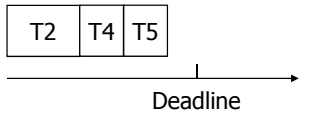


P1



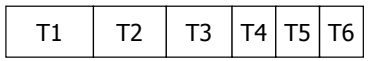
La tâche T3 est ralentie d'un facteur 6/3.
Pb : on dépasse le deadline

P2 :

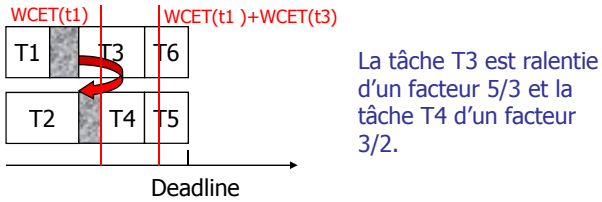


Allocation et ordonnancement dynamique [Zh03]

File :



(WCET décroissant)



La tâche T3 est ralentie d'un facteur 5/3 et la tâche T4 d'un facteur 3/2.

Généralisation à n processeurs (homogène) et pour une ensemble de tâches avec dépendance.

Allocation et ordonnancement dynamique [Zh03] Quelques Résultats

Etude de E en fonction de α (AETi = α.WCETi) :

2 processeurs et Pour $\alpha < 0.5$

des tâches non dépendantes (100) : gain de 60% sur E,

des tâches dépendantes (20) : gain de 40% sur E.

Etude de E en fonction du nombre de processeurs (α=0.5):

Energie sensiblement constante jusqu' à 8 processeurs au delà augmentation très rapide.

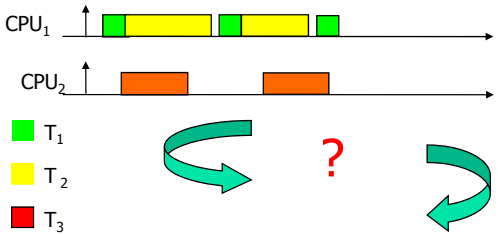
Allocation, ordonnancement, Température [Mer06]

- Problème couplé de la température et de l'énergie (P est liée à T).
- Système homogène, chaque CPU a sa propre file des tâches exécutables => état initial ?
- Basé sur la présence d'un compteur d'activité HW (souvent présent dans les processeurs actuels) => mesures de température (différence de courant)=> profile d'énergie pour les tâches
- Equilibre des tâches hot et cool sur les processeurs
- Si un processeur s'approche de sa température haute (pour éviter le throttling) => migration de la tâche hot vers un processeurs cool (soit idle, soit en train d'exécuter une tâche cool, dans ce dernier cas, échange des tâches)
- Coût de la migration : P est directement proportionnel à la composition de la file des tâches exécutables
- Intégration dans Linux

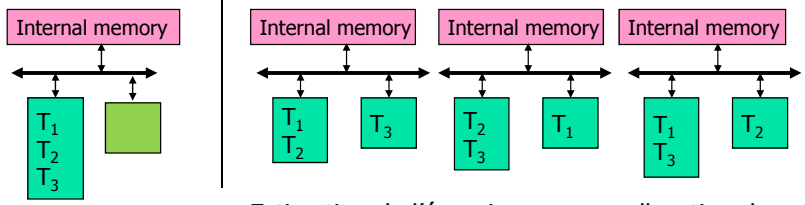
PLAN

1. Problématique
2. Systèmes monoprocesseur
   1. Modèle énergétique du processeur
   2. Modèle énergétique de la mémoire
   3. Stratégies basse consommation
3. Systèmes multiprocesseur
   1. Exemple de modèle de consommation
   2. Ordonnancement basse consommation
   3. Optimisation de la consommation mémoire (projet ANR PHERMA - ANR-06-ARFU-003)

Consommation mémoire



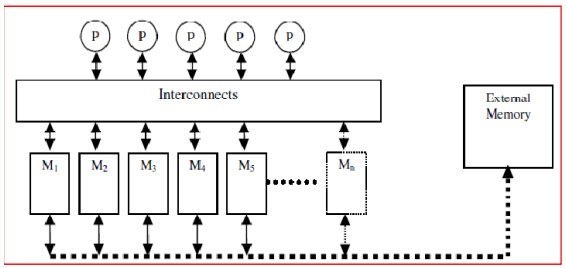
Tâche : WCET , nombre d'accès à la mémoire externe, taille mémoire
Mémoire : nombre de bancs, taille des bancs, modèle de consommation (pour ses différents modes de fonctionnements)



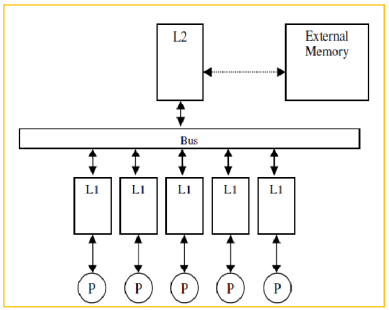
Estimation de l'énergie pour une allocation donnée => optimisation

Consommation mémoire et DVFS (projet PHERMA)

2 architectures multicoeurs :



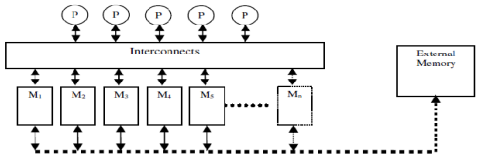
SCMP (CEA-list)
(Scalable Chip MultiProcessing)

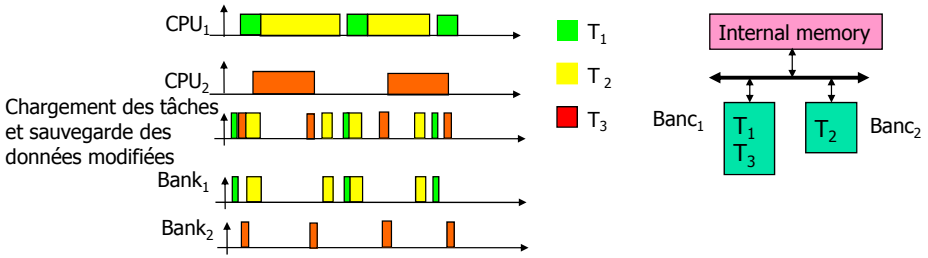


SMP (exemple : MPcore)

Mémoire externe et architectures multiprocesseur

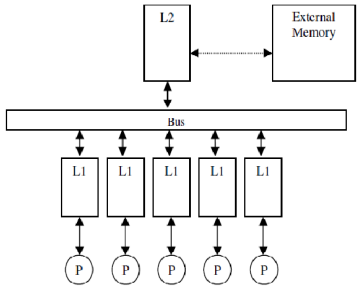
- Architecture SCMP
  - les tâches sont préchargées dans la mémoire locale avant leur exécution,
  - Après exécution de la tâche, écriture en mémoire des données modifiées en cours d'exécution.

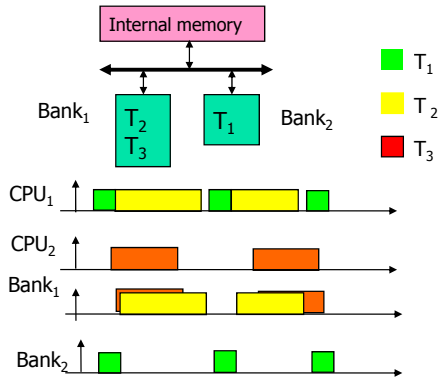




Mémoire externe et architectures multiprocesseur

- Architecture SMP :
  - Durant toute l'exécution d'une tâche, la mémoire peut être accédée en lecture ou en écriture.
  - Activation du banc contenant la tâche pendant toute la durée de son exécution





Paramètres liés aux tâches

Tâche Ti
WCET (Ci)
Period/Deadline (Di)

SMP architecture → **Number of memory access**

SCMP architecture → **Precharging time (Tpc_i)**
Writeback Time (Twb_i)

Mémoire externe : modèle énergétique

$$E_{memory} = E_{active} + E_{LP\_mode} + E_{mode\_switch} + E_{preemption}$$

$$E_{access} = \sum_{b_j / j=1}^{k} \left(\sum_{T_i / (\varphi(T_i))=(b_j)} N_{access\_Ti} * E_{access} \right)$$

$$E_{nonaccess} = \sum_{b_j / j=1}^{k} \left(\sum_{T_i / (\varphi(T_i))=(b_j)} T_{noaccess\_Ti} * P_{noaccess} \right)$$

$(\varphi(T_i)) = (b_j)$: fonction d' affectation de la tâche Ti au banc Bj

T_{access_Ti} : temps d'accès à la mémoire externe
- Architecture SCMP :
  - temps proche d'une constante lors du préchargement,
  - temps faible lors de la réécriture des données en mémoire
- Architecture SMP :
  - nombre de défaut de cache x Temps d'accès à la mémoire ou
  - WCET (à fmax) x % d'accès à la mémoire

DVFS ?

Pas d'incidence sur E_{access} et $E_{noaccess}$

Influence sur $T_{nonaccess\_Ti}$

Mémoire externe : modèle énergétique

$$E_{memory} = E_{active} + E_{LP\_mode} + E_{mode\_switch} + E_{preemption}$$

$$E_{LP\_mode} = \sum_{b_j / j=1}^{k} (\sum_{i=1}^{n} T_{bj\_LP\_mode_i} * P_{LP\_mode_i})$$

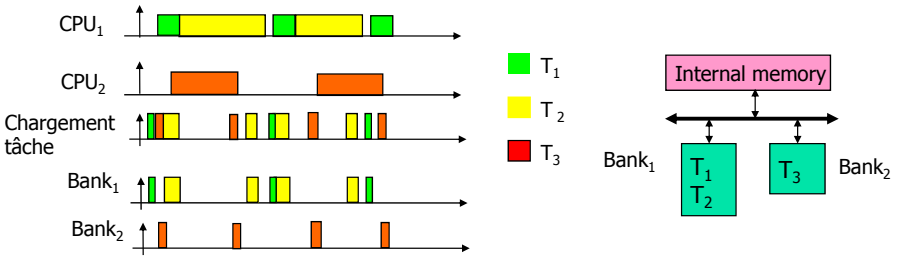
i : nombre de mode basse consommation.
Rmq: Actuellement 2 modes LP sont exploitables dans le cas de systèmes temps réels.

$$E_{mode\_switch} = \sum_{b_j / j=1}^{k} (\sum_{i=1}^{n} N_{bj\_mode_i\_switch} * T_{mode_i\_switch} * P_{mode_i\_switch})$$

$$E_{preemption} = \left(\sum_{T_i / (\varphi(T_i))=(b_j)} N_{preemption} * E_{context\_switch} \right)$$

Paramètres importants - SCMP

- **Successivité (dans Tmin_lpmode) : localité temporelle**
  - ⇒ Réduire le nombre de réveil (wakeup)
  - ⇒ Prolonger les temps d'inactivité

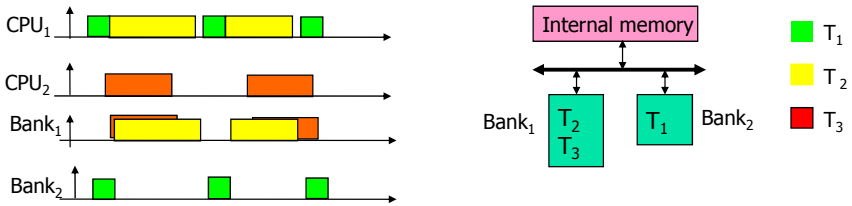


Basé sur un ordonnancement statique, comment prendre en compte la variabilité de l'ordonnancement ?

Probabilité de successivité (déduite de simulation)

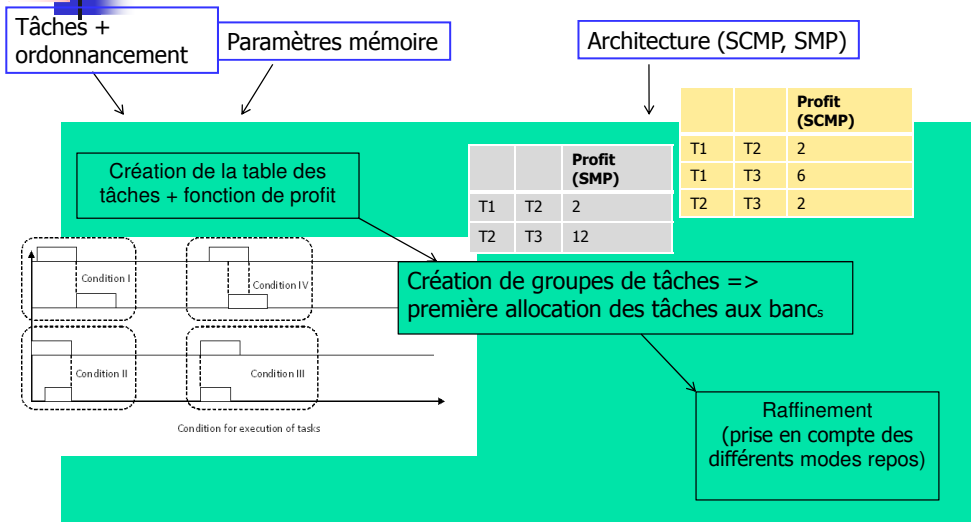
Paramètres importants - SMP

- **Successivité (dans Tmin_lpmode) : localité temporelle**
  - ⇒ Réduire le nombre de réveil (wakeup)
  - ⇒ Prolonger les temps d'inactivité
- **Exécution parallèle des tâches**
  - ⇒ Nombre de bancs actifs en même temps



Ordonnancement statique, => variabilité de l'ordonnancement ?
- Successivité : probabilité
- Exécution parallèle : temps moyen d'exécution // (peu importe le n° de CPU)

Allocation des bancs mémoires - energy aware



Conclusion

- Système monocoeur :
  - Modèle processeur E, P en fonction de V et F
    => nouvelle technologie ?
    => combien de points de fonctionnement sur V ?
  - Modèle d'un système complet ?
- Systèmes multicoeurs :
  - Politique statique par scenario d'ordonnancement basse consommation
  - Modèle de consommation mémoire
  - Modèle de communication
  - Modèle thermique

Bibliographie

- [Ac03] A. Acquaviva, T. Simunic, L. benini, « LP-ECOS : An energy efficient RTOS », HP Lanoratories, April 2003.
- [Be98] Luca Benini, Giovanni De Micheli , « Dynamic Power Management, Luca Benini, Giovanni De Micheli», Kluwer Academic Plublishers,1998.
- [Be00] L. Benini, A. Bogliolo, G. De Micheli, « A survey of design technique for system Level dynamic power Management », IEEE transactions on VLSi Systems, vol. 8, No. 3, Juin 2000, pp. 299-316.
- [Ber06] S. Bertozzi, A. Acquaviva, A. Poggiali, D. Bertozzi, "Supporting Task Migration in MPSoCs: A Feasibility Study," DATE 2006.
- [Ca04], B.H. Calhoun, F.A. Honoré, A. P. Chandrakasan, » A Leakage Reduction Methodology for distributed MTCMOS », proceeding of IEEE Joiurnal of Solid-state circuits, vol.39, no.5, may 2004.
- [Chen05] Energy efficient scheduling of Periodic Real-time tasks over homogeous Multiprocessors », Jian-Jia Chen, Tei Wei Kuo, PARC05, 2005
- [Da99] P. dave, G. Lakshminarayana, N. Jha, « COSYN: Hardware-Software Co-Synthesis of heterogeneous distributed Embedded Systems, IEEE Transactions on VLSI systems, Vol. 7, No. 1, mars 1999.
- [Do94] M. Doyle, J. Newman, J. Reimers, « A quick method for measuring the capacity versus the discharge rate for a dual Lithium-ion insertion cell undergoing cycling », Journal of pwoer sources, Vol. 52, No. 2, décembre 1994, pp. 211-216.
- [Ge01] Jun fei Geng, Battery Modeling: a Longer Lifetime, (duke University).
- [Gr02] F. Gruian, « Energy centric Scheduling for Real Time Systems », PhD thesis Lund Institute of Technology, 2002.

Bibliographie

- [Gu01] Guitton-Ouhamou Patri-cia, Belleudy Cécile, Auguin Michel, «*Power consumption Model for the DSP OAK Processor*», proceedings of 11th IFIP International Conference on Very Large Scale Integration-System-On Chip 2001, December 3-5 2001, Montpellier, France, pp73-78.
- [Gu03] P. Guitton-Ouhamou, K. Ben Chehida, C. Belleudy, M. Auguin, «Automatical architecture exploration : Energy optimisations of a codesign tool for embedded HW/SW systems», first Northeast Workshop on Circuits and Systems (NEWCAS'2003), sponsored by IEEE June 17 - 20, 2003, in Montréal, Canada.
- [FEL03] Xiaobo Fan, Carla Ellis, and Alvin Lebeck. Interaction of power-aware memory systems and processor voltage scaling. In *Proceedings of the Workshop on Power-Aware Computer Systems PACS'03*, December 2003.
- [Je04] R. Jejurikan, C. Pereira, R. Gupta, « Leakage Aware Dynamic Voltage Scaling For Real-Time Embedded Systems », Proceeding of *Design Automation Conference 2004.*
- [Ki04] W.Kim, J.Kim, S.L Min « Preemption-awware dynamic voltage scaling in hard real time systems », ISLPED 2004.
- [Ian06] Multiprocessor Scheduling to Reduce Leakage Power Pepijn de Langen, Ben Juurlink, Stamatis Vassiliadis, 17th Int. Conf. Parallel and distributed Symposium (IPDPS'06), 2006
- [Le00] X. Fan, C. Ellis, A. Lebeck, « The synergy between Power-Aware Ssystems and Processor Voltage Scaling », *Proceedings of the International Symposium on Low-Power Electronics and Design ISLPED'01*, August 2001.
- [Lu01] Jiong Luo, Niraj K. Jha "Battery-Aware Static Scheduling for Distributed Real-Time Embedded Systems", *in proceedings of 38th Design Automation Conference 2001 [p. 444] June 18-22, 2001 Las Vegas.*
- [Lu02] J. Luo and N. K. Jha, "Static and dynamic variable voltage scheduling algorithms for real-time heterogeneous distributed embedded systems," Int. Conf. on VLSI Design, Jan. 2002.

Bibliographie

- [Mer06] A. Merkel, F. Bellosa, Energy Power consumption in Multiprocessor systems, EuroSys2006.
- [Va03] Girish Varatkar, Radu Marculescu, Communication-Aware Task Scheduling and Voltage Selection for Total Systems Energy Minimization, ICCAD 2003.
- [Va04] Chuanjun Zhang, Jun Yang and Frank Vahid, " Low static power frequent value data caches" Date 2004.
- [Wu03] Dong Wu and Bashir M. Al-Hashimi, Petru Eles, "Scheduling and Mapping of Conditional Task Graphs for the Synthesis of Low Power Embedded Systems", *in proceedings of Design Automation Test in Europe 2003 Munich, Germany, March 3-7, 2003, p.90-95.*
- [ZC04] F. Zhang, S. Chanson, " Blocking-Aware Processor Voltage Scheduling for Real-Time tasks ", ACM transaction On Embedded Computong System, Vol.3, Issue 2, pp.307-335
- [Zh03] D. Zhu, R. Melhem, and B. Childers; "Scheduling with Dynamic Voltage/Speed Adjustment Using Slack Reclamation in Multi-Processor Real-Time Systems", IEEE Trans. on Parallel & Distributed Systems, vol. 14, no. 7, pp. 686 - 700, 2003.
- [Zh04] Chuanjun zhang, Frank Vahid, " Using a victim buffer in application-specific memory hiearachy", DATE 2004.

Autre :

- [Ca] CACTI : http://www.ece.ubc.ca/~stevew/cacti.html
- [Mo] Mosys low power design –don't forget about the memory, http://www.mosys.com
- [softexplorer] http://www.softexplorer.fr/