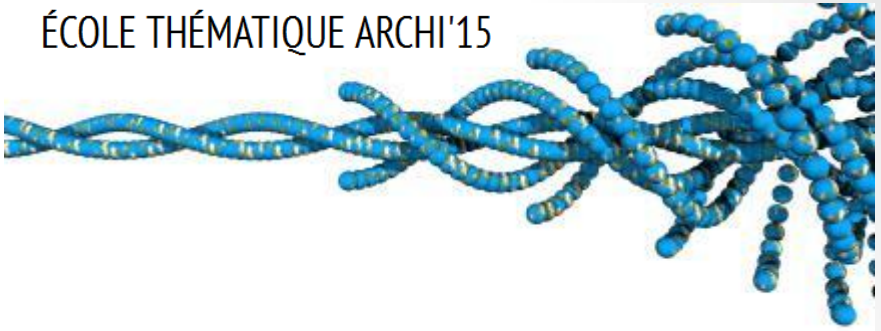
ÉCOLE THÉMATIQUE ARCHI'15



Architectures de SoC basse consommation : modélisation TLM du contrôle

Michel Auguin, Hend Affes, François Verdier

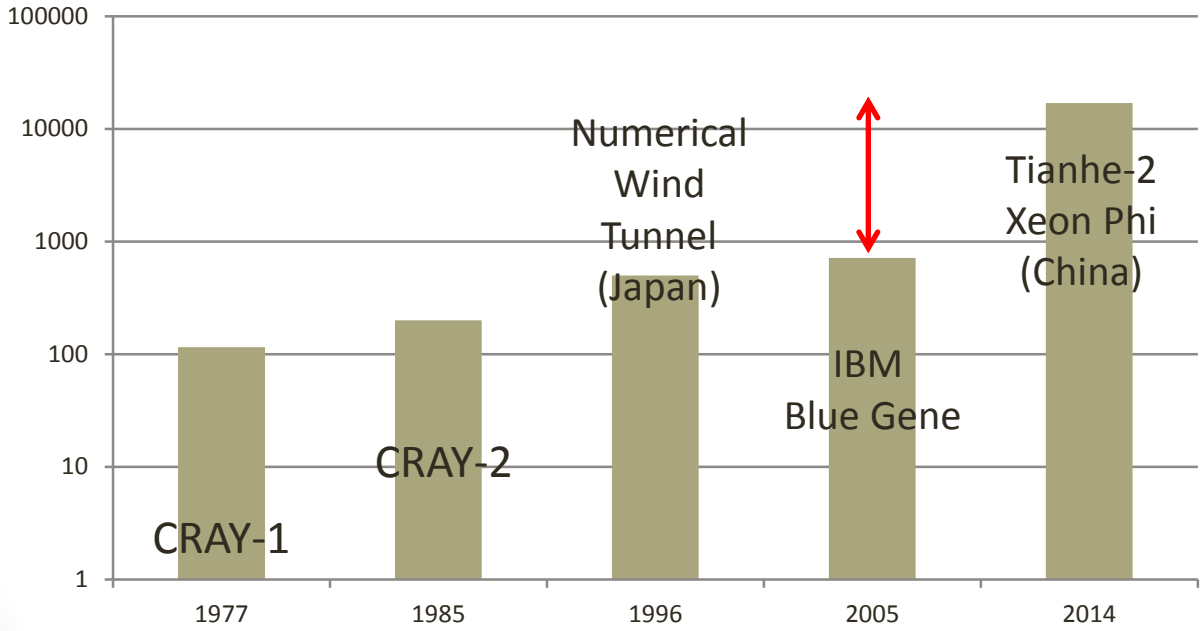
Université Nice Sophia Antipolis

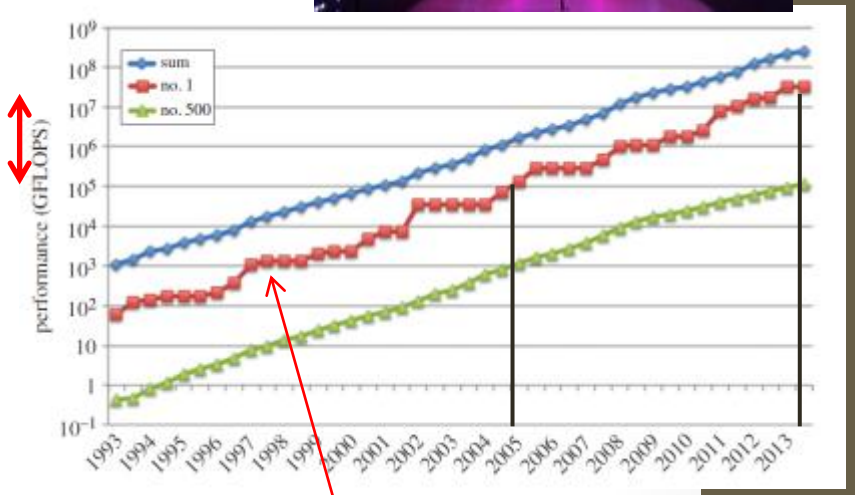
LEAT
LABORATOIRE D'ELECTRONIQUE ANTENNES ET TELECOMMUNICATIONS

cnrs
dépasser les frontières

- Commençons par le HPC : Consommation d'énergie des supercomputers les plus puissants

Puissance Consommée (kW)





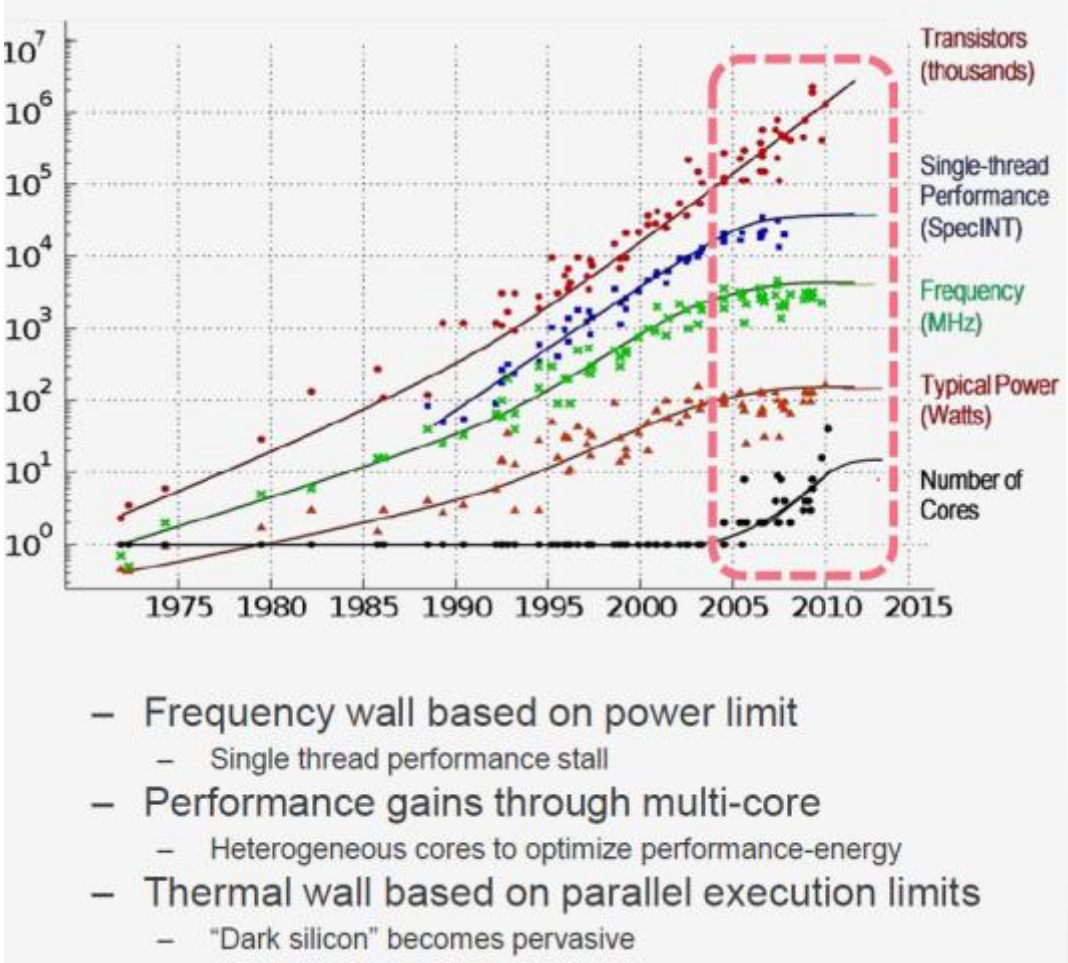
Performance max du supercomputer le plus puissant (Gflops)

Peut on encore longtemps suivre ce rythme d'augmentation de la puissance dissipée ?

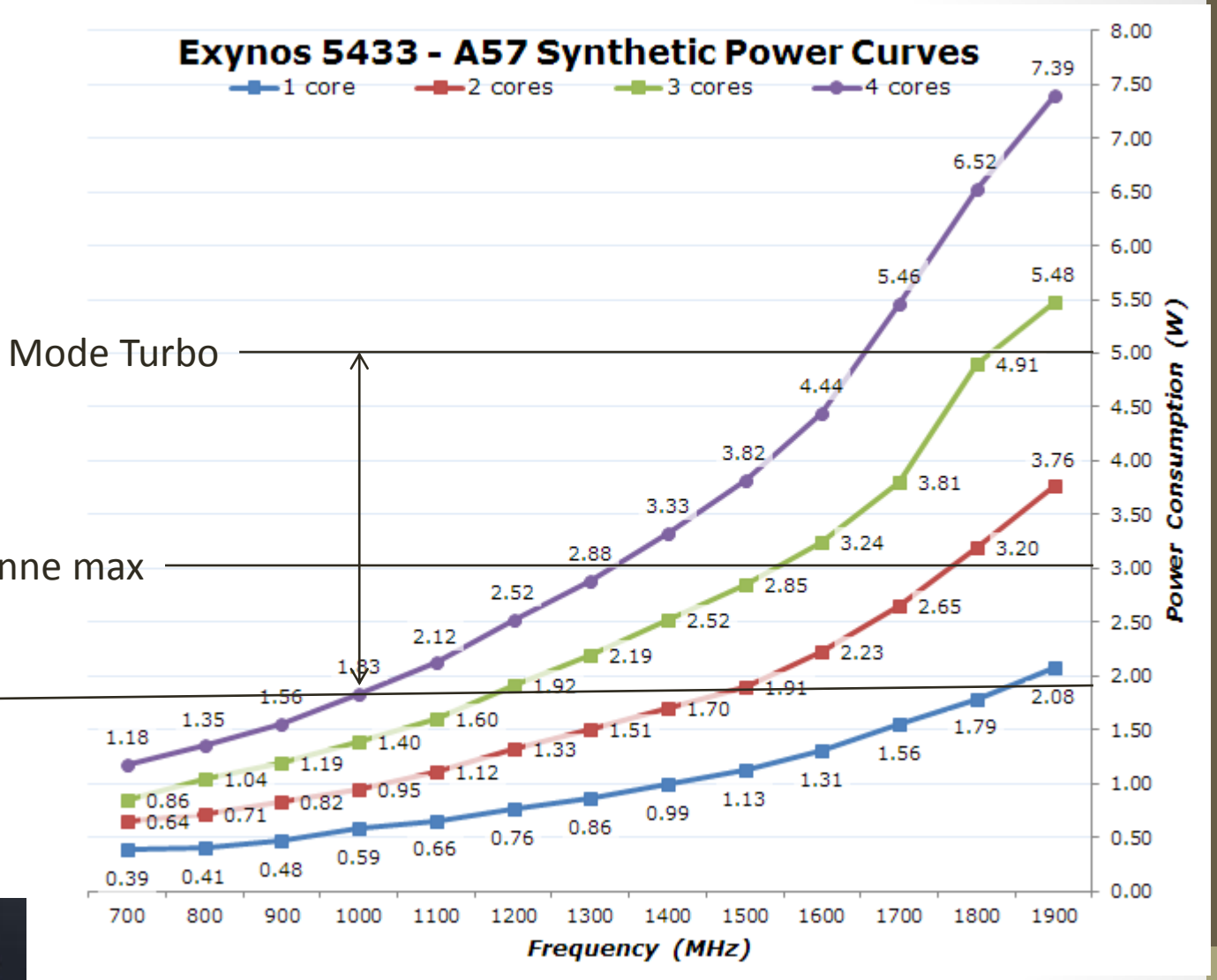
Giles MB, Reguly I., Trends in high-performance computing for engineering calculations, 2014, http://dx.doi.org/10.1098/rsta.2013.0319

- Si des contraintes existent sur l'accroissement du taux d'intégration dans un chip, d'autres contraintes limitent l'exploitation des capacités d'intégration de la technologie CMOS de type bulk.

- La dissipation de puissance et les effets thermiques associés sont ainsi les principaux freins à une augmentation des performances embarquées dans un circuit.



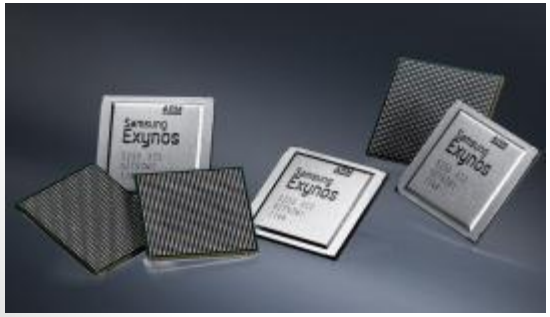
Présentation Cadence :
http://www.eda.org/edps/Papers/1-1%20Steve%20Carlson.pdf



Mode Turbo

Puissance moyenne max

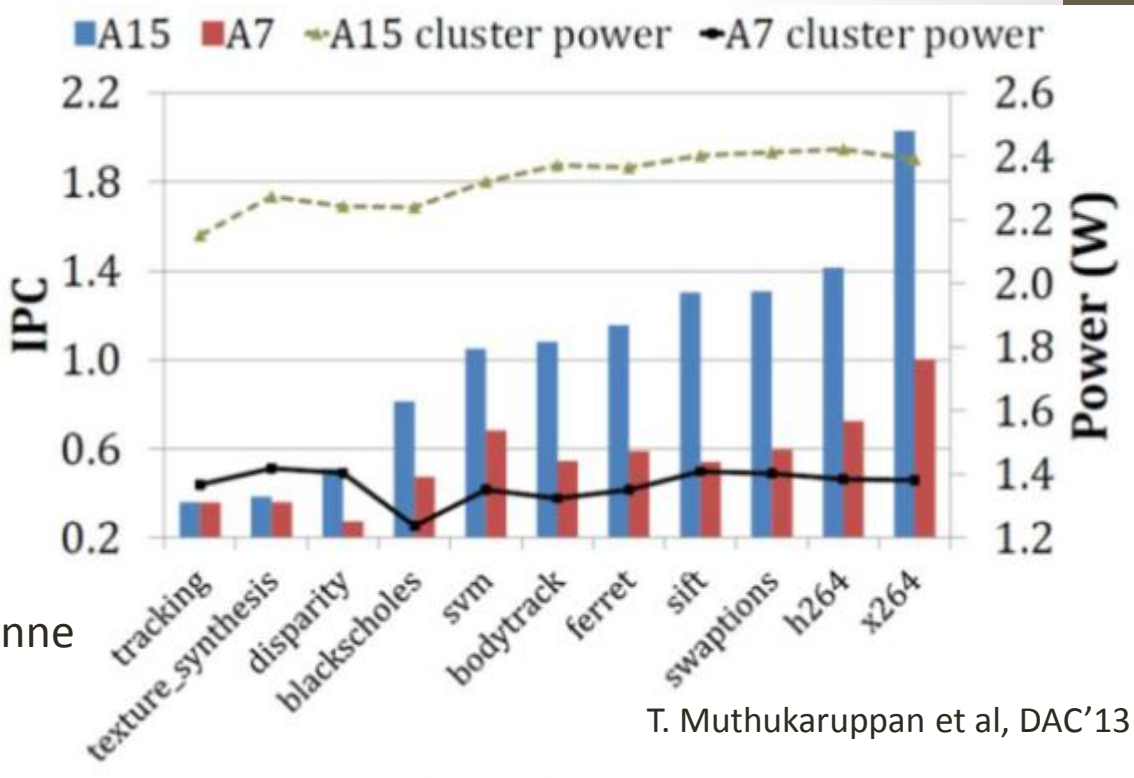
Point de fonctionnement pour attendre retour à température « normale »



ARM Big.LITTLE

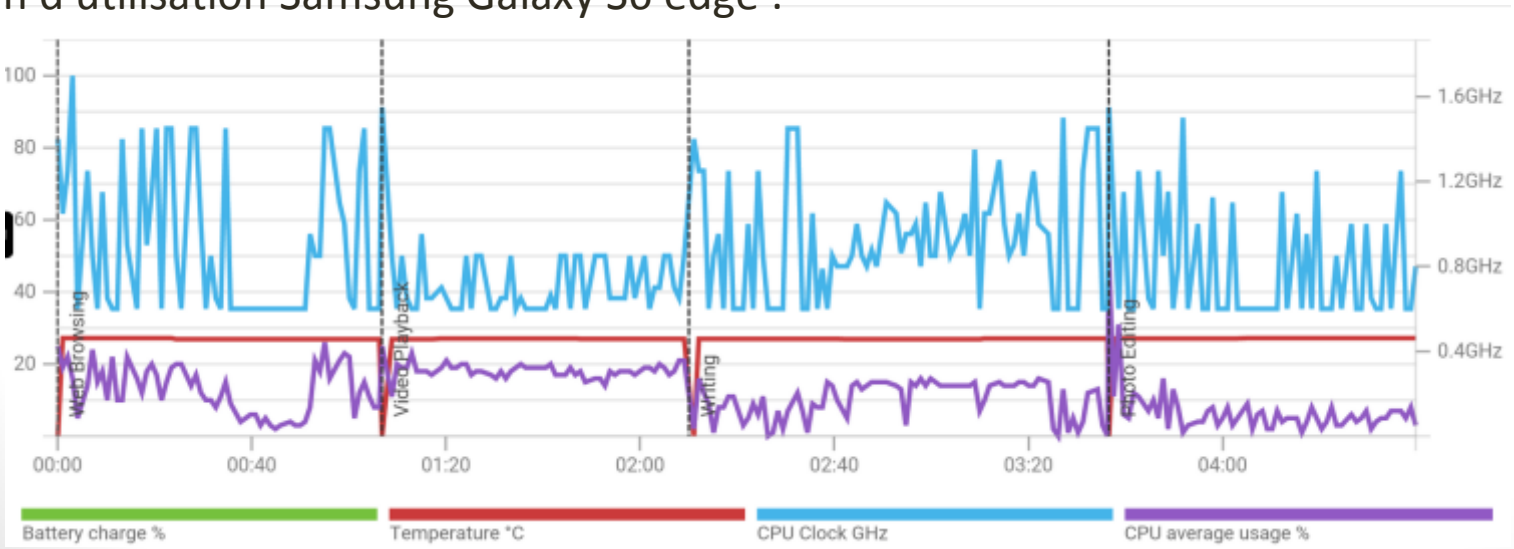
- 2 cœurs A15 et 3 cœurs A7
- A15 : out of order;
- A7: in order

1 seul cœur actif dans chaque cluster
Fréquence 1GHz pour tous les cœurs
Vdd = 1,05V
A15 vs A7 :
Accélération = 1,7 en moyenne
Puissance consommée = x1,86 en moyenne



T. Muthukaruppan et al, DAC'13

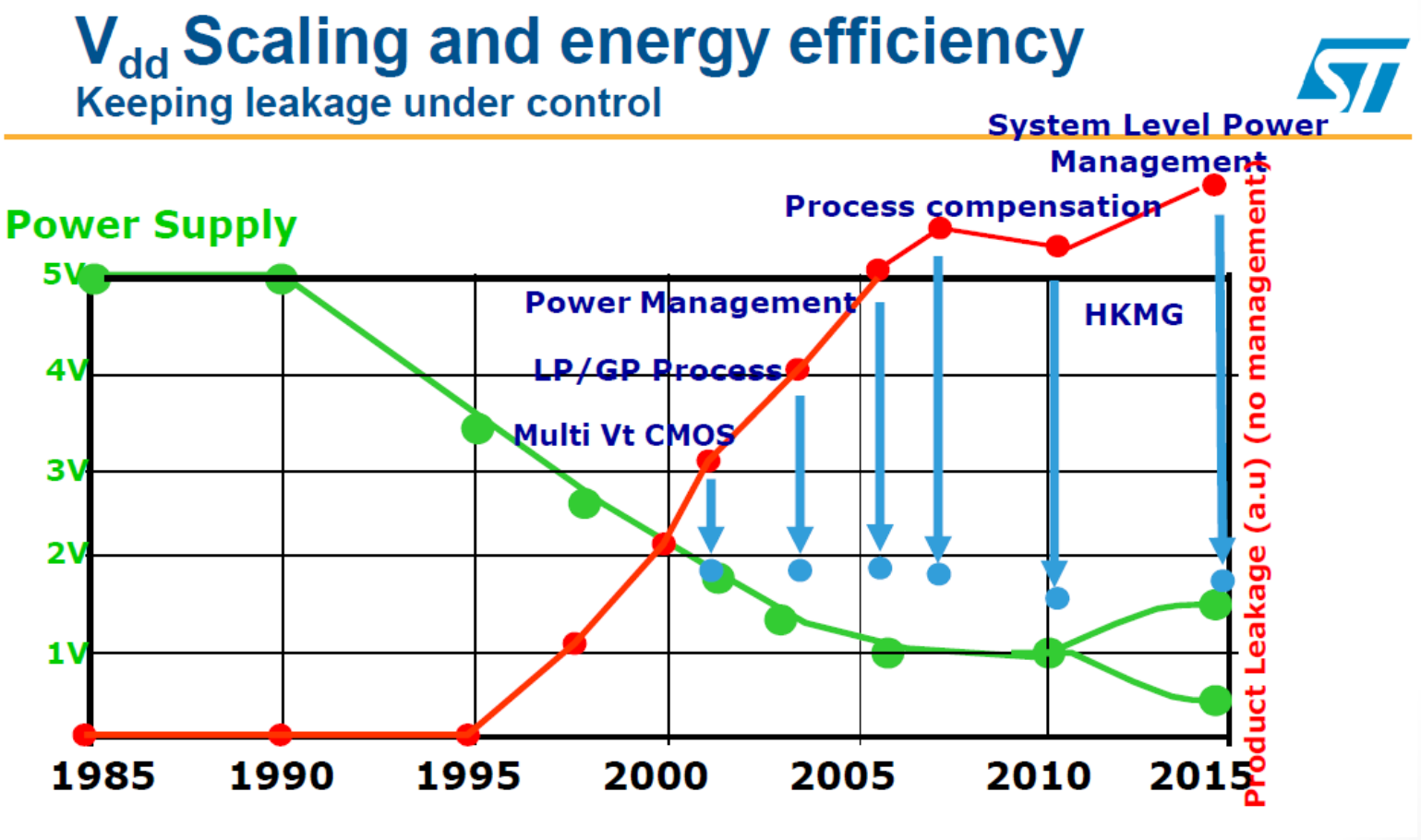
4mn d'utilisation Samsung Galaxy S6 edge :



R. Pandev, laptopmedia.com, 30/04/15

big.LITTLE avec 4x Cortex-A57 (2.0GHz) et 4x Cortex-A53 (1.6GHz)

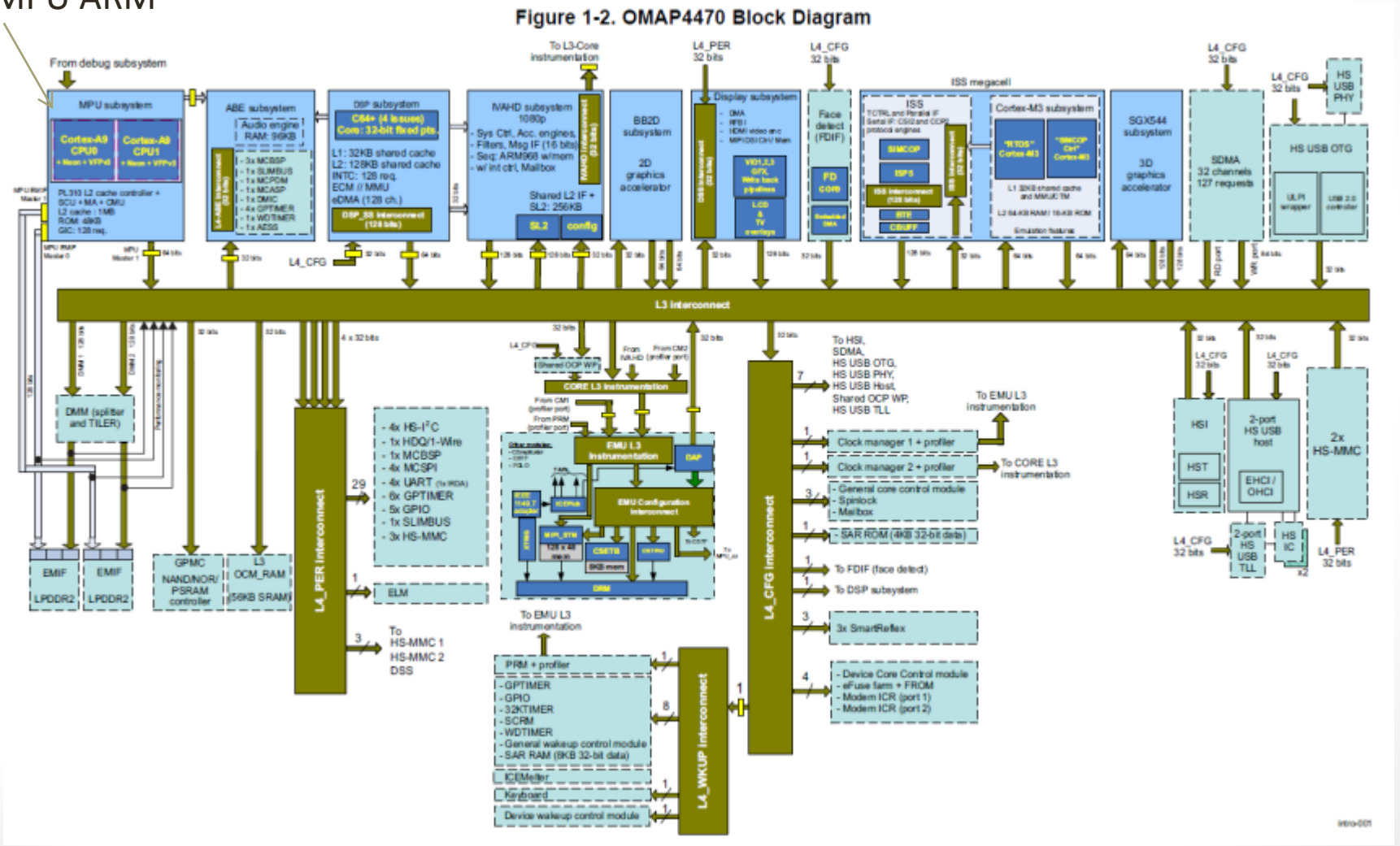
- Pourtant, les techniques pour réduire la puissance dissipée n'ont cessé d'évoluer depuis 1995



P. Magarshak DATE-2011

Exemple d'un SOC avec un OMAP 4470 de Texas Instruments
Le processeur à N cœurs : un sous-système parmi d'autres dans le SoC.

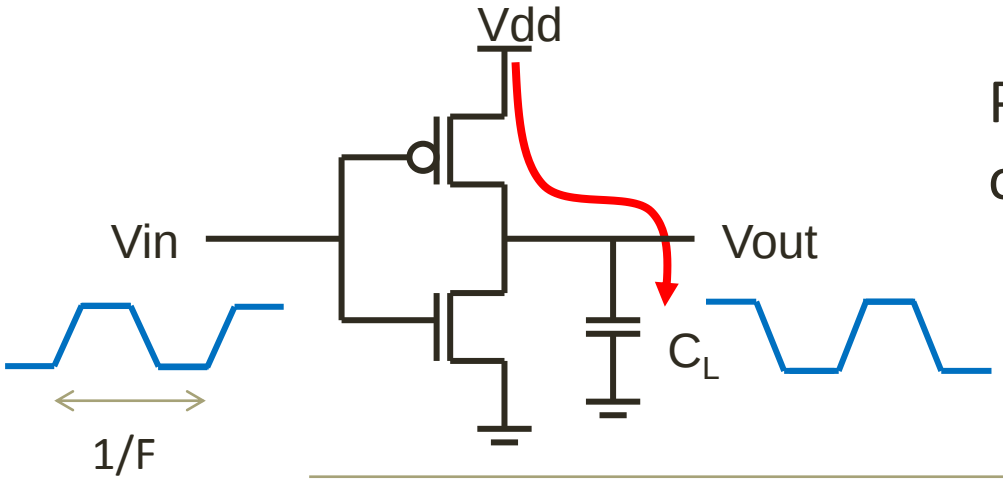
Le MPU ARM



Contenu de l'exposé

- Introduction
- Quelques éléments sur la conception Low Power
- *Power Intent* et *Clock Intent*
- Quelques éléments sur le flot de conception low power
- Modélisation low power au niveau TLM
- Exemple
- Conclusion

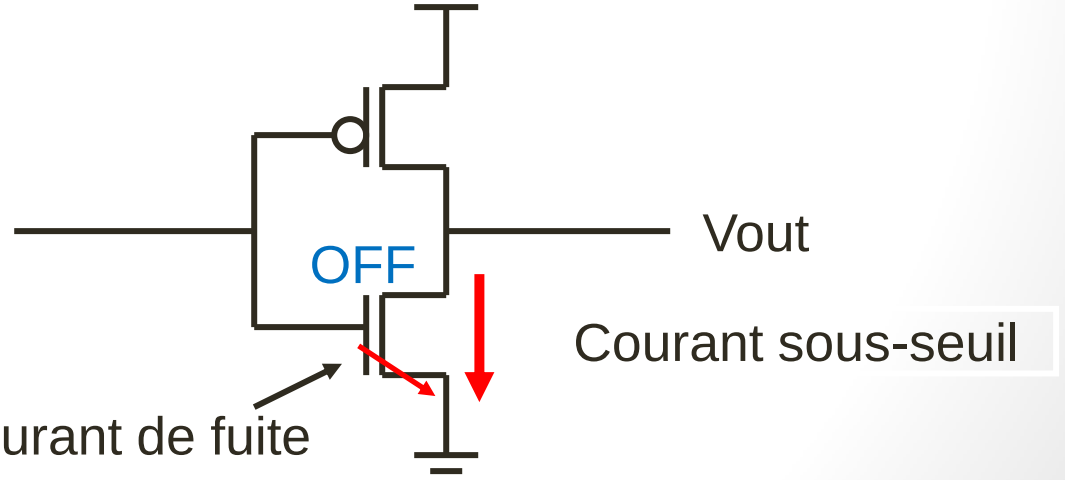
Puissance consommée $P_{total} = P_S + P_D$



Puissance « dynamique »
ou de commutation

$$P_D = C_L V_{DD}^2 F$$

Puissance statique liée aux courants de fuite

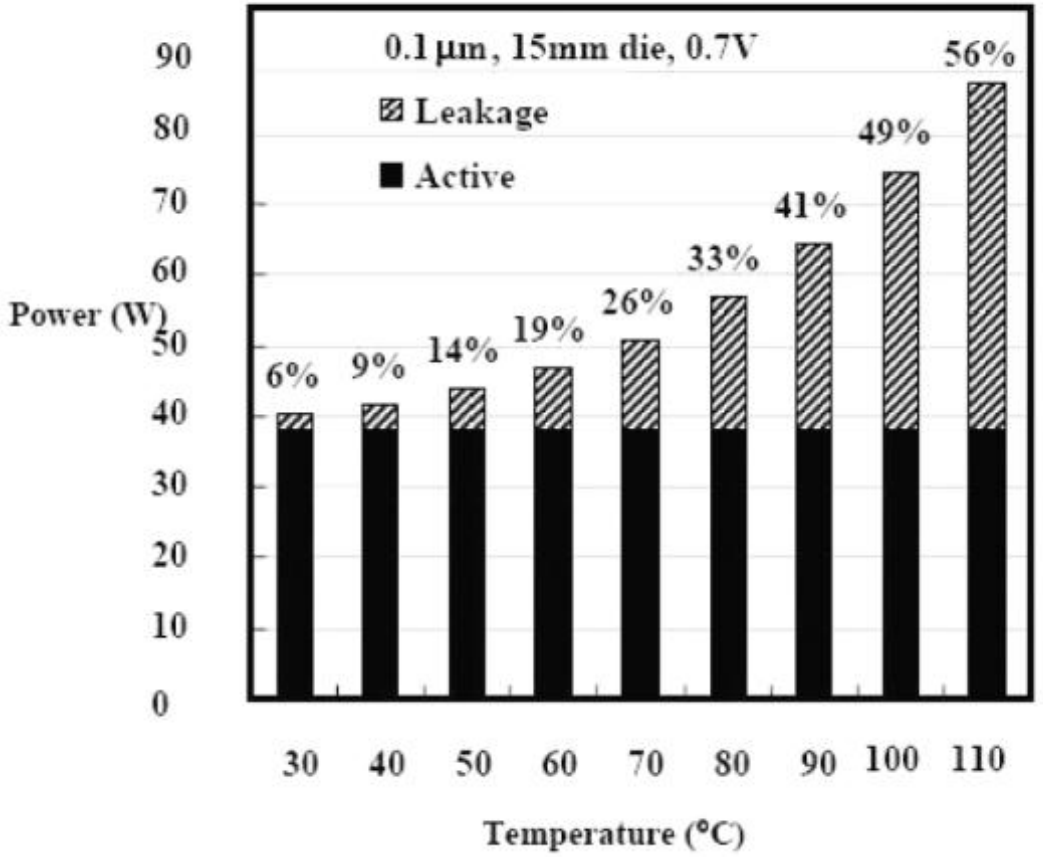


$$P_S = V_{DD} I_{leakage}$$

$$I_{leakage} = f(T°)$$

Effet de la température sur la puissance consommée

Source : Vivek De, Intel



F. Fallah, M. Pedram 2005

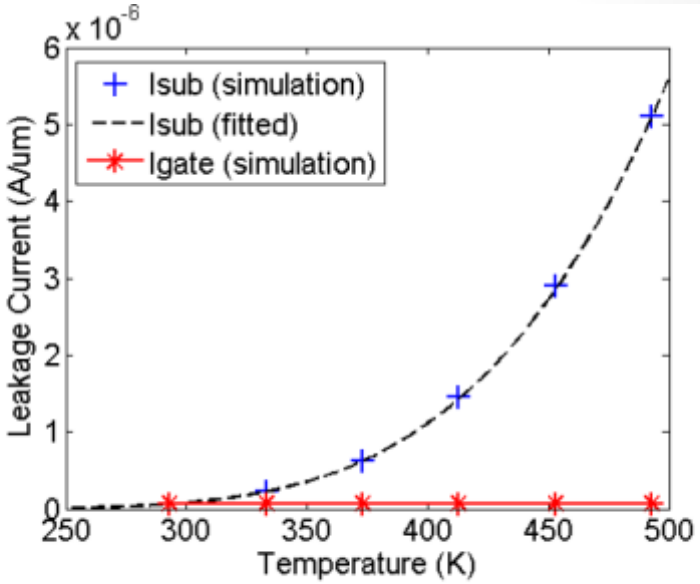


Figure 1. Leakage current components varying with temperature for nMOS FinFET transistor.

I.H. Choi et al., ICCAD'2006

Mécanismes pour réduire la consommation

$$P_D = C_L V_{DD}^2 F$$

- Réduction de la puissance dynamique : DVFS
  - Temps de traversée d'une porte : $Delay \propto \frac{V_{dd}}{(V_{dd} - Vt)^n}$ V_t : tension de seuil
  - En diminuant la fréquence F on peut diminuer aussi V_{dd}

    Dynamic Voltage and Frequency Scaling (DVFS) :
    Technique efficace pour gérer la puissance dynamique en fonction de la charge de calcul.

    Implique de pouvoir agir localement sur V_{dd} et F

    Mais pendant le changement de fréquence et de tension le système et soit bloqué soit fortement ralenti (temps stabilisation horloge).

$$P_D = C_L V_{DD}^2 F$$

- Réduction de la puissance dynamique : Parallélisme

1 cœur

$P_D = C_L V_1^2 F_1$

2 cœurs

$F_2 = F_1/2$

$V_2 < V_1$

$P_D = 2C_L V_2^2 F_1/2$

Rapport des puissances dynamiques : $\frac{2C_L V_2^2 F_1/2}{C_L V_1^2 F_1} = V_2^2/V_1^2$

- Réduction de la puissance dynamique : Multi-Voltage V_{dd}

1 seul Vdd : toutes les cellules sont alimentées avec Vdd même si elle ne fonctionnent pas à la même fréquence

Plusieurs Vdd : nécessite de partitionner le système en différents domaines

$$P_D = C_L V_{DD}^2 F$$

- Réduction de la puissance dynamique : Clock Gating

  Si l ’état d’un composant ne nécessite pas d’évoluer : inhiber son horloge

  - Principe peut s’appliquer :

    - A grain fin :

      Exemple : une partie d’un registre de pipeline ne fait pas de sens dans l’exécution de certaines instructions d’un processeur

    - A grain moyen :

      Horloge d’un cache d’instructions tant qu’un cache miss n’est pas résolu (mono-thread)

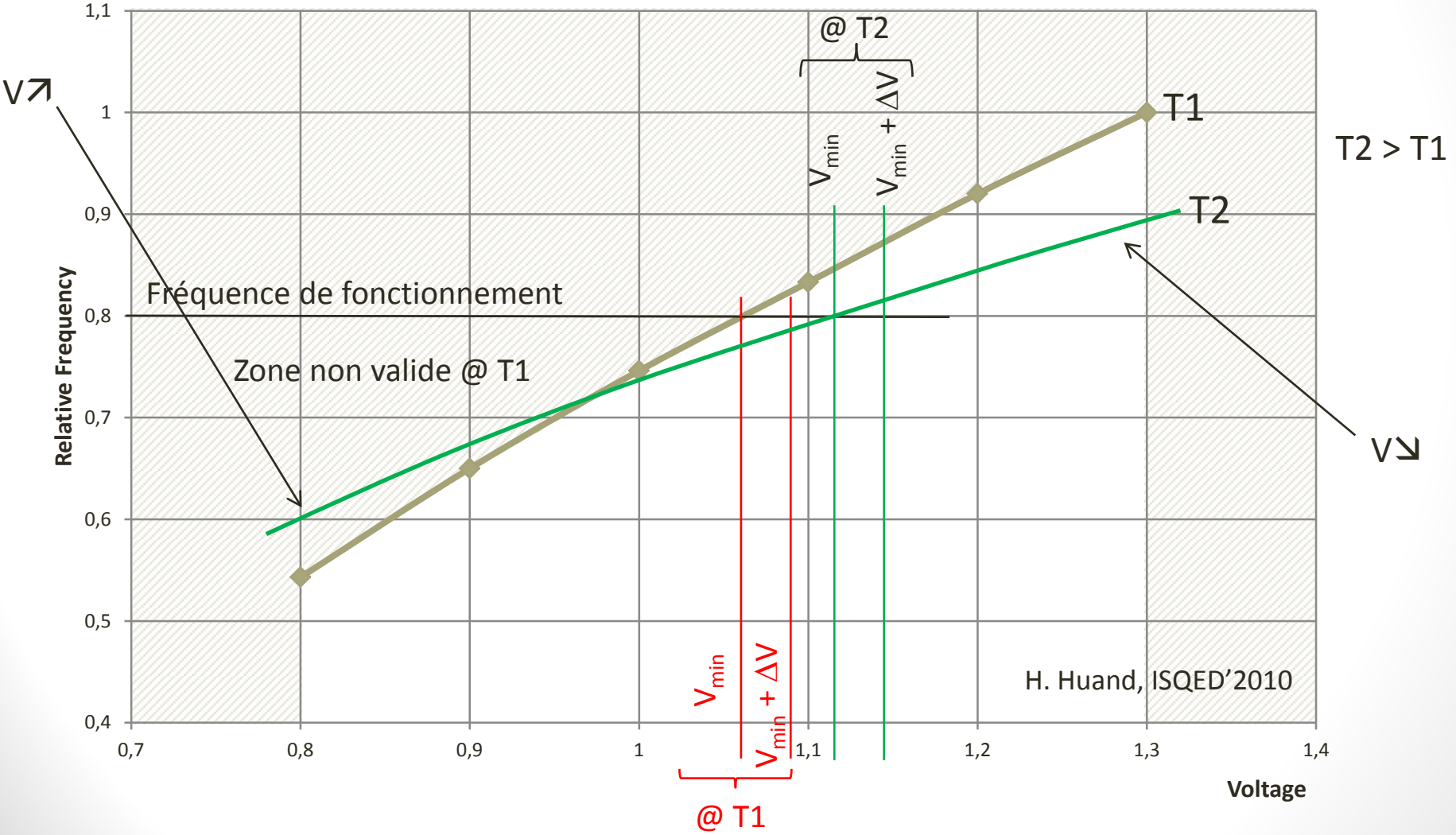
    - A gros grain :

      L’unité de traitement audio sur occurrence d’un silence

$$P_D = C_L V_{DD}^2 F$$

- Réduction de la puissance dynamique : AVS

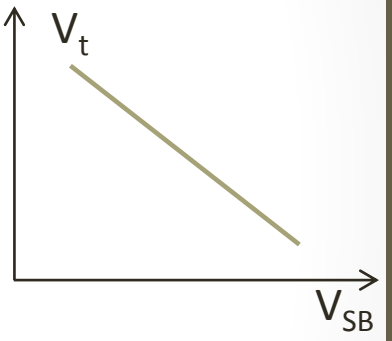
Adaptive Voltage Scaling : adapter V en fonction de T° (+ variabilité de process)



- Mécanismes pour réduire la puissance statique : Body Biasing

Transistor NMOS : si au lieu de mettre le substrat à 0V (comme la source), V_{SB}= 0, si on applique une tension $V_{SB} < 0$, alors cela revient à accroitre la valeur de V_t du transistor.

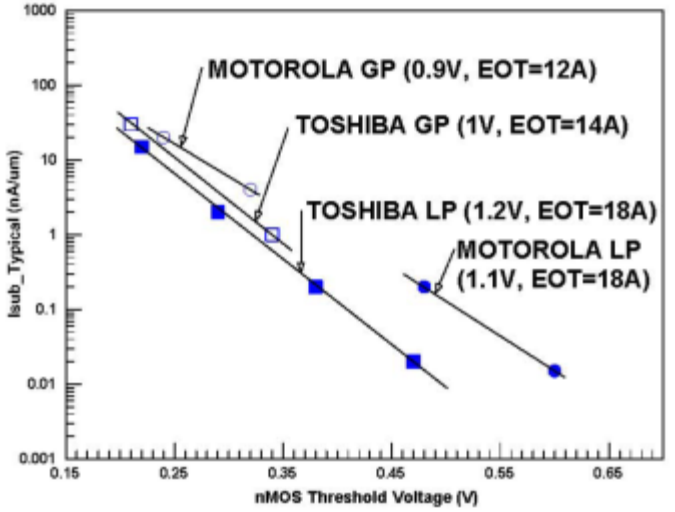
On ajuste alors dynamiquement V_{SB} en fonction des besoins en performances.



$$Delay \propto \frac{V_{dd}}{(V_{dd} - Vt)^n}$$

Augmenter V_t augmente les délais : on doit réduire la fréquence

Augmenter V_t diminue sensiblement la puissance statique !



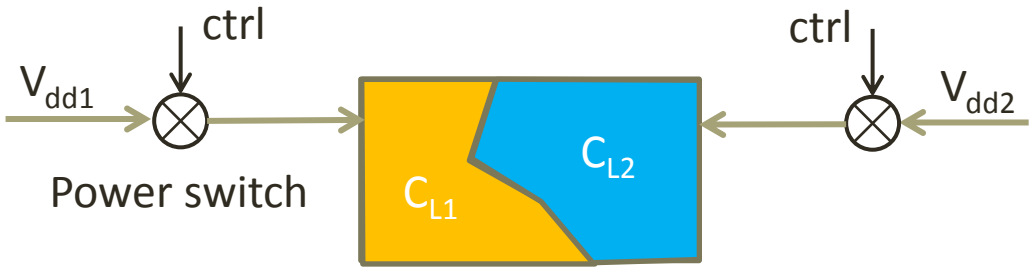
E.N Shauly, J. Low power Elec, 2012

$P_S = V_{DD} I_{leakage}$

- Mécanismes pour réduire la puissance statique : Librairie Multi-V_t
  - Les cellules sont disponibles suivant 3 valeurs de V_t :
    - LVT : cellules rapides mais avec un courant de fuite important
    - HVT : cellules low power (peu de courant de fuite) mais à délais alongés
    - SVT : cellules de type « standard » mi-rapides mi-consommatrices
    - Les cellules LVT sont placées uniquement sur les chemins critiques
    - Distribution sur un IBM P5 :
      - SVT : 65%
      - HVT : 33%
      - LVT : 2%

$P_S = V_{DD} I_{leakage}$

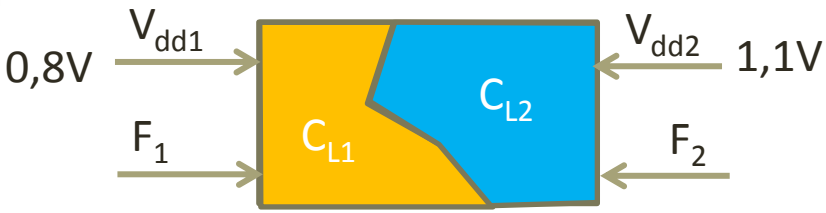
- Mécanismes pour réduire la puissance statique : Power Gating



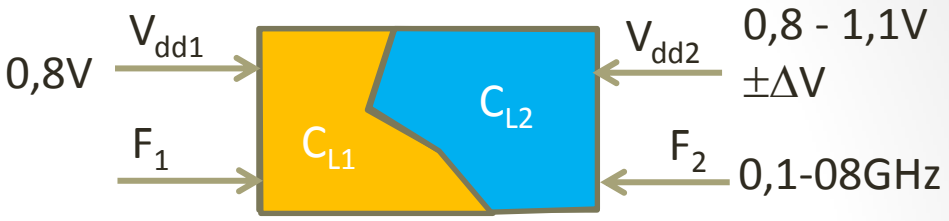
Si un sous-système n'est pas nécessaire pour exécuter une fonctionnalité, on coupe l'alimentation de ce sous-système.

Puissance statique et puissance dynamique sont annulées.

Quelles techniques sont pertinentes dans une modélisation au niveau global de l'architecture système ?



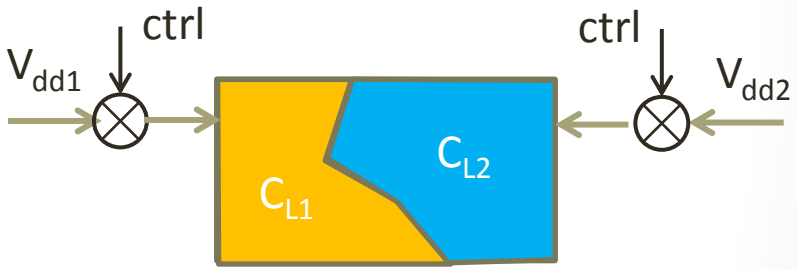
Multi-V_{dd} pour l'évaluation du power en fonction de l'activité



Eventuellement AVS : évaluation du power

Implique des modèles thermiques couplés

DVFS : pénalités en temps, évaluation power



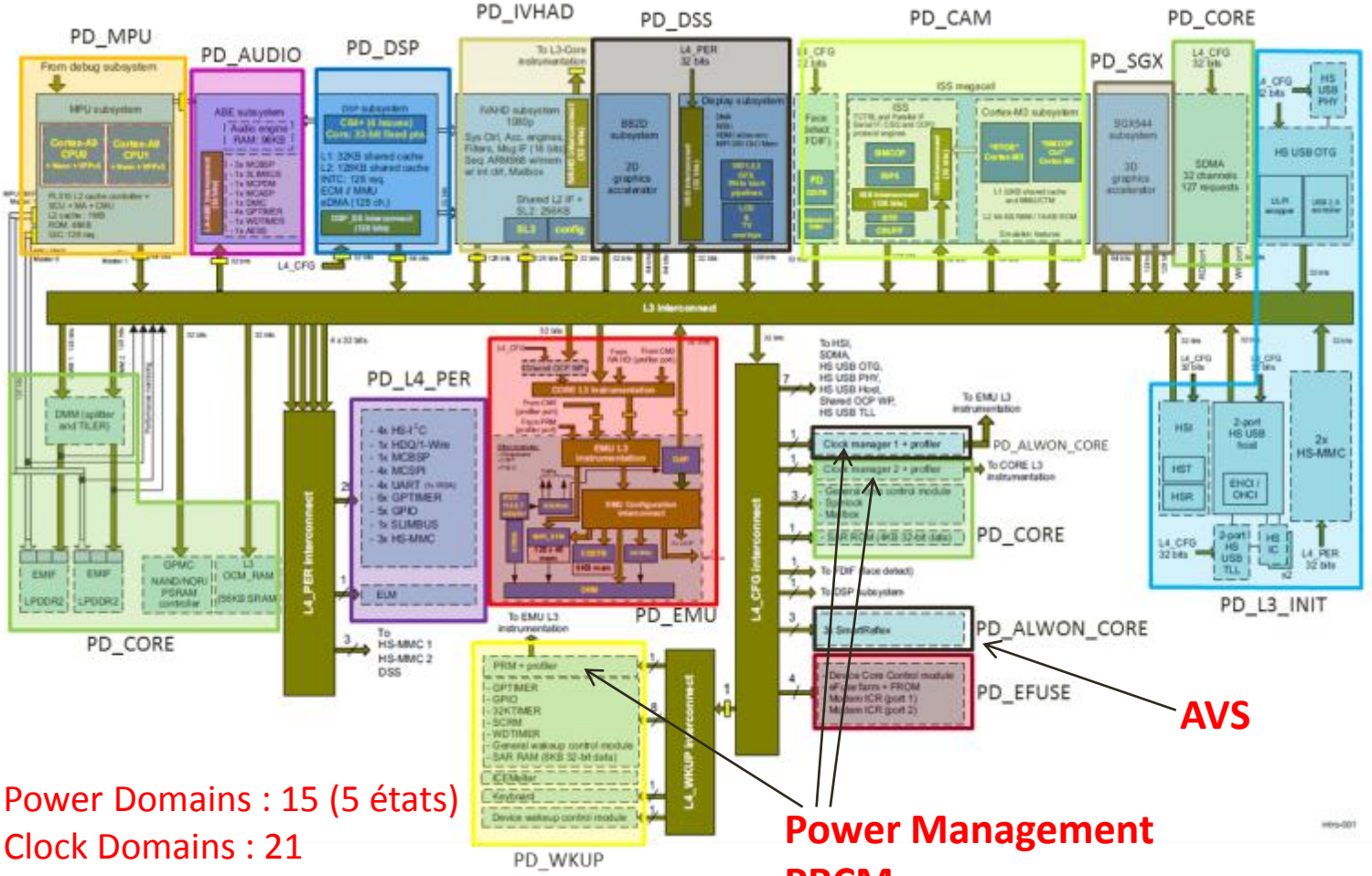
Power Gating :
pénalités en temps, évaluation power

Clock Gating : évaluation power

Contenu de l'exposé

- Introduction
- Quelques éléments sur la conception Low Power
- *Power Intent* et *Clock Intent*
- Quelques éléments sur le flot de conception low power
- Une approche développée dans le projet ANR HOPE
- Exemple
- Conclusion

Retour sur l'OMAP 4470



Plus de 300 blocs IP

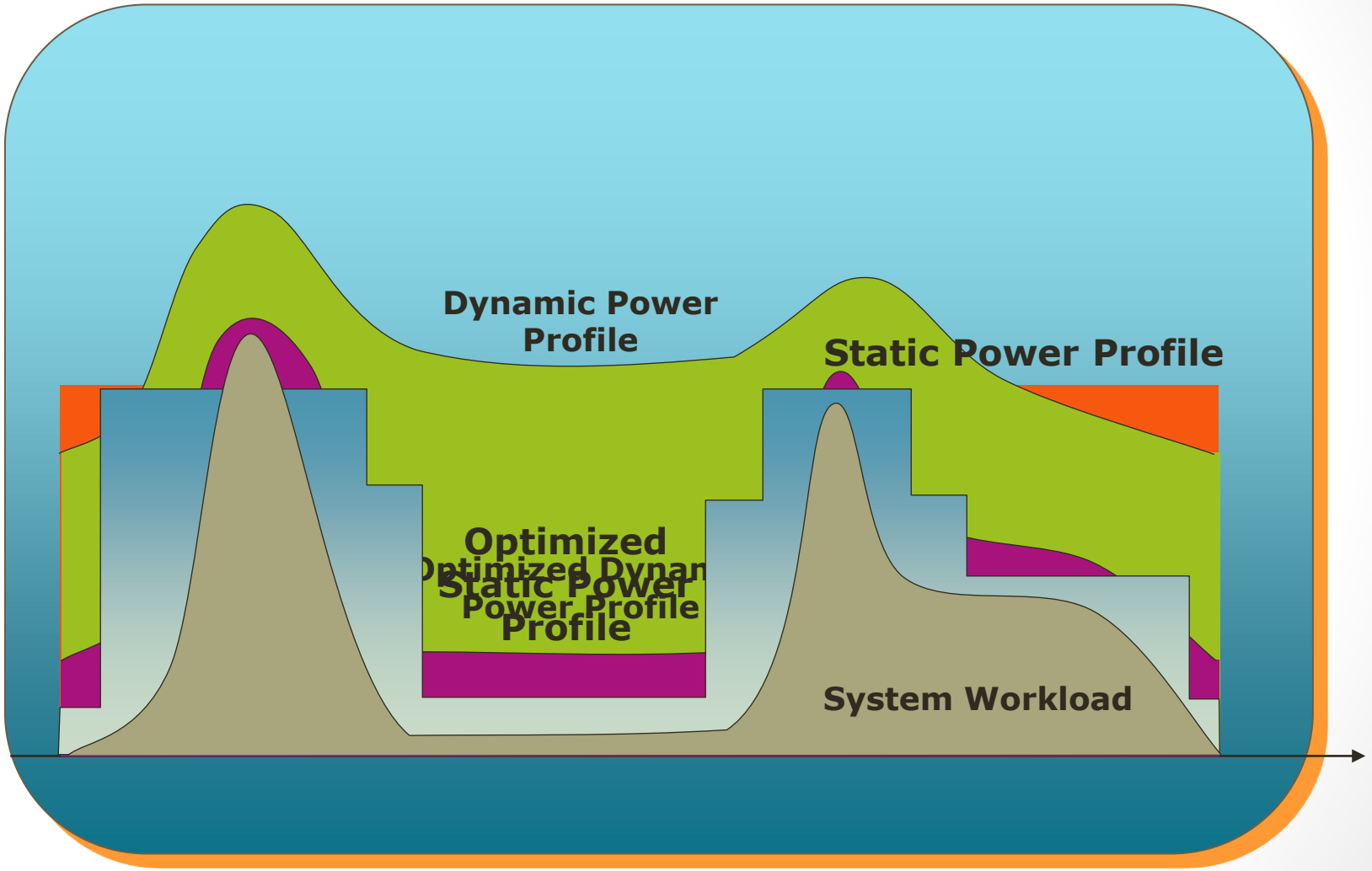
Ref. Manuel du PRCM : ~ 700 pages

Nb Power Domains : 15 (5 états)
Nb Clock Domains : 21
Nb AVS controllers : 3

Power Management PRCM

Comment exploiter par software toutes les possibilités d'état de tous ces domaines ? Optimum ?

Objectif : optimiser le profil de puissance en fonction de la fonctionnalité à exécuter.



ITRS 2011 :

Design Technology Improvement	*Year*	*Improvements*		*Description*
		Dynamic	*Static*	
Low Power Physical Libraries	Before 2011	1.50	1.50	Optimizing transistor size, layout style and cell topology for the standard-cell library
Back Biasing		1.00	1.35	Biasing wells of devices independently of the sources to shift the threshold voltage
Adaptive Body Biasing (ABB)		1.20	2.00	Delivering a positive or negative voltage below a transistor to reduce leakage
Power Gating		0.90	10.00	Turning off the power supplies to idle blocks for leakage reduction
Dynamic Voltage/Frequency Scaling (DVFS)		1.50	1.00	Dynamic management of supply voltage and operating frequency for power reduction
Multilevel Cache Architecture		1.00	1.20	Reduce amount of off-chip memory accesses for performance improvement and power reduction
Hardware Multithreading		1.00	1.30	Using multithreads to improve hardware utilization with leakage reduction
Hardware Virtualization		1.00	1.20	Using one physical server to support multiple guest operating systems simultaneously
Superscalar Architecture		1.00	2.00	Parallel instruction issueing and executing for performance improvement and power reduction
Symmetric Multiple Processing (SMP)		1.50	1.00	Lowering the frequency by using multiple processors and the parallel programming
Software Virtual Prototype	2011	1.23	1.20	Allow the programmer to develop software prior to silicon
Frequency Islands	2013	1.26	1.00	Designing blocks that operate at different frequencies
Near-Threshold Computing	2015	1.23	0.80	Lowering Vdd to 400 - 500 mV
Hardware/Software Co-Partitioning	2017	1.18	1.00	Hardware/software partitioning at the behavioral level based on power
Heterogeneous Parallel Processing (AMP)	2019	1.18	1.00	Using multiple types of processors in a parallel computing architecture
Many Core Software Development Tools	2021	1.20	1.00	Using multiple types of processors in a parallel computing architecture
Power-Aware Software	2023	1.21	1.00	Developing software using power consumption as a parameter
Asynchronous Design	2025	1.21	1.00	Total Non-clock driven design

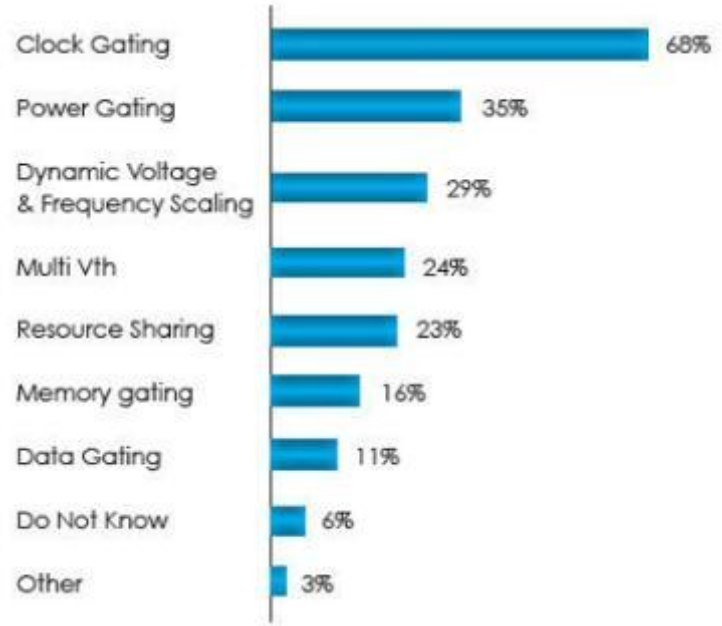
Décomposition en Power Domains & Clock Domains

- Objectif : ne mettre en Power-On que les domaines nécessaires à l'exécution de la fonctionnalité courante. Les autres peuvent être mis en power-off.
- Les domaines dans l'état Power-On peuvent avoir leur fréquence d'horloge ajustée (individuellement) pour réaliser la fonctionnalité courante avec le niveau de performance juste suffisant (Active Power Management, e.g. DVFS).
- Les composants inactifs dans un domain Power-On peuvent avoir leur horloge en état Clock Gated.
- Dans chaque power domain, le choix de cellules de type LowVT ou HighVT permet d'ajuster performance vs puissance statique.

Décomposition en Power Domains & Clock Domains

- Difficultés :
  - Quelle répartition des IPs dans les Power Domains et Clock Domains ?
    - Les contraintes de performances, de temps réel doivent être vérifiées
    - Réduction de la consommation d'énergie/puissance doit être efficace :
      - Le profile power doit être le plus proche possible du profile workload
      - Le sur-coût induit doit être limité
    - Et ceci pour toutes les différentes fonctionnalités à exécuter par le système
- Problème de nature NP-Complet

Les 10 meilleures techniques pour réduire la consommation d'énergie

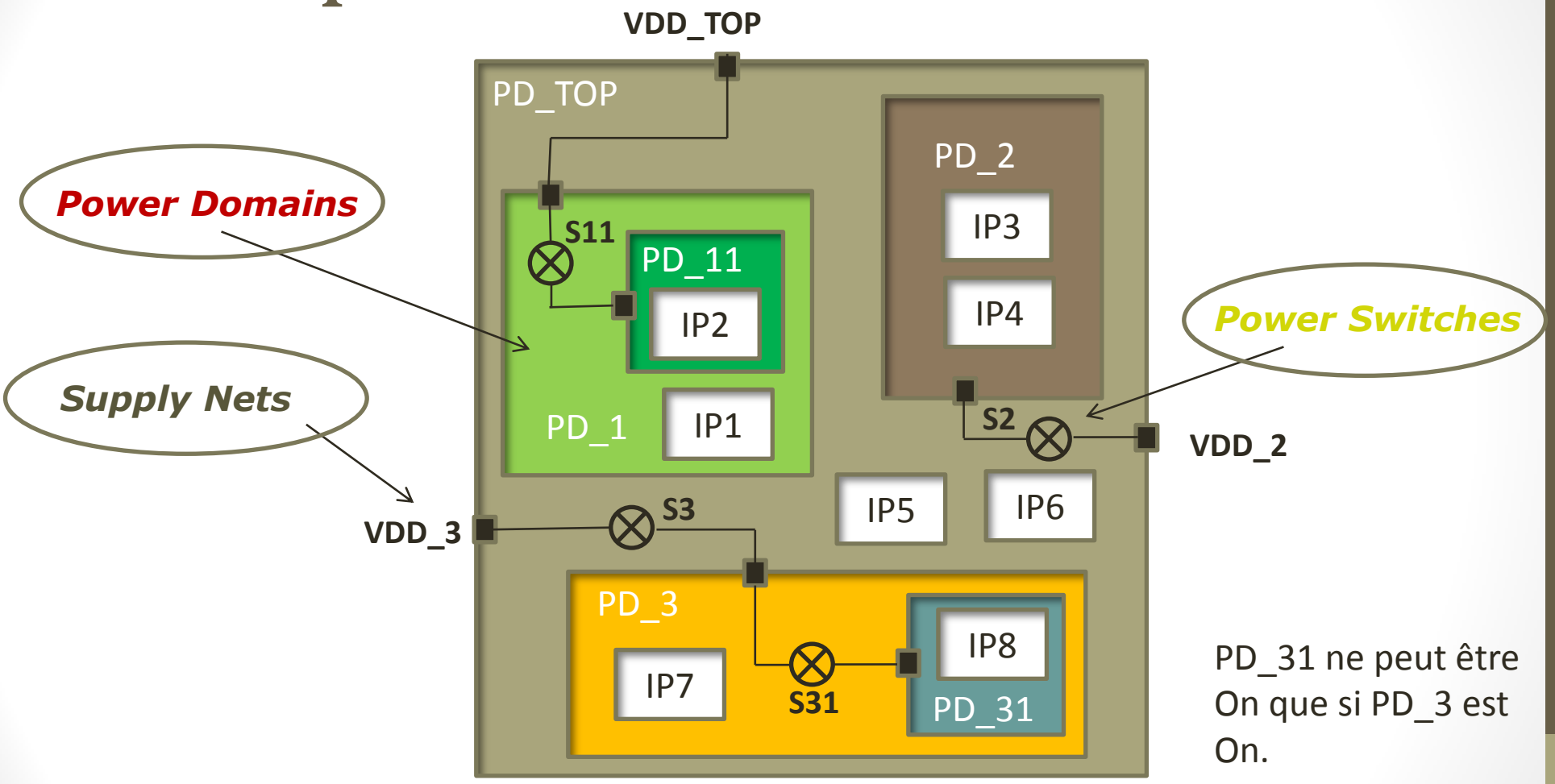


S. McCloud, Calypto Design Systems, Inc
Low-Power RTL Report 2012, SOCcentral.com

Power Domains d'une architecture matérielle

- Un Power Domain :
  - Un sous-ensemble de composants du système qui reçoivent leur alimentation depuis un même Supply Net au travers d'un Power Switch
  - Un Power Domain peut être mis dans différents états de fonctionnement : les Power States.
    - Par exemple : Active, Sleep, Deep Sleep,…
  - Les Power States sont définis par l'état des Power Switches sur les Supply Nets et les valeurs des tensions sur les Supply Nets.

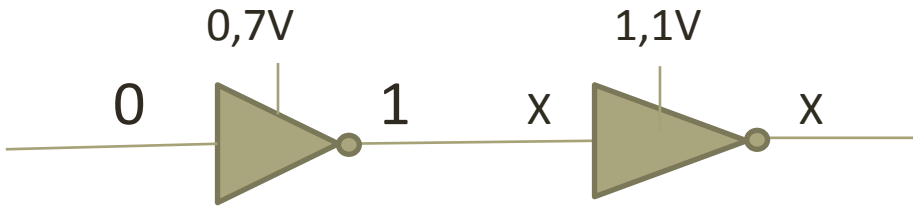
Exemple :



PD_31 ne peut être On que si PD_3 est On.

Power Domains et logique fonctionnelle

- Des power domains alimentés avec des tensions différentes peuvent induire des erreurs logiques fonctionnelles :

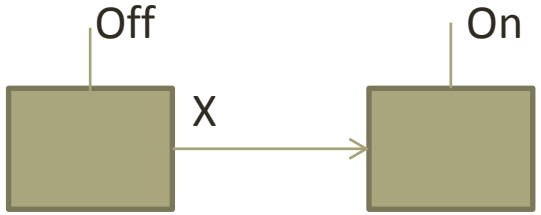


Simulation : Un « 1 » logique ne décrit pas le niveau de tension correspondant :
Le simulateur logique doit intégrer ces cas.

➢ Architecture : Introduire des Level Shifters

- Transition d'un power domain : On puis Off puis On
  - Quel est l'état logique des registres, mémoires internes aux IPs ?
    - ➢ Introduire des Retention Registers si des informations vivantes
    - ➢ Gestion de signaux de Reset

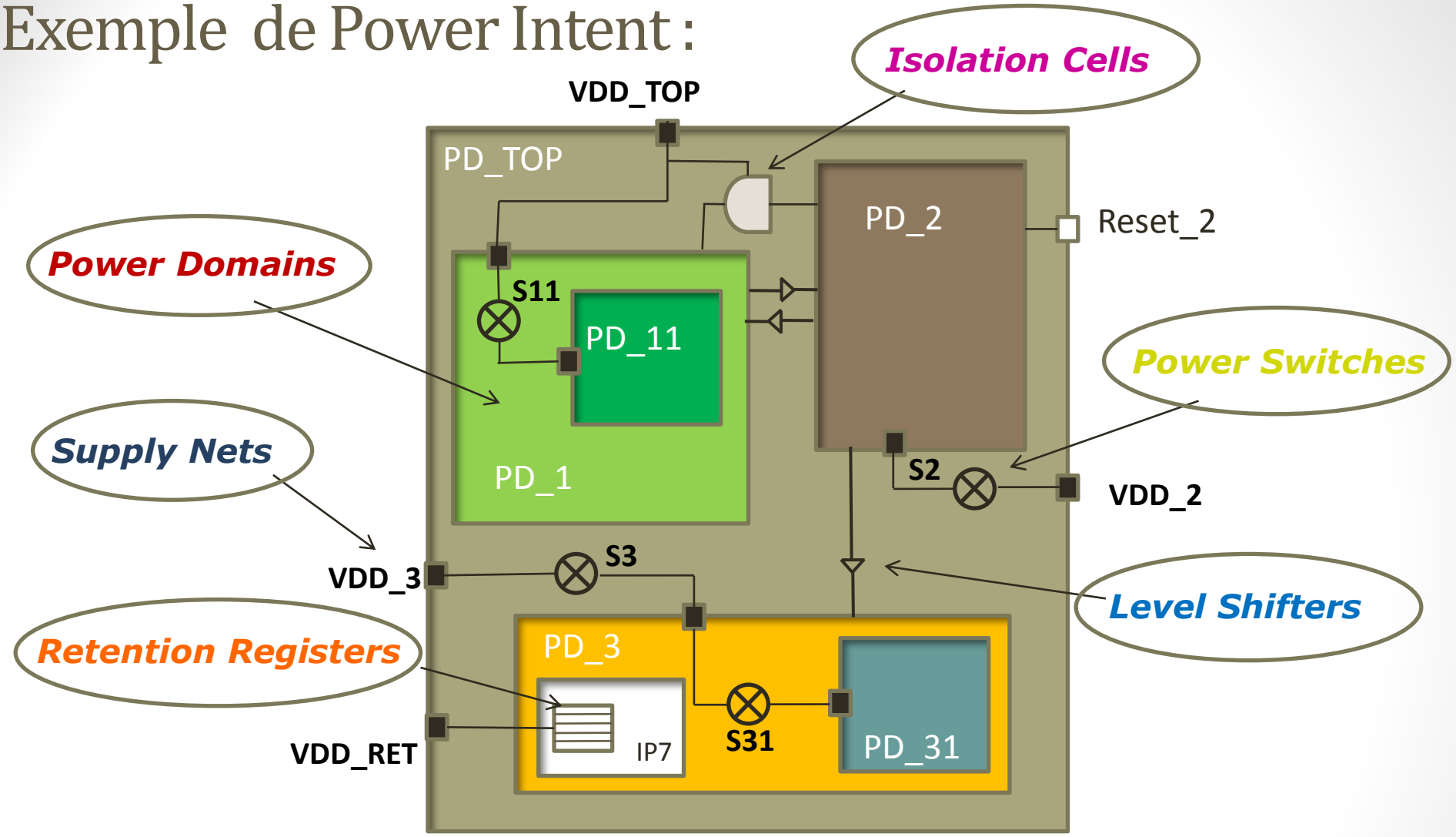
- Power Domain Off connecté à un Power Domain On :
  - Niveau logique de l'entrée non alimentée ?



Simulation : forcer des valeurs X sur ces entrées

➢ Architecture : Introduire des Isolation Cells

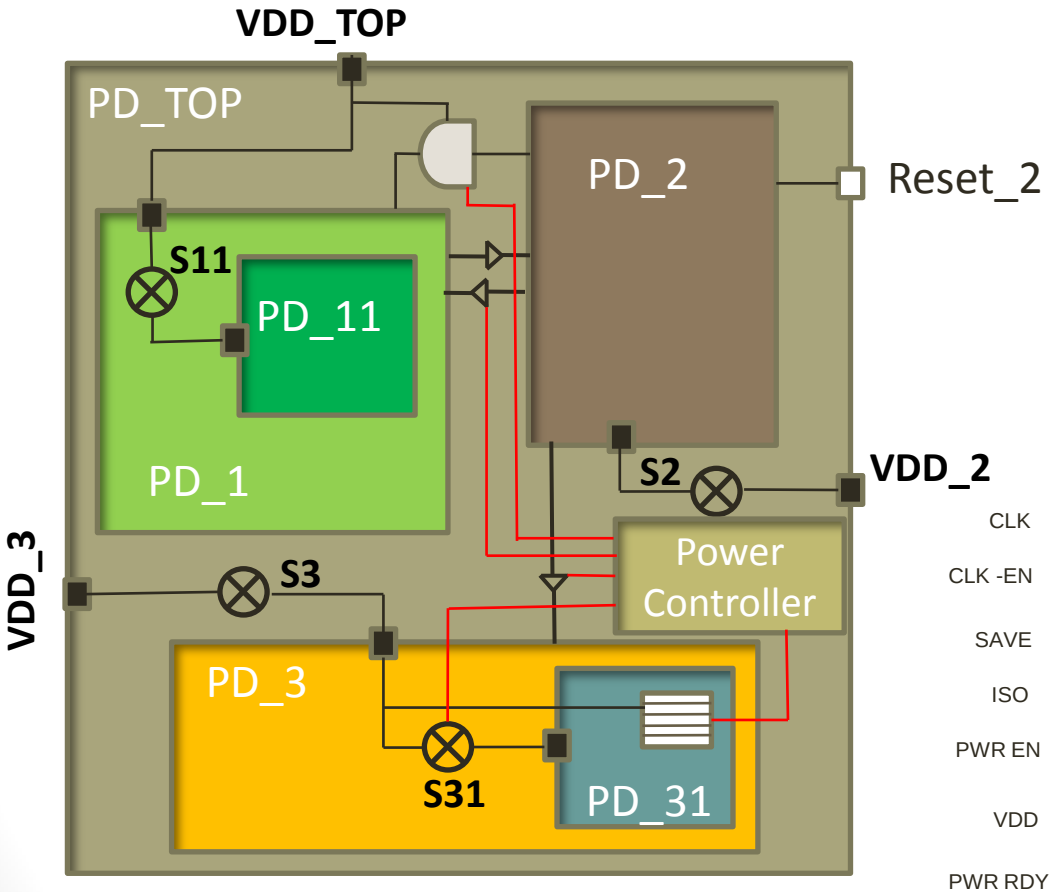
Exemple de Power Intent :



Eléments qui font du sens au niveau TLM :

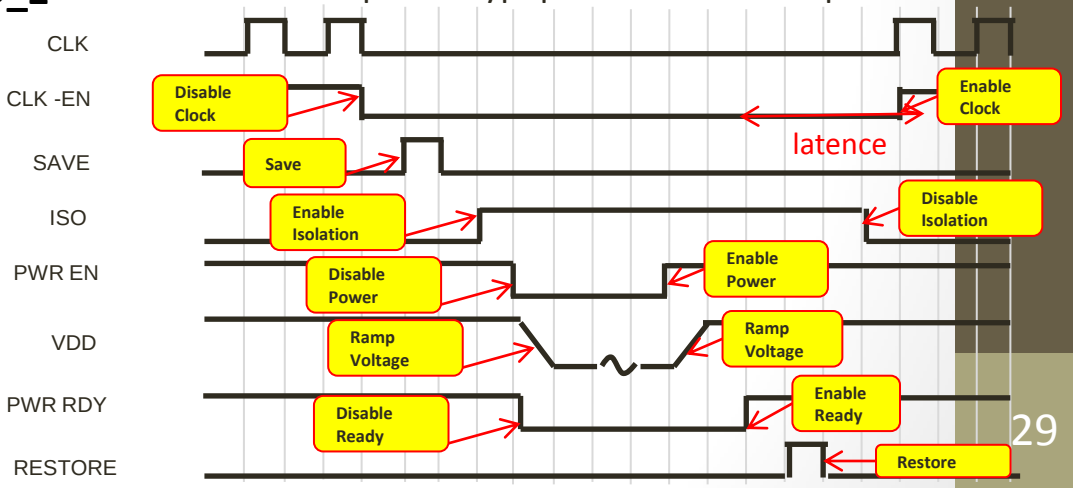
- Power Switches : effet sur le power
- Supply nets : valeur du power
- Power domains : vérification structure
- Retention Registers : vérification fonctionnelle

Changement d'état : séquence de contrôle



- Changer d'état on/off implique de respecter un certain timing des signaux de contrôle des éléments de l'architecture power

Séquence typique de contrôle du power



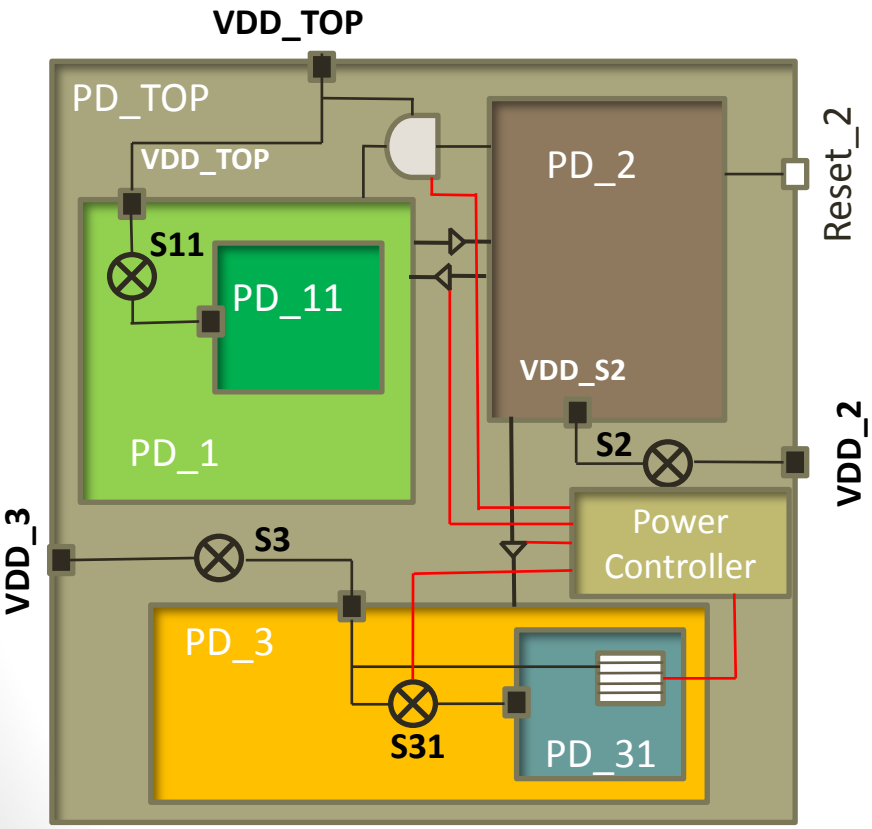
Gary Delp et al.

Timing et Power States

- Utiliser des power switches intégrés (on-chip power gating) sur les rails d'alimentation permet d'obtenir des latences plus faibles qu'avec des power switches externes.
- La latence du passage Off vers On dépend de la taille du power domain, des caractéristiques des rails d'alimentation, des caractéristiques des power switches :
- Mais :
  - Diminuer la latence peut se faire en augmentant le nombre de power domains (et donc réduire leur taille)
  - Contrôler plusieurs petits domaines est alors plus complexe à réaliser et à valider
  - Sur-coût dû aux power switches et au contrôle associé
- Ainsi chacun des Power Domains doit être justifié par rapport aux gains en énergie attendus, à la capacité à les exploiter et à la complexité induite.

UPF : formalisme standard permettant de décrire un Power Intent

UPF : Standard IEEE 1801




```
set_scope PD_TOP
create_power_domain PD_1 -elements {SRAM} -supply { VDD_TOP}

# On définit un supply set contenant les supply nets du power domain :
create_supply_set PD_1.SSVDD_TOP –function {power VDD_TOP} \
        –function {ground GND} –update

# VDD_S2 est interne, on le déclare, puis on déclare le supply set :
create_supply_net VDD_S2
create_supply_set PD_2.SSVDD_S2 –function {power VDD_2} \
        –function {ground GND} –update
# On crée le power switch puis le power domain PD_2
create_logic_port PS_PD_2 –direction in
create_power_switch S2 -output_supply_port {out_vdd SSVDD_S2.power}\
        -input_supply_port {in_vdd VDD_2} \
        -control_port {ctrl PS_PD_2} \
        -on_state {ON in_vdd !ctrl } -off_state {OFF ctrl}
create_power_domain PD_2 –elements{BUS Timer} -supply {SSVDD_S2} \
        -scope PD_TOP
……..
```

M. Keating, et al, Low Power Methodology Manuel for System on Chip Design, Sringer, 2007

Clock Domain d'une architecture matérielle

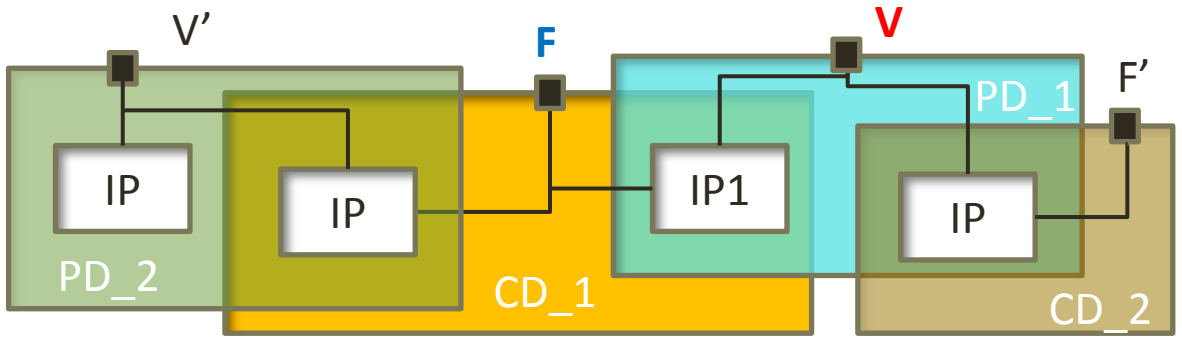
- Un Clock Domain (définition non communément adoptée) :
  - Un sous-ensemble de composants (design elements de UPF) du système qui ont en commun la même source d'horloge :
    - Les design elements d'un Clock Domain ont tous des horloges synchrones et en phase.
  - Il n'y a pas de hiérarchie dans les clock domains :
    - si un sous-domaine reçoit son horloge source d'un clock domain parent, toutes les horloges du sous-domaine sont synchrones et en phase avec celle du clock domain parent, c'est donc le même clock domain.
- Contrairement au power intent (e.g. UPF), actuellement il n'y a pas de standard pour la description d'un clock intent.
  - Les outils de synthèse RTL et de HLS savent introduire du clock gating au niveau d'un composant.

- Il n’y a pas a priori de relation entre clock domains et power domains, mais cela dépends des règles de chaque concepteur.

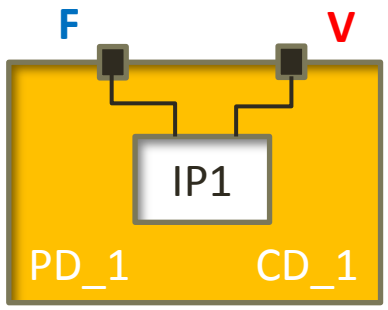
- Le Clock Domain complet peut être mis en état Clock Gated ou individuellement chaque composant du clock domain

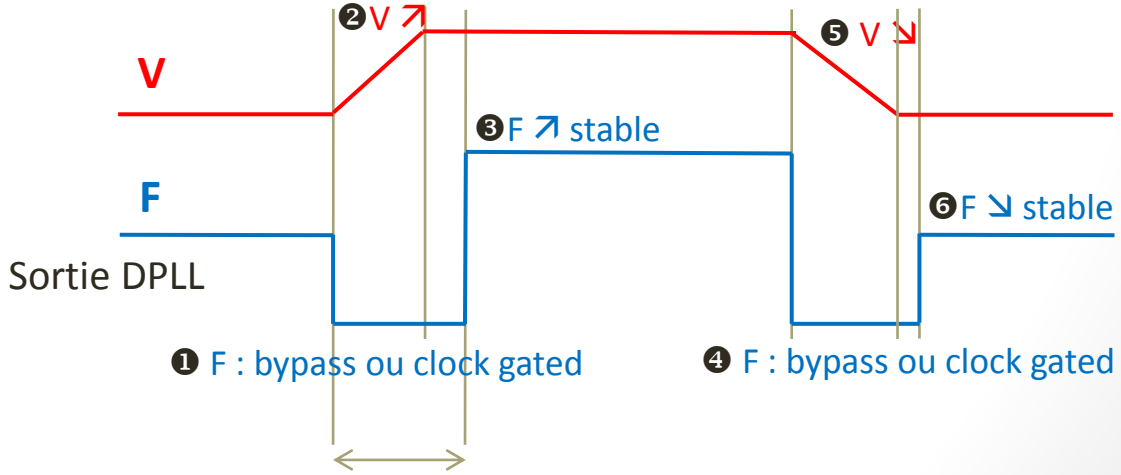
- Cas du DVFS : associe Power Domain et Clock Domain
  - Les composants gérés par DVFS qui partagent la même source d'horloge et le même rail d'alimentation (supply net) sont dans un power domain et dans un clock domain qui a priori se confondent.

IP1 géré par DVFS suivant **V** et **F**



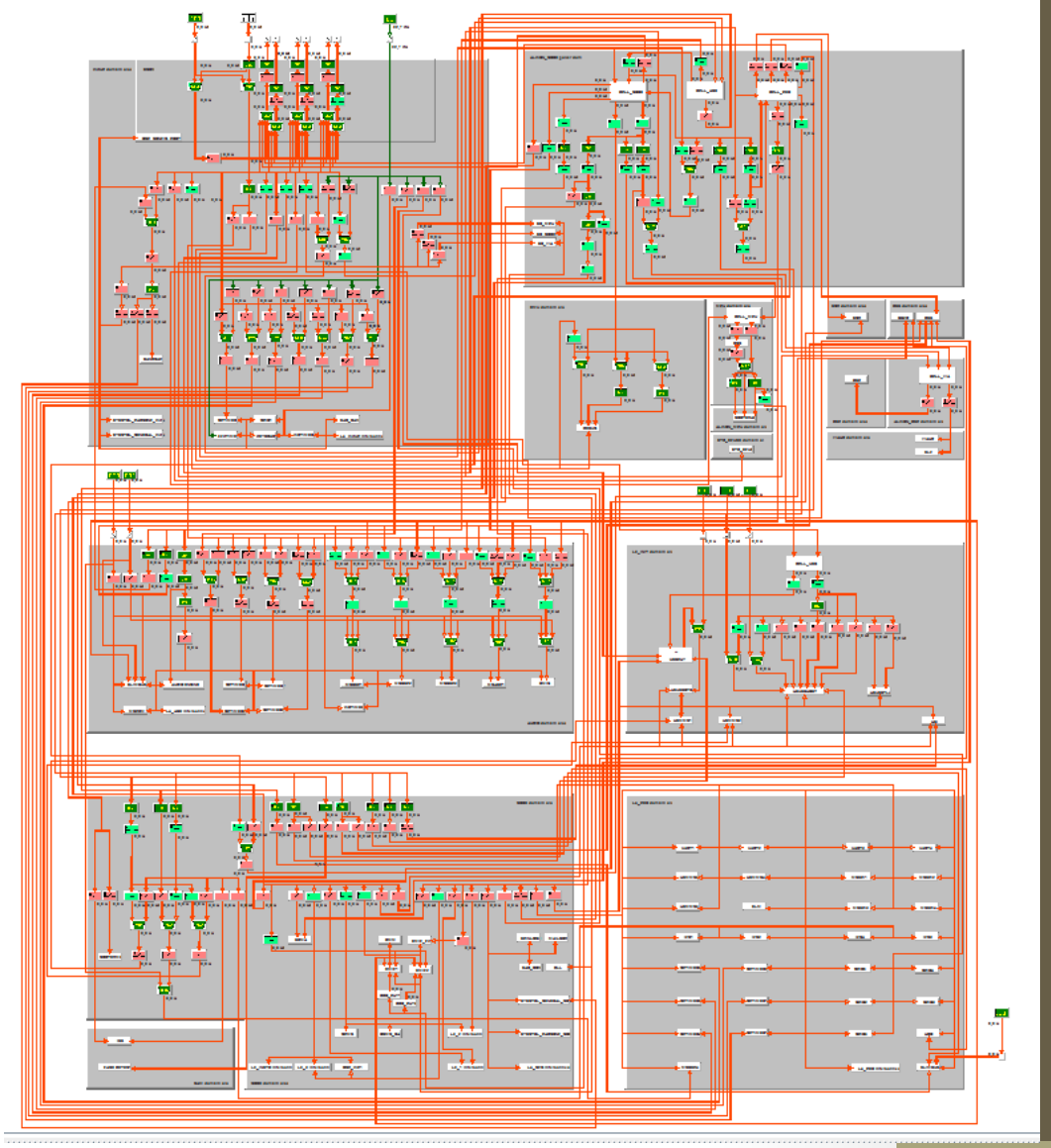
Structure quasiment impossible à contrôler





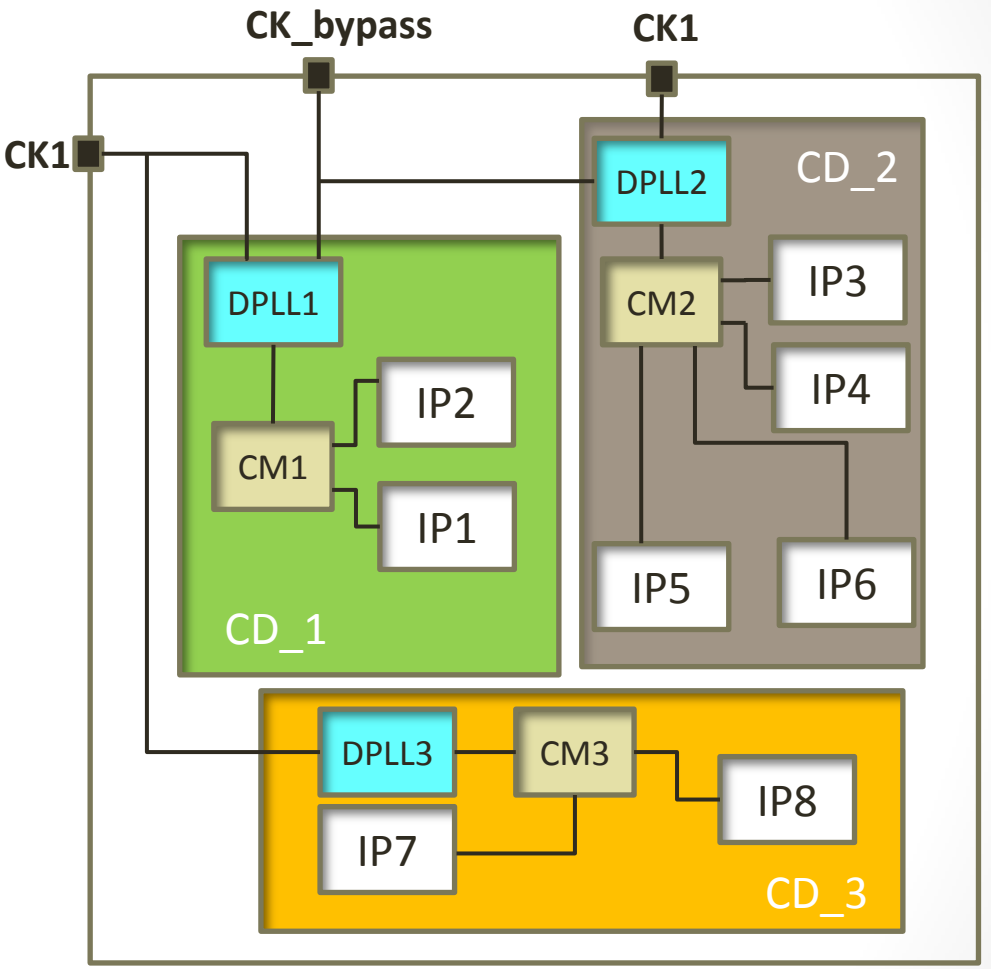
Pénalité en temps sur le fonctionnel

- Source d'horloge :
  - PLL (DPLL)
  - Horloges externes
  - 1 source d'horloge : 1 clock domain
  - Modification fréquence de sortie PLL par programmation des facteurs de division/multiplication
- Gestion des horloges (synchrones) dans un clock domain :
  - Clock Manager (CM) :
    - Clock Gating
    - Facteur de division
    - Multiplexing
- Clock gating, facteurs de division :
  - Écriture dans des registres de configuration des DPLL et CM par un composant de type Master

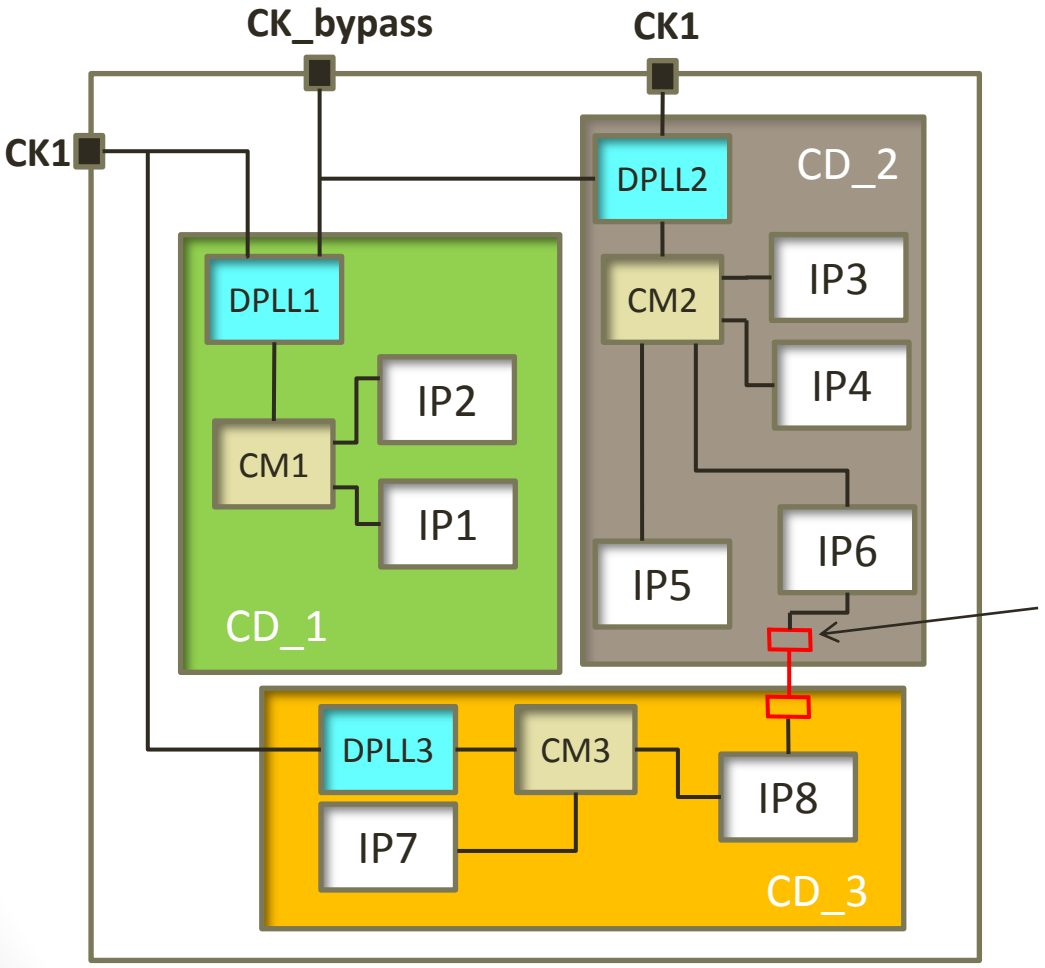


Clock Tree de l'OMAP4470 (TI)
(Clock Tree Tool de TI)

- Exemple de partition en clock domains
- Horloge de « Bypass » (basse fréquence) sert par exemple dans les phases de Reset et pendant les phases de changement de fréquence de l'horloge (DVFS)



Clock Domain Crossing

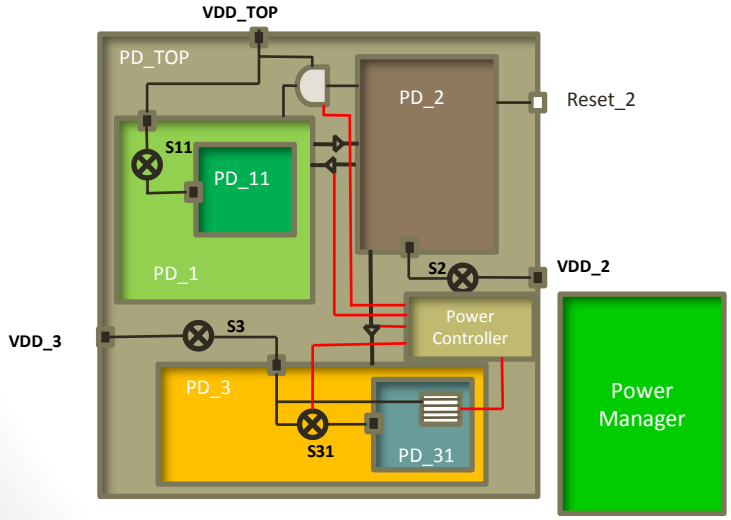


Cellule de synchronisation pour tout signal entre deux IP appartenant à deux domaines d'horloge différents.

Cellules de synchronisation : non pertinent au niveau TLM

Le contrôle : le Power Manager (PM)

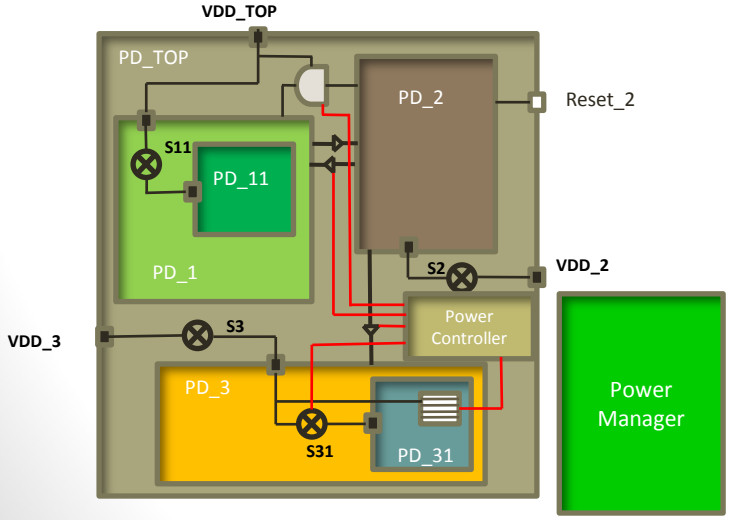
- Le PM implémente les couches basses de la stratégie de power management
  - Interface de contrôle entre le fonctionnel (Sw) et le matériel
    - Fonctionnel = logiciel système + applicatif
  - Contient par exemple les Operating Perfomance Points (OPP) qui déterminent les couples <F,V> admis pour l'architecture



- Le PM contient généralement un microcontrôleur
- Le PM est a priori dans un power domain de type Always-On

Le contrôle : le Power Manager (PM)

- Dans UPF la stratégie de power management peut être représentée par une PST : Power State Table
  - Le passage d'un power state à un autre implique que toutes les séquences de contrôle soient exécutées par le ou les Power Controllers
  - PST : approche statique de définition des power states. Adapté à la simulation pour l'exploration, l'évaluation de l'efficacité d'un power intent
  - Ne concerne que les Power Domains. (UPF)



	VDD_TOP PD_1	S11 PD_11	VDD_2, S2, PD_2	VDD_3, S3, PD_3	S31, PD31
Full Run	1.1V	On	1V	1.1V	On
Boot	1V	Off	1V	Off	Off
Sleep	0.9V	Off	Off	0.9V	Off

Conception low power : les sources d'erreurs

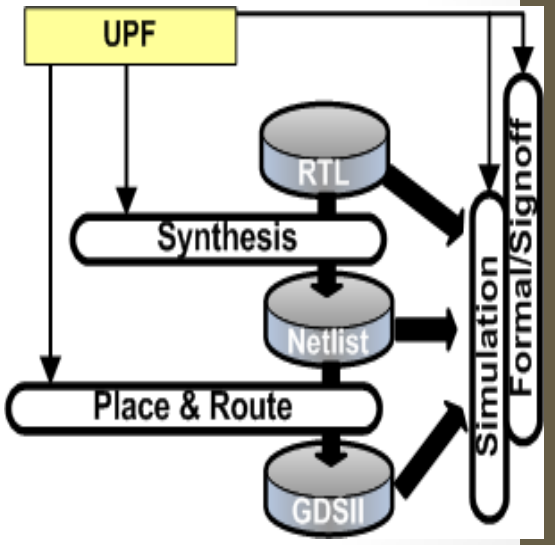
- La structure power/clock est elle correcte ?
  - Présence des cellules d'isolation, des Level Shifters, des registres de rétention
- Le contrôle est-il correct ?
  - Les séquences On => Off et Off => On sont elles correctes ? Les changements de power states (transitions dans la PST) sont elles correctes ?
  - La gestion des signaux de reset et des registres de retention est-elle correcte par rapport aux transitions On/Off des power domains ?
- L'architecture power est elle correcte ?
  - Partitionnement en power domains vs performances vs énergie
  - La stratégie de power management est-elle compatible avec les comportements fonctionnels attendus ?
    - E.g. Pénalités en temps des changements de power states

Contenu de l'exposé

- Introduction
- Quelques éléments sur la conception Low Power
- *Power Intent* et *Clock Intent*
- Quelques éléments sur le flot de conception low power
- Une approche développée dans le projet ANR HOPE
- Exemple
- Conclusion

Le flot de conception RTL low power

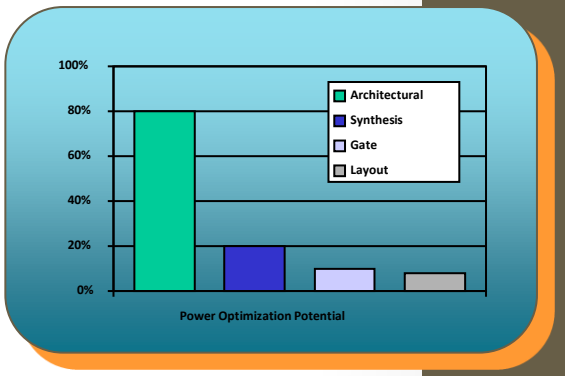
- Les outils EDA permettent la conception d'architectures low power depuis le niveau RTL
  - Le flot classique est complété avec les aspects power



  - La description du Power Intent en UPF est utilisée conjointement avec la description RTL
    - Couvre les étapes de synthèse, simulation et placement/routage.
    - Séparation des aspects fonctionnels (e.g. VHDL) des aspects orientés power (UPF)

Avantages/inconvénients de l'approche

- Méthode outillée : les environnements de CAO automatisent la synthèse et le P&R à partir des descriptions (RTL) du fonctionnel et du power
- Les comportements simulés permettent de représenter l'influence du contrôle du power sur le fonctionnel :
  - estimation de la consommation d'énergie,
  - détection d'erreurs



- Simulation au niveau RTL :
  - Difficultés à simuler un système complet, à valider un power intent en termes de :
    - Gains en énergie au niveau global,
    - Validation de la stratégie power vis-à-vis du fonctionnel (e.g. avec le Sw)

- Ne s'applique qu'à partir du niveau RTL. Pas de standard et peu d'outils pour le niveau TLM

Approches classiques TLM & Power

- Instrumenter le code fonctionnel SystemC-TLM des composants de l'architecture avec du code spécifique pour l'estimation de puissance.
  - Exemple avec librairie TLM-Power du CEA-LETI (M. Moy, Verimag) :

```
void compute(void) {
  while (true) {
    set_value<int>("bandwidth", get_value<int>("bandwidth") + 1);
    set_value<std::string>("mode", "idle");
    update_power();
    wait(1, SC_SEC);
    set_value<std::string>("mode", "run");
    update_power();
    wait(1, SC_SEC); }
  }
```

- Inconvénients de ce type d'approche :
  - Adhérence du code « power » avec le code fonctionnel (pb exploration)
  - On ne décrit pas le contrôle relatif à la stratégie de Power Management
    - On décrit la conséquence de la stratégie qui se trouve noyée dans le code
    - Vérification difficile de la cohérence entre états fonctionnels et états power

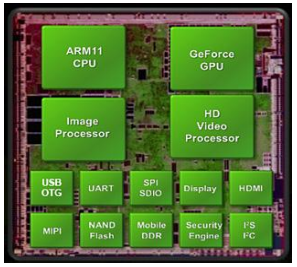
- D’autres approches du même type ont été développées :
- Elles consistent à intégrer des modèles d’estimation du power dans des descriptions SystemC-TLM, e.g. [1], [2], [3], [4], [5], [6]
  - Mais ces approchent visent l’estimation de la consommation en vue d’évaluer ce que pourrait être une stratégie power mais ne concernent pas véritablement le problème de la description du Power Intent au niveau TLM et son contrôle pour une vérification en relation avec les états fonctionnels et les objectifs de performance.
- Ces approches se situent en amont d’un processus de conception (niveau système) mais ne sont pas réellement accrochées à un processus de conception
  - e.g. développement du logiciel système embarqué y compris celui de contrôle du power

Contenu de l'exposé

- Introduction
- Quelques éléments sur la conception Low Power
- *Power Intent* et *Clock Intent*
- Quelques éléments sur le flot de conception low power
- Modélisation low power au niveau TLM
- Exemple
- Conclusion

Un peu de pub : Projet HOPE

Environnement pour la conception système de **systèmes sur puce** optimisés en performance vs puissance consommée vs température.

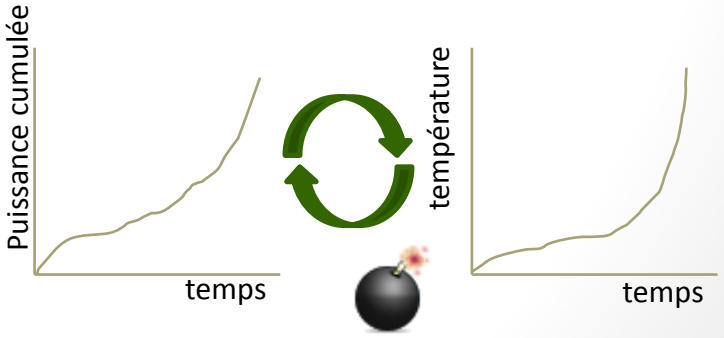


Prototype virtuel : un modèle abstrait exécutable du matériel, supportant l'exécution du logiciel système et applicatif

Standard : SystemC/TLM

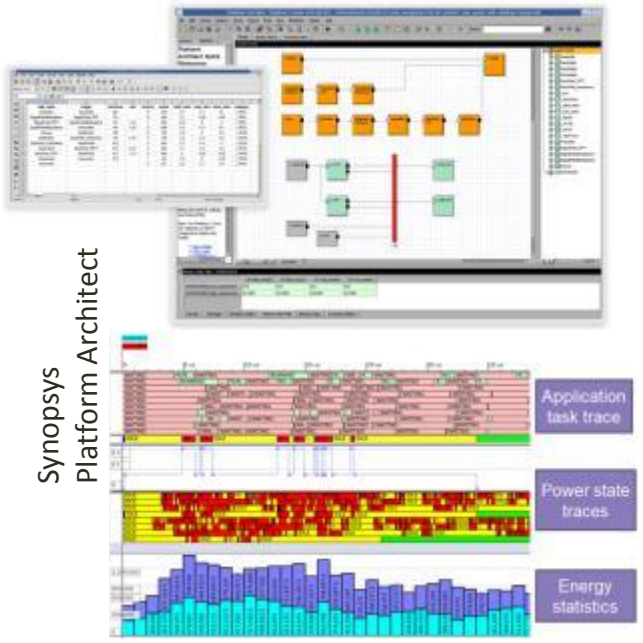
Platform Architect de Synopsys

Consommation et température, Phénomènes interdépendants:



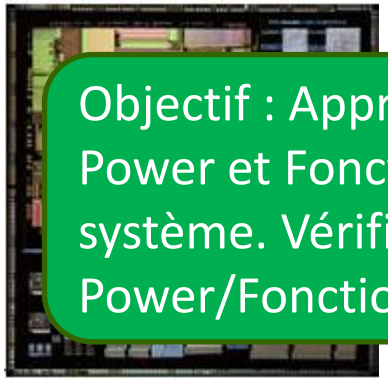
Projet HOPE : partenariat

Modèles pour le prototypage virtuel



Modèles de niveau transactionnel (TLM) :
+ Simple à modéliser, à développer
+ Rapide en simulation, fonctionnellement
- Plus ou moins synthétisable
+/- Abstrait en temps et donnée

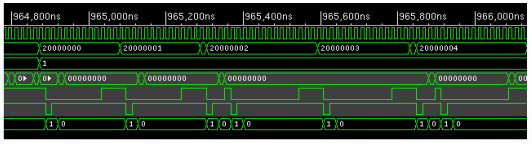
- conso : nok



Objectif : Approche par séparation des préoccupations Power et Fonctionnel de modélisation TLM d'un système. Vérification cohérence comportements Power/Fonctionnels/Thermiques

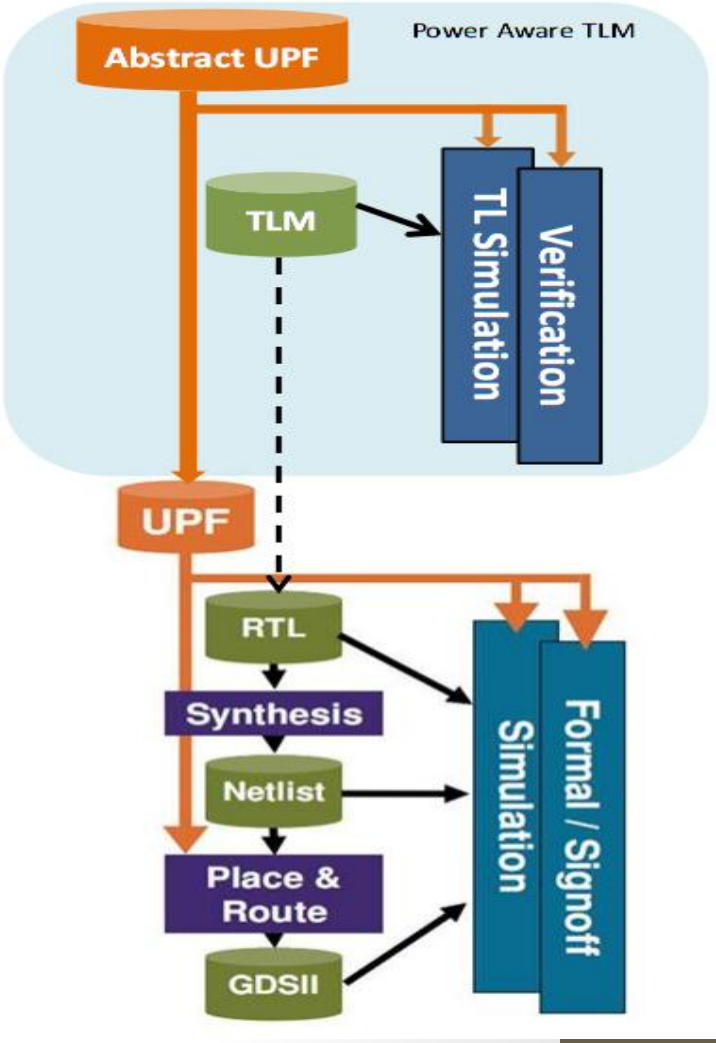
Modèle niveau transfert de registre (RTL) :
Précis en temps et données +
Synthétisable +
TRES lent en simulation -

+ conso : ok



Objectif : étendre le flot de conception « power » vers le niveau transactionnel

- Les abstractions proposées dans SystemC-TLM permettent des vitesses de simulation élevées (vs RTL), mais avec une précision moindre (vs RTL ou CABA).
- UPF (ou CPF) décrit un power intent aux niveaux RTL et en dessous.
  - Séparation des descriptions fonctionnelles et power
- But appliquer une approche similaire au niveau TLM
- Mais UPF (comme CPF) ne contient pas de mécanismes pour décrire un clock intent.

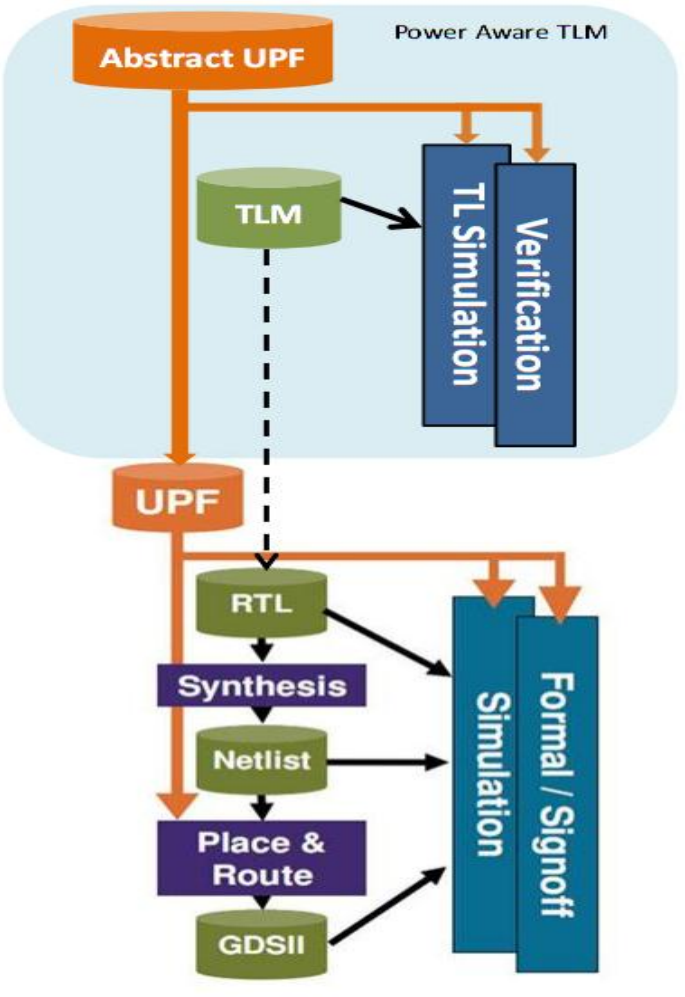


Abstraction d'UPF au niveau TLM

- Les simulateurs niveau TLM ne capturent pas les effets liés à une description d'un power intent de type UPF

- Objectifs :
  - Permettre de valider au niveau TLM un power/clock intent:
    - Cohérence états fonctionnel/power
    - Evaluer la consommation de puissance locale/globale en fonction des états fonctionnels
  - Explorer plusieurs power/clock intent, plusieurs stratégies de power management compatibles avec le fonctionnel
  - Produire une description d'un "golden low power reference design" pour les étapes suivantes de conception RTL.

- Quelles constructions sémantiques relatives au power font du sens au niveau Transactionnel ?

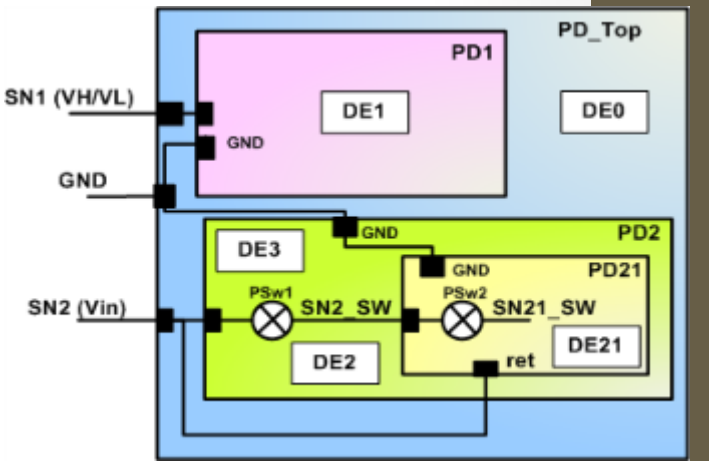


Power/Clock et TLM

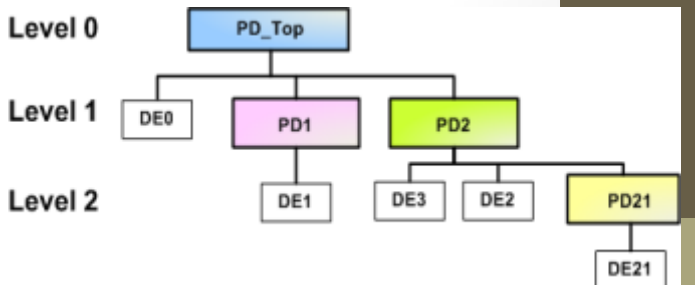
- Concepts faisant du sens au niveau TLM :
  - Power domains, supply nets, power switches
  - UPF : Power state tables, legal power state transitions
  - Clock source, generated clock, clock gating, DVFS

- Règles de composition ajoutées :
  - Les composants (ou Design Elements de UPF) sont attachés à un seul clock domain
  - DVFS implique appartenance des mêmes Design Elements dans power domain et clock domain

- Ces concepts font l'objet de la librairie PwARCH/ClockARCH développée au LEAT
  (Ons Mbarek & Hend Affes, projets ANR HELP et HOPE).

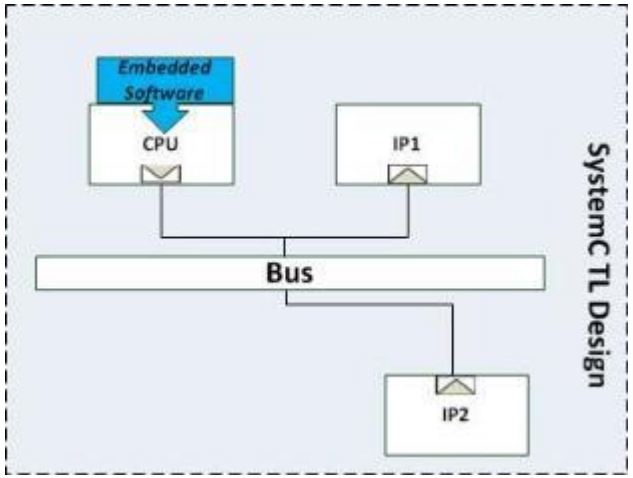


Un exemple de Power Intent



Hiérarchie des power domains correspondante

- Exemple avec une architecture très simple :
  - Prototype virtuel d'une architecture décrit en SystemC-TLM.
  - Logiciels système et applicatif exécutés par le CPU
  - Chaque composant maintient une variable d'état indiquant son état fonctionnel, par exemple :
    - Active_High, Active_Low, Idle, Idle_Retention

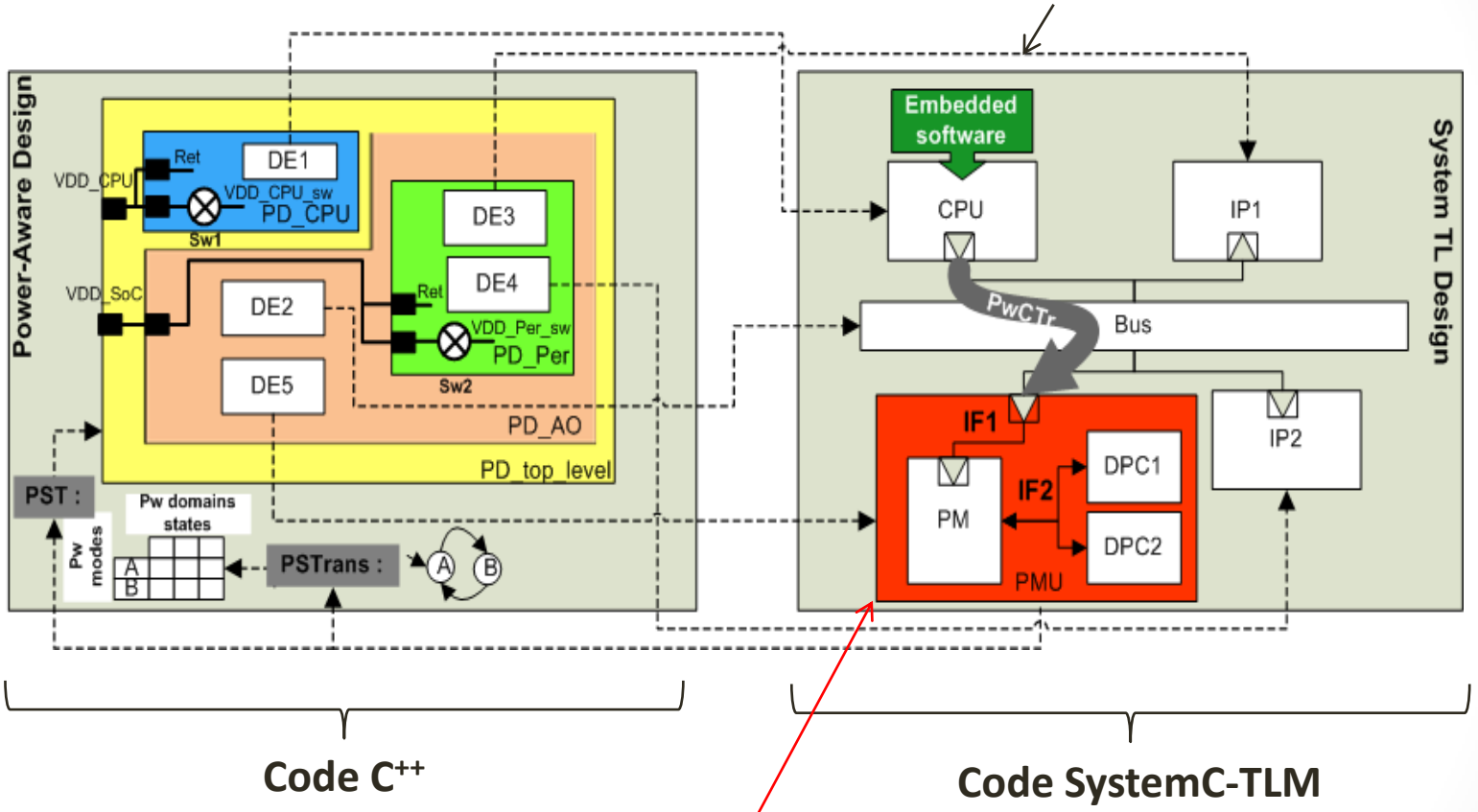


Modèle SystemC-TLM

- Introduction d'un **power intent**

DE : Design Element

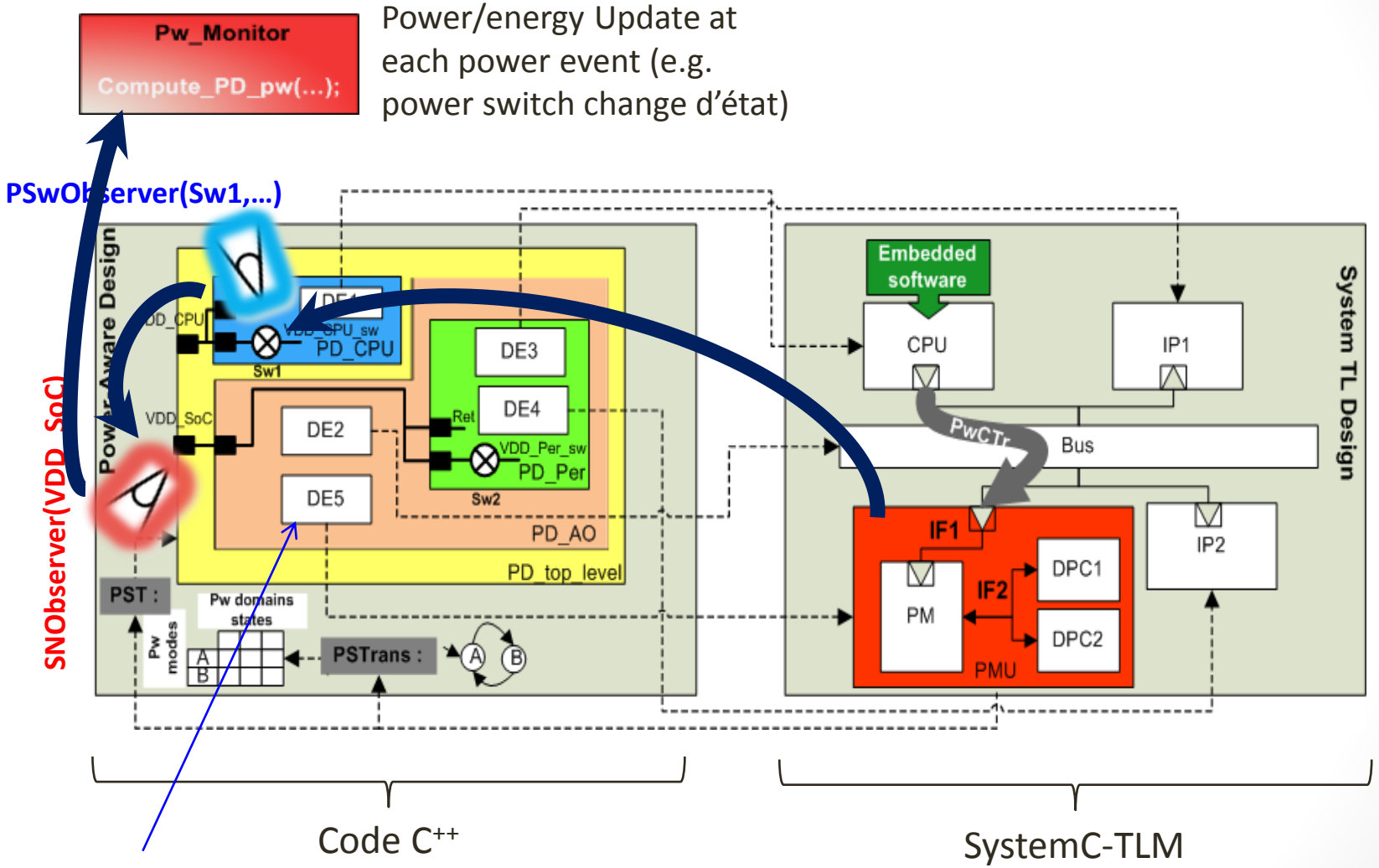
Pointeur vers composant SystemC-TLM



Séparation Power/Fonctionnel

PMU : Power Management Unit

Seul composant SystemC-TLM ajouté

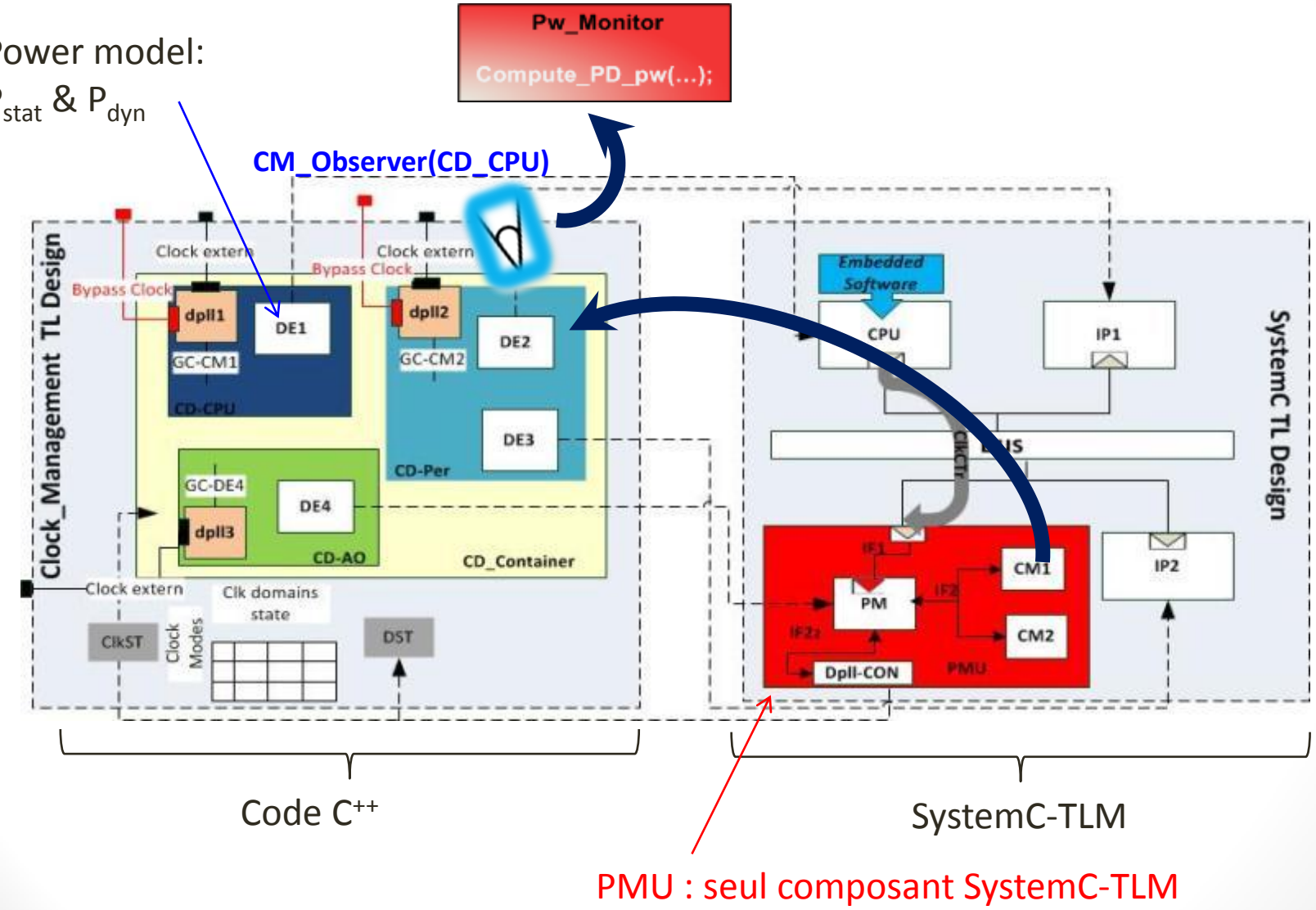


Power model composant:

P_{stat} & P_{dyn}

- Introduction d'un **clock intent**

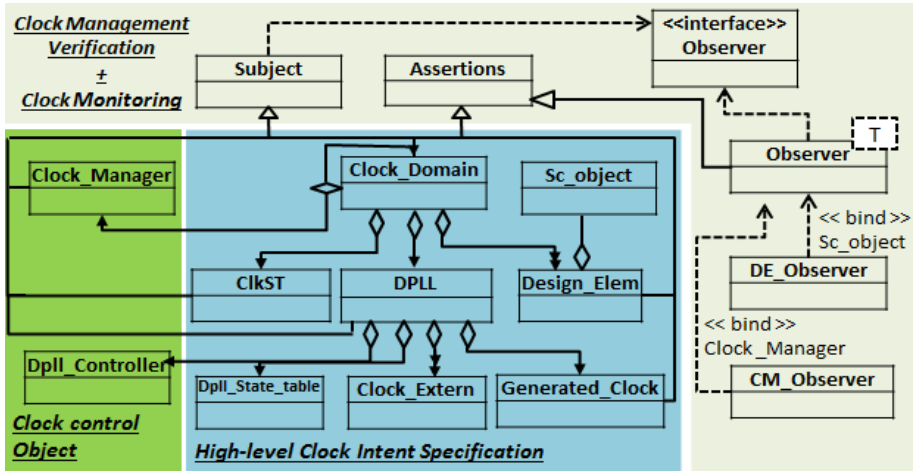
Power model:
P_{stat} & P_{dyn}



- Evénements qui conduisent à une ré-évaluation du power pendant la simulation :
  - Un power switch change d'état
  - Un clock manager change d'état
  - Changement facteur de division d'une DPLL (fréquence de sortie modifiée)
  - La tension sur un supply net est modifiée
  - Un changement d'état fonctionnel d'un composant
    - e.g. Idle -> Active_Low

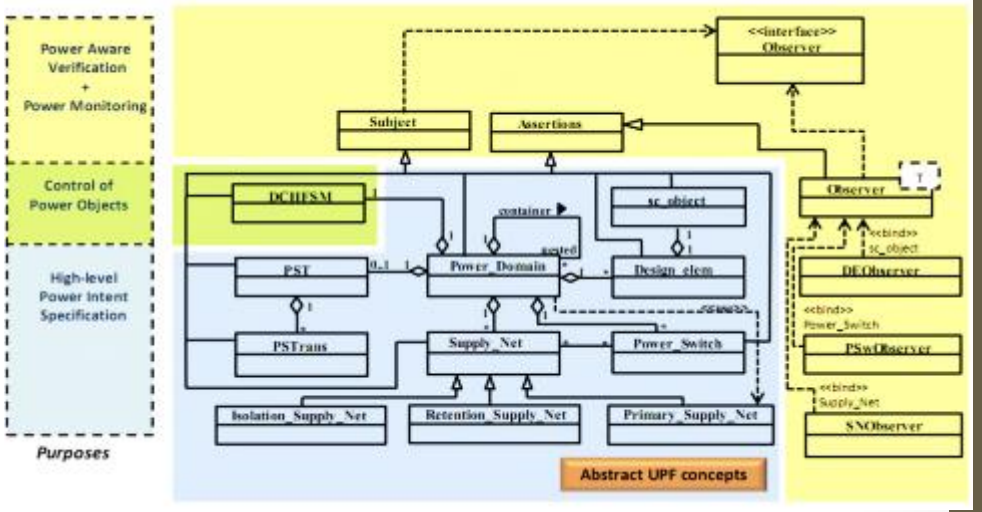
PowerARCH-ClockARCH

Clock Intent Library



SystemC C++

Power Intent Library



Objets de type UPF

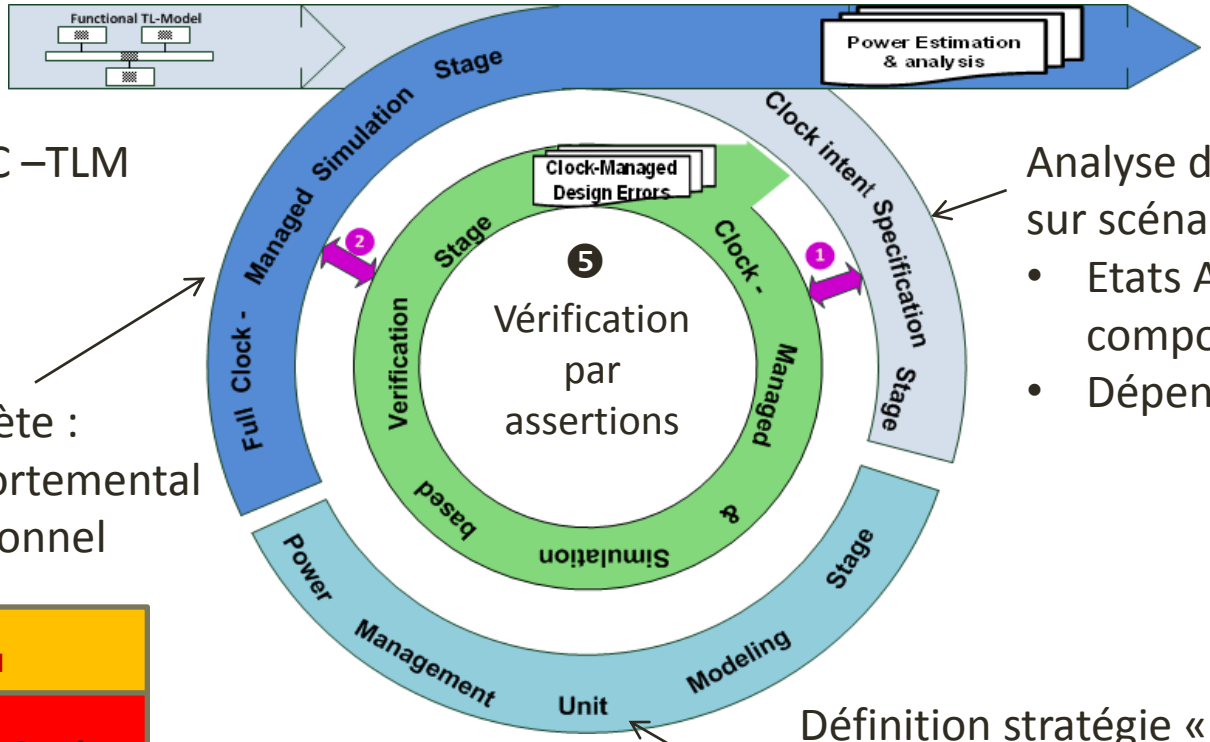
- Les deux librairies C++ & SystemC-TLM définissent toutes les classes utiles à la description d'un power/clock intent.
  - Possibilité de décrire un clock intent indépendemment d'un power intent et réciproquement

Méthodologie

❻ Analyse résultats :
- Performances vs power

❶

Sc_main.cpp



Modèle SystemC –TLM fonctionnel

Analyse des comportements sur scénarios : ❷
- Etats Actif/idle des composants
- Dépendances entre états

Simulation complète :
- Modèle comportemental power + fonctionnel

❹

```
#define PwARCH
#define ClockARCH
#ifdef PwARCH
    PowerMain Section
#endif
```

Sc_main.cpp

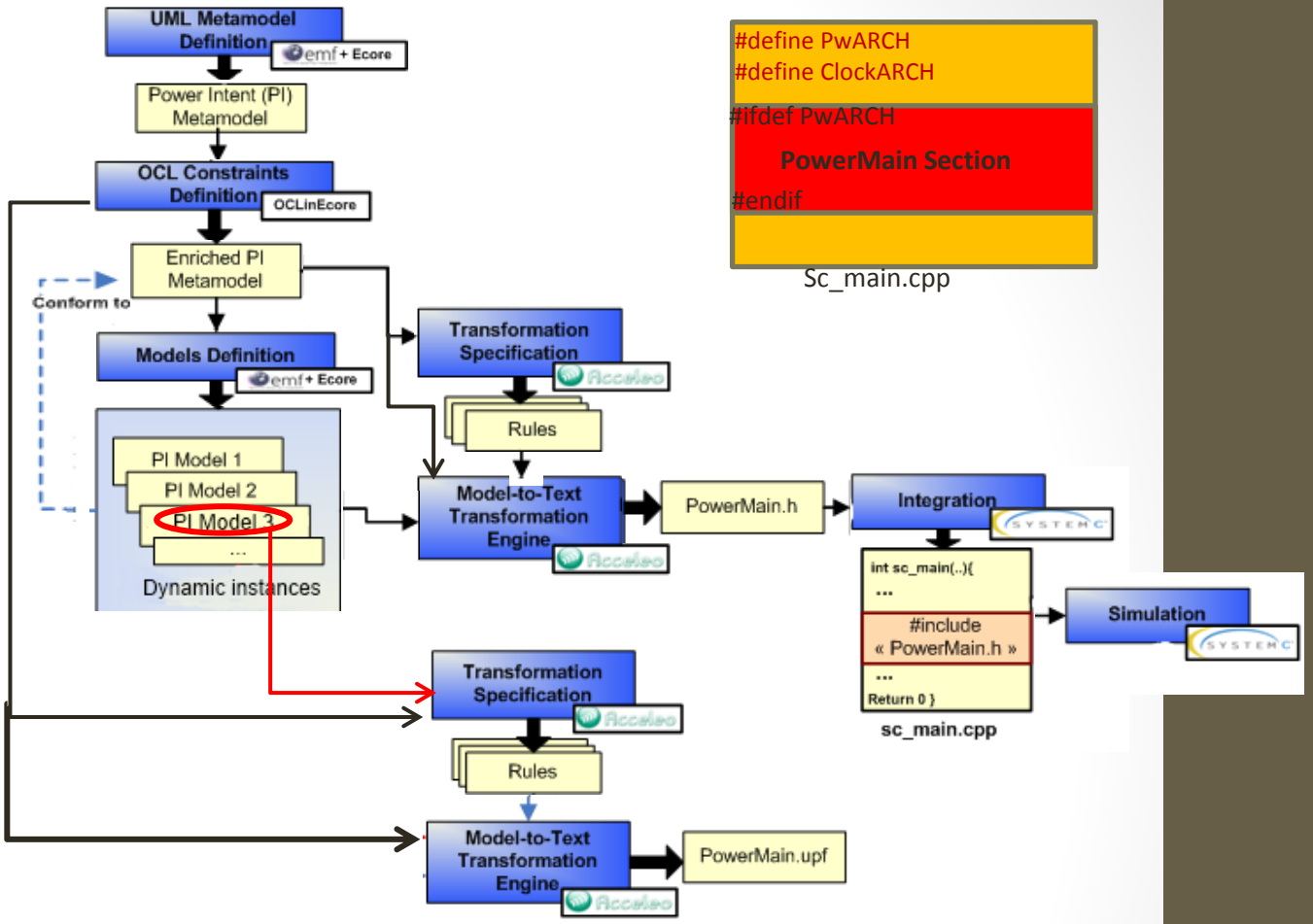
Insertion Power/Clock Intent

Définition stratégie « power » : ❸
- Power/Clock domains
- PMU mettant en œuvre la stratégie

Cela donne ….

```
int sc_main(int, char**)
    /************* Partie fonctionnelle ***************/
    NativeWrapper& cpu = *NativeWrapper::get_instance(); // Instance du CPU
    G726ENC g726enc("G726_encoder"); Bus bus("bus");
    bus.map(g726enc.target, 0x41C40000, 0x00001000); ……..
        /************* Début Partie « power » : création power & clock domain *****/
        Power_Domain* PD_ENC = new Power_Domain(PD_AO,"PD_ENC","top/G711G726ENC");
        Clock_Domain* CLKD_ENC_DEC=new Clock_Domain(CLKD1_top,"CLKD_ENC_DEC", "TOP/CLKD_ENC_DEC");
        /************* Association Design Elements *********/
        Design_elem* DE_G726ENC= new
        Design_elem(PD_ENC,CLKD_ENC_DEC,g726enc,g726enc_capacitance,g726enc_activity,g726enc_leakage);
        /************** Création DPLL et clocks associées*****/
        DPLL* DPLL_ENC_DEC=new DPLL(CLKD_ENC_DEC,"DPLL_ENC_DEC");
        Clock_Externe* Clk_ENC_DEC_R=new Clock_Externe("Clk_ENC_DEC_R","reference",DPLL_ENC_DEC,200);
        /************** Création supply nets et association au power domain**/
        primary_supply_net* VDDENC_SW = new primary_supply_net("VDDENC_SW","SWITCHED",PD_ENC);
        PD_ENC->set_domain_supply_net(VDDENC_SW,GND);
        /************** Création Power switches & Observers ***********/
        Power_Switch* SW1 = new Power_Switch(PD_ENC,"sw1",1);
        Observer<Power_Switch>* Ob_ENC=new Observer<Power_Switch>(SW1,"ob_enc");
        /************** Instanciation du PMU **************/
        PMU pmu("PMU",list_PD,list,clkst,list_dpll,dpll_table);
        bus.map(pmu.PM.target_socket,0x41C80000,0x10); ……..
```

Génération de modèles SystemC-TLM + Power à partir d'une spécification



- Environnement basé ingénierie dirigée par les modèles pour :
  - Génération automatique des déclarations et instanciations des éléments d'un power intent dans un modèle fonctionnel C++ et SystemC-TLM
  - Génération d'une description UPF
- En cours d'extension pour supporter éléments orientés Clock Intent.

Les types de contrats génériques (assertions)

Type	Type des interfaces	Objectif de Verification	Exemples
Type 1 Power/Clock Intent	Interfaces entre éléments power	Power/clock intent est correctement structuré Respect des règles de composition (e.g. hiérarchie)	Création d'un power switch dans un power domain valide DE appartient à plus d'un power/clock domain
Type 2 Etape Définition PM	Interfaces entre PM et Power Controllers	Fonctionnement correct du PM	Power Controller : transitions d'états correctes dans chaque power domain PM : PM scheme/PST-PSTrans
Type 3 En simulation	Unités fonctionnelles - éléments power	Les changements d'états des power et clock domains sont cohérents	Une activité fonctionnelle dans un power (clock) domain alors que le power switch (DPLL ou CM) correspondant est OFF (clock gated).
Type 4 En simulation	Unités fonctionnelles – PM	L'introduction des éléments power n'altèrent pas les comportements fonctionnels attendus	Exécuter une transaction fonctionnelle vers un power domain inactif. La rétention intervient avant un power off.

- Contrats de type Assume/Guarantee inclus dans les classes des deux librairies.
- Permet de révéler des erreurs liées à l'insertion des modèles liés au power

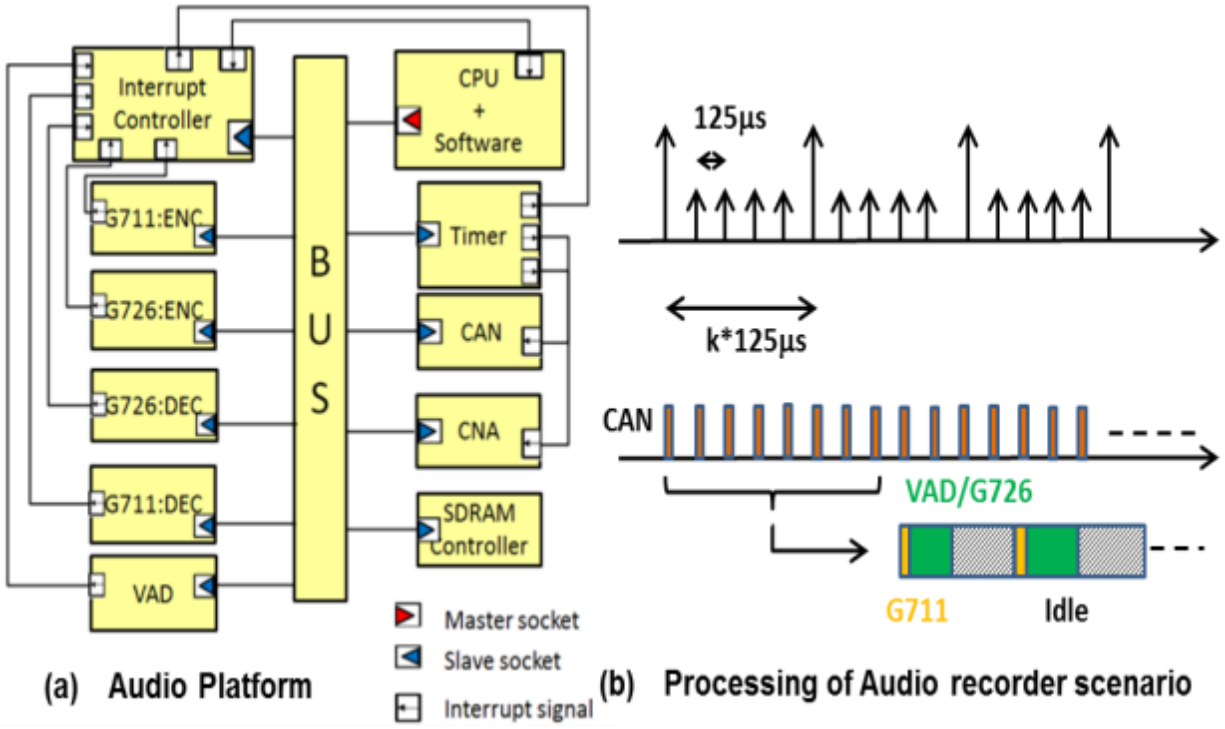
Contenu de l'exposé

- Introduction
- Quelques éléments sur la conception Low Power
- *Power Intent* et *Clock Intent*
- Quelques éléments sur le flot de conception low power
- Modélisation low power au niveau TLM
- Exemple
- Conclusion

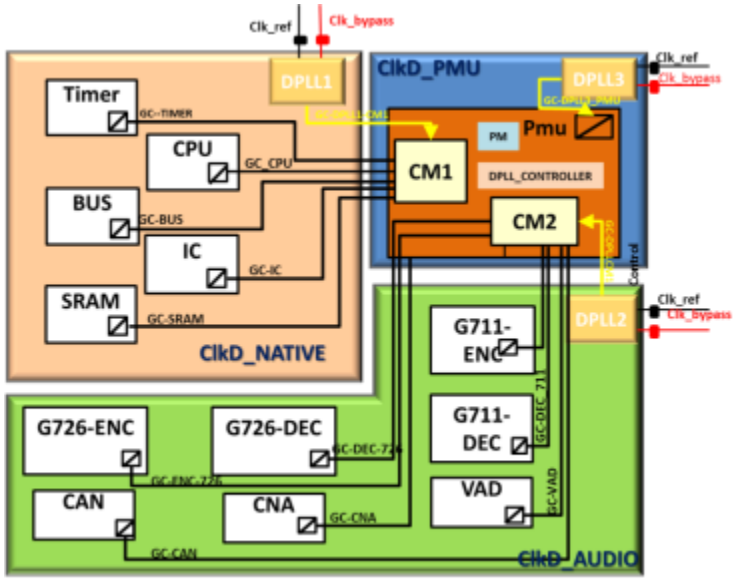
Exemple

- Un système simple de traitement audio :
  - 3 scénarios : Audio Player, Audio Recoder, Audio Player/Recorder
  - G711 : quantification, G726 : compression, VAD : Voice Activity Detection

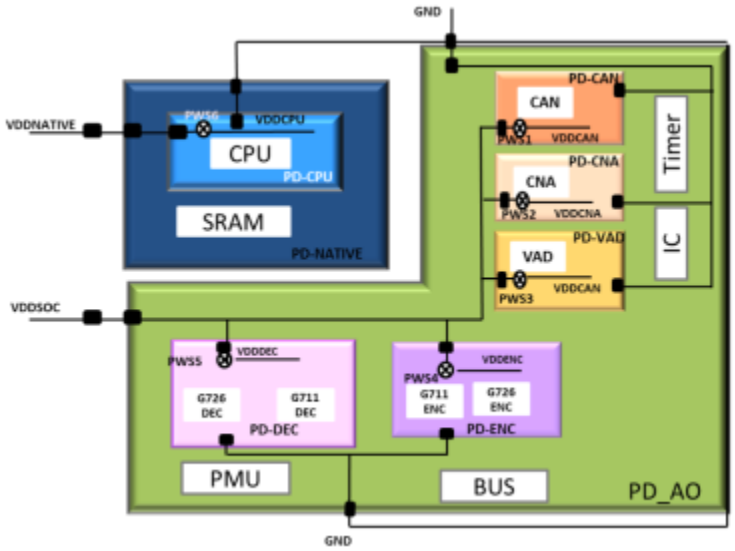
Plateforme SystemC-TLM



- Partitionnement en Clock Domain et Power Domain :



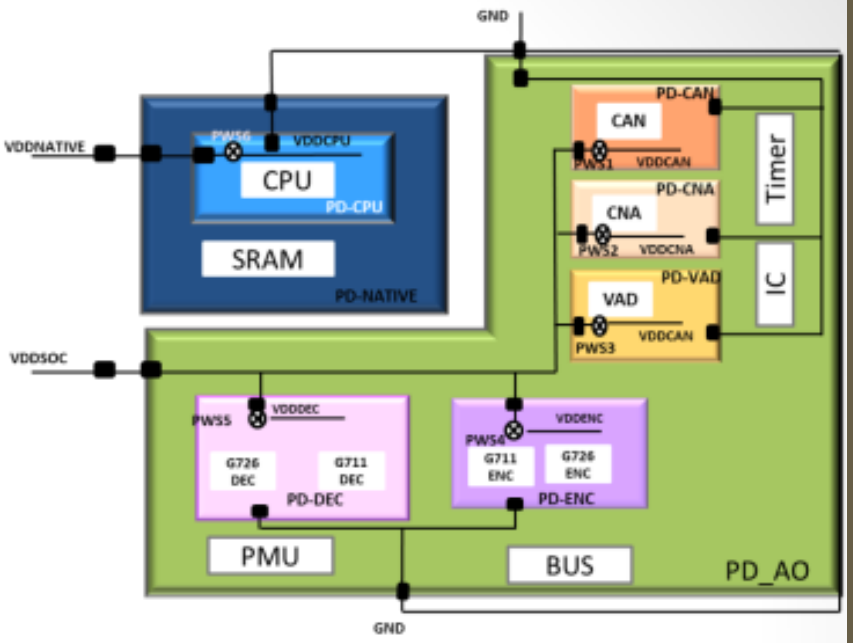
Clock domains

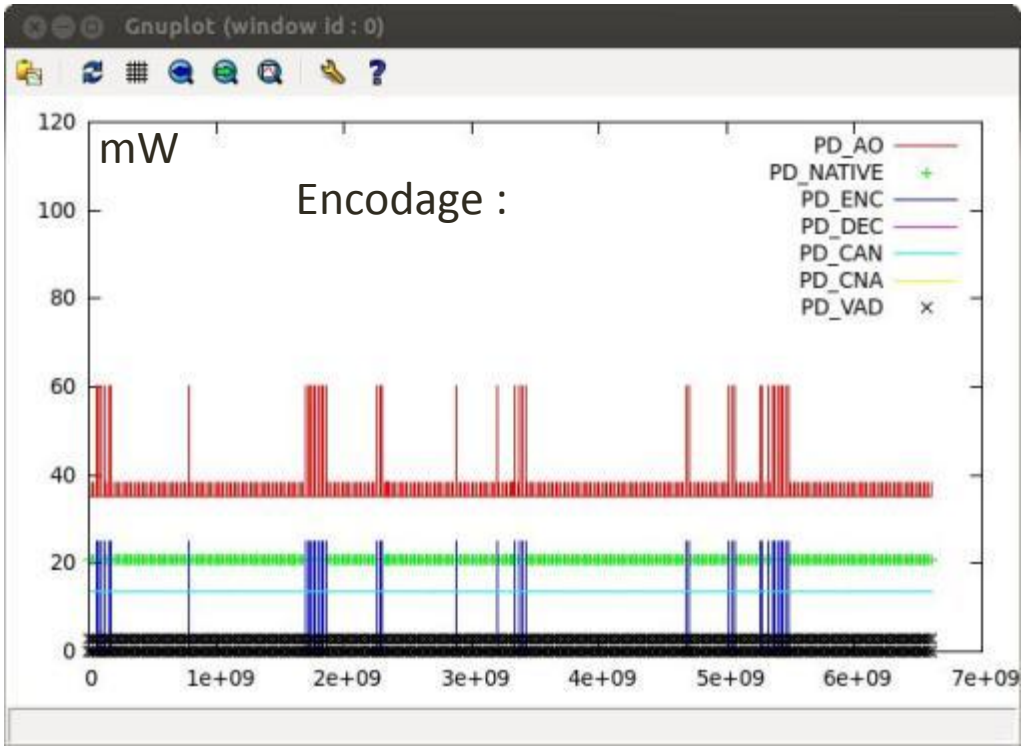


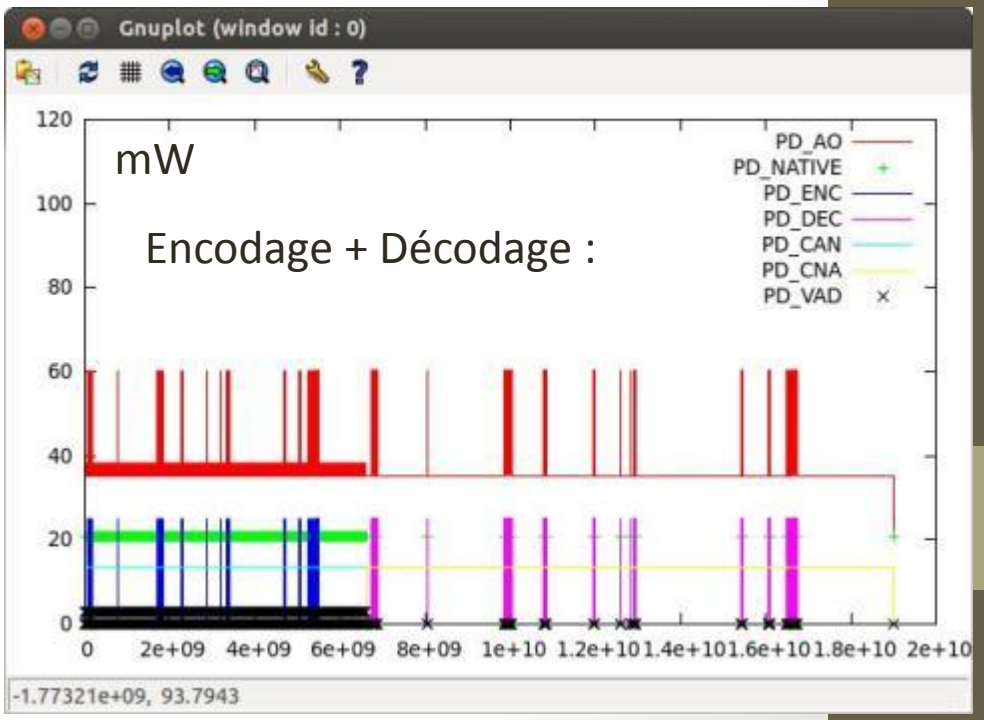
Power domains

D’autres partitionnements sont bien sûr possibles

- Stratégie uniquement axée sur les power domains :

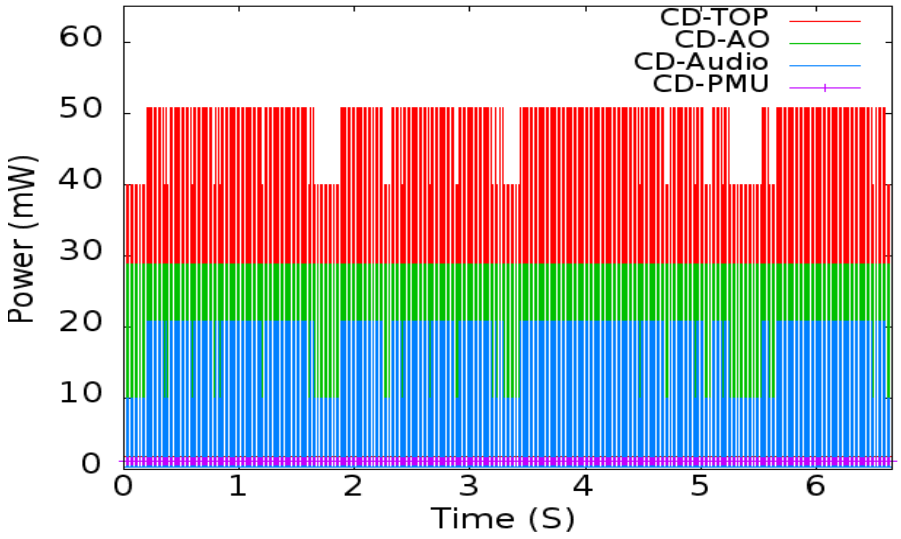


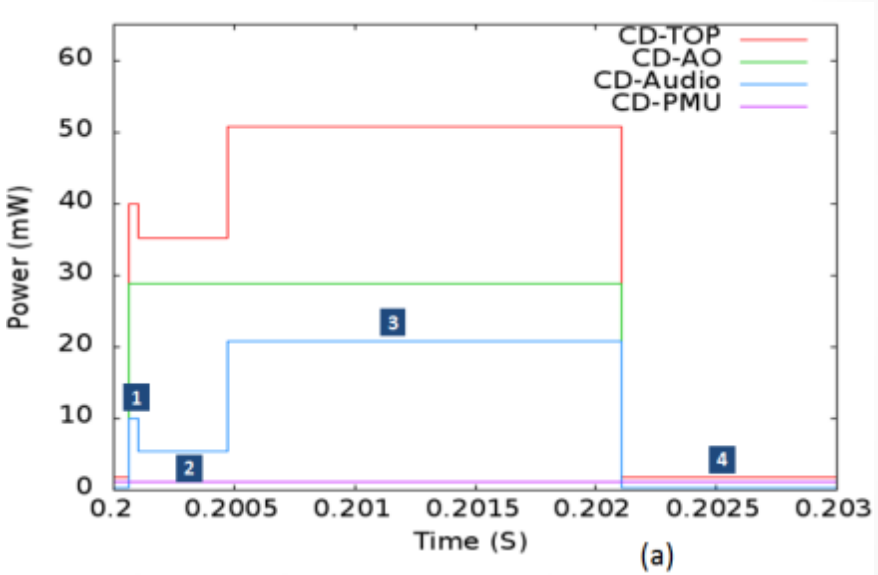




- Puissance consommée par power domain
- Séquence test de 7 secondes de parole avec des silences

- Stratégie uniquement axée sur les clock domains :





(a)

IP \ Step	Frequency (MHZ) 1	2	3	4
VAD	64	0	0	0
G711:ENC	0	32	0	0
G726:ENC	0	0	128	0
CPU	64	64	64	0

(b)

- Clock gating
- Facteurs de division dans les clock managers

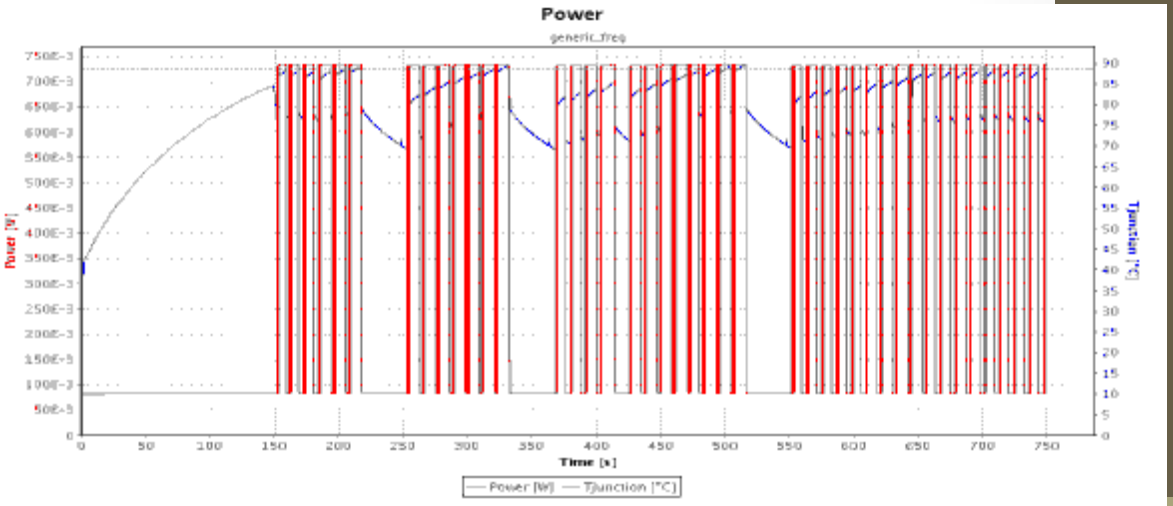
Conclusion

- La consommation de puissance (et le contrôle de la température) est un des premiers critères considérés dans la conception de SoC.
- Introduire une gestion efficace de la consommation d'énergie peut altérer le fonctionnement d'un système
- Cette gestion a un caractère de contrôle global :
  - Couper l'horloge au niveau DPLL : tout l'arbre ne distribue plus d'horloge !
- Valider une stratégie de gestion d'énergie ne peut réellement se faire qu'un niveau du système complet
- Les outils au niveau RTL sont matures (standard UPF) mais peu d'outils commerciaux disponibles au niveau TLM

- Approche TLM de description et vérification (en simulation) de Power/Clock Intent
- Approche par séparation (le plus possible) des préoccupations (fonctionnel/power)
- Permet l'exploration de différentes alternatives
- Vérification de la cohérence entre les comportements fonctionnels et la stratégie de gestion de puissance consommée
  - Evite de décrire des power states à mettre en relation avec des états fonctionnels ou d'insérer du code orienté power dans les modèles fonctionnels des composants.
- Suit le schéma de spécification de power intent développé dans UPF (standard IEEE 1801)
- Vitesse de simulation très peu altérée après ajout des éléments power (moins de 0,1%)

- Co-simulation fonctionnelle – thermique :
  - L'approche proposée modélise les états d'un arbre d'horloge et ceux d'une distribution d'alimentation.
  - A partir de la simulation fonctionnelle on peut connaître les états fonctionnels des composants (idle/actif, taux d'activité,…) et le trafic sur les bus transactionnels

- Des outils comme Aceplorer de Docea Power permettent de développer des modèles thermiques d'un système.
- Possibilité de modéliser l'influence réciproque entre température et puissance statique.
  - Exemple : test de stratégie AVS



Simulation en boucle fermée SystemC-TLM/Aceplorer

T. Sassolas, C. Bechara, P. Vivet et al, DAC 2013

- Quid des technologies avancées telle que FDSOI ?
  - La technologie FDSOI permet d'ajuster dynamiquement la consommation de puissance statique consommée en fonction de la fréquence de fonctionnement souhaitée (par ajustement dynamique de la tension de back-plane).
    - A fréquence faible, la puissance statique peut être divisée par un facteur important.
  - La stratégie de power management doit intégrer ce nouveau paramètre
  - Les modèles de calcul de puissance doivent évoluer en ce sens
  - La prise en compte de ce paramètre complémentaire milite encore plus pour disposer de méthodes/modèles pour explorer des solutions

Quelques références

[1] Nagu Dhanwada, Reinaldo A. Bergamaschi, William W. Dungan, Indira Nair, Paul Gramann, William E. Dougherty, and Ing-Chao Lin , Transaction-level modeling for architectural and power analysis of PowerPC and CoreConnect-based systems,. Design Automation for Embedded Systems, Vol. 10, No. 2-3. (September 2005), pp. 105-125.

[2] Lee, I., Kim, H., Yang, P., Yoo, S., Chung, E.-Y., Choi, K.-M., Kong, J.-T., Eo, S.-K.: PowerViP: Soc Power Estimation Framework at Transaction Level. In: 11th Asia and South Pacific Design Automation Conference (ASP-DAC), Japan, pp. 551–558 (2006)

[3] Ben Atitallah, R., Niar, S., Dekeyser, J.L.: MPSOC Power Estimation Framework at Transaction Level Modeling. In: 19th International Conference on Microelectronics (ICM), Egypt, pp. 245–248 (2007)

[4] Lebreton, H., Vivet, P.: Power Modeling in SystemC at Transaction Level, Application to a DVFS Architecture. In: Proc. of the 2008 IEEE Computer Society Annual Symposium on VLSI, France, pp. 463–466 (2008)

[5] Trabelsi, C., Ben Atitallah, R., Meftali, S., Dekeyser, J.-L., and Jemai, A. A model-driven approach for hybrid power estimation in embedded systems designs. In EURASIP Journal on Embedded Systems (2011).

[6] Daniel Chillet, Eric Senn, Olivier Zendra, Smail Niar, Cécile Belleudy, Victor Tissier, Christian Samoyeau, Open-PEOPLE ANR Project, Open Power and Energy Optimization Platform and Estimator", HIPEAC Newsletter No 24, October 2010

[7] Martin Streubuhr, Rafael Rosales, Ralph Hasholzner, Christian Haubelt, and Jurgen Teich, ESL power and performance estimation for heterogeneous MPSOCS using SystemC, Forum on specification and Design Languages (FDL), Oldenburg, Germany, September 13-15, 2011

[8] Feng Liu, QingPing Tan, Xiaoyu Song, and Naeem Abbasi, AOP-based High-level Power Estimation in SystemC,“ in Proceedings of the 20th ACM Great Lakes Symposium on VLSI. 2010, ACM

[9] Nikhil Bansal, Kanishka Lahiri, Anand Raghunathan, Srimat T. Chakradhar,, Power Monitors: A Framework for System-Level Power Estimation Using Heterogeneous Power Models, in Proceedings of the International Conference on VLSI Design, Los Alamitos, CA,USA, 2005, pp. 579{585

[10] O. Mbarek, A. Khecharem, A. Pegatoquet, M. Auguin, Using Model Driven Engineering to Reliably Accelerate Early Low Power Intent Exploration for a System-on-Chip Design, 27ème Symposium On Applied Computing, Riva del Garda (Trento), Italy, March 26-30, 2012

[11] O. Mbarek, A. Pegatoquet, M. Auguin, A Methodology for Power-Aware Transaction-Level Models of Systems-on-Chip Using UPF Standard Concepts, PATMOS’2011, Madrid, Spain, September 26 - 29, 2011

[12] Ons Mbarek, Alain Pegatoquet, Michel Auguin, Using UPF Standard Concepts for Power-Aware Design and Verification of Systems-on-Chip at Transaction-Level, IET Circuits, Devices & Systems.

[13] DJ Greaves et MM Yasin, “TLM POWER3: Power Estimation Methodology for SystemC TLM 2.0”, FDL,September 2012

[14]T. Bouhadiba et al., “System-Level Modeling of Energy in TLM for Early Validation of Power and Thermal Management”, DATE, 2013

[15] F. Mischkalla and W. Mueller, “Efficient power intent validation using loosely-timed simulation models”, in PATMOS, 2013.

[16] F. Mischkalla and W. Mueller, “Architectural Low-Power Design Using Transaction-Based System Modeling And Simulation”, in Int. Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS), 2014