Exploration des techniques de fouille de données pour un monitoring efficace des systèmes intégrés sur puce

Mohamad Najem

Collège Doctoral du Languedoc-Roussillon

THÈSE

Pour obtenir le grade de

Docteur

Délivré par **l'Université de Montpellier**

Préparée au sein de l'école doctorale **I2S**
Et de l'unité de recherche **LIRMM**

Spécialité : **Systèmes Automatiques et Microélectroniques**

Présentée par **Mohamad Najem**

Exploration des techniques de fouille de données pour un monitoring efficace des systèmes intégrés sur puce

Soutenance le 08 Décembre 2015 devant le jury composé de

M. Francois PÊCHEUX	Pr	LIP6, UPMC	Rapporteur
M. Alain PEGATOQUET	MCF, HDR	LEAT, UNSA	Rapporteur
M. Christophe JEGO	Pr	IMS/ENSEIRB-MATMECA	Examinateur
Mme. Anne LAURENT	Pr	LIRMM, UM	Examinateur
M. Gilles SASSATELLI	DR CNRS	LIRMM, UM	Directeur
M. Pascal BENOIT	MCf, HDR	LIRMM, UM	Co-encadrant

UNIVERSITÉ MONTPELLIER

REMERCIEMENTS

Soyons reconnaissants aux personnes qui nous donnent
du bonheur ; elles sont les charmants jardiniers
par qui nos âmes sont fleuries
Marcel Proust

Le seul moyen de se délivrer d'une tentation, c'est d'y céder paraît-il ! Alors j'y cède en disant en grand Merci aux personnes qui ont cru en moi et qui m'ont permis d'arriver au bout de cette thèse.

En premier lieu, je tiens à remercier mon co-encadrant de thèse, monsieur *Pascal Benoit*, pour la confiance qu'il m'a accordée en acceptant d'encadrer ce travail doctoral, pour ses multiples conseils et pour toutes les heures qu'il a consacrées à diriger cette recherche. J'aimerais également lui dire à quel point j'ai apprécié sa grande disponibilité et son respect sans faille des délais serrés de relecture des documents que je lui ai adressés. Enfin, j'ai été extrêmement sensible à ses qualités humaines d'écoute et de compréhension tout au long de ce travail doctoral. Je tiens à exprimer aussi mes plus vifs remerciements à *Gilles Sassatelli* qui fut pour moi un directeur de thèse attentif et disponible malgré ses nombreuses charges. Sa compétence, sa rigueur scientifique et sa clairvoyance m'ont beaucoup appris. Il a toujours été là pour me soutenir et me conseiller au cours de l'élaboration de cette thèse.

Je sais infiniment gré à monsieur *François Pechêux* de s'être rendu disponible pour la soutenance et d'avoir accepté la fonction de rapporteur. De même, je suis particulièrement reconnaissante à monsieur *Alain Pegatoquet* de l'intérêt qu'il a manifesté à l'égard de cette recherche en s'engageant à être rapporteur.

Monsieur *Christophe Jego* et Madame *Anne Laurent* m'ont fait l'honneur d'être examinateurs de ma thèse, ils ont pris le temps de m'écouter et de discuter avec moi. Leurs remarques m'ont permis d'envisager mon travail sous un autre angle. Pour tout cela je les remercie.

Au cours de ces années j'ai fait parti de l'équipe ADAC « ADAptive Computing group » au sein du laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier (LIRMM). Les discussions que j'ai pu avoir durant les réunions d'équipe ou en dehors avec Monsieur

Michel Robert et Monsieur *Lionel Torres* m'ont beaucoup apporté. Je remercie donc tous les membres de l'équipe ADAC.

Durant ma thèse j'ai aussi effectué de nombreux enseignements à l'école Polytechnique de Montpellier et je remercie de nouveau *Pascal Benoit, Mariane Comte, Arnaud Virazel* et *Bruno Rouzeyre* pour leur aide.

Ayant mon bureau au sein du LIRMM, je remercie *Mohamad El Ahmad, Sophiane Senni, Anu Asokan, Imran Wali* qui m'ont entouré et m'ont conseillé, ainsi que tous les thésards pour m'avoir supporté tous les jours depuis plusieurs années : *Syhem Larguesh, Aymen Touati, Alejandro Nocua.* Ils m'ont beaucoup aidé et sont devenus des amis à qui je souhaite tout le courage qu'ils m'ont apporté.

Je remercie toutes les personnes avec qui j'ai partagé ces années de thèse : *Hussein Awada* et *Mohamad El Ahmad* pour leurs supports moraux et leurs soutiens, *Mahmoud Fakih* pour les supports logistiques et les beaux moments.

Un grand merci à *Romain Boissard* pour avoir joué le rôle de relecteur du document final.

Heureusement que mes parents, mes frères et mes amis sont la pour me changer les idées. Ils ont tous cru en moi et ouf ! maintenant j'y suis ! J'adresse donc toute mon affection à ma famille, et en particulier à mon père et ma mère. Malgré mon éloignement depuis de (trop) nombreuses années, leur intelligence, leur confiance, leur tendresse, leur amour me portent et me guident tous les jours. Merci pour avoir fait de moi ce que je suis aujourd'hui.

Enfin, les mots les plus simples étant les plus forts, j'adresse mille mercis pour ma fiancée *Sandra* qui a su me soutenir, me supporter et m'encourager pendant toute la durée de ma thèse et plus particulièrement durant les derniers mois de rédaction qui n'ont pas toujours été des plus agréables. Merci pour ton amour qui m'a été essentiel durant ces années, pour ton sacrifice et de m'avoir attendue avec tant de patience. Cette thèse et moi te devons beaucoup. Merci ! !

J'en oublie certainement encore et je m'en excuse.

Je dédie cette thèse à mon amour
Sandra Soueid...

RÉSUMÉ

La miniaturisation des technologies de semi-conducteurs a permis en quelques décennies de concevoir des systèmes toujours plus complexes, comprenant aujourd'hui plusieurs milliards de transistors sur un même substrat de silicium. Cette augmentation des densités d'intégration fait face à une contrainte physique représentée par la quantité de puissance consommée par unité de surface. À cela s'ajoutent également des problèmes de fiabilité, en raison notamment des hotspots, qui peuvent accélérer la dégradation des transistors et réduire en conséquence la durée de vie du composant. L'efficacité énergétique des circuits devient un enjeu majeur, aussi bien dans le domaine de l'embarqué que pour des applications de calcul haute performance. La prise en compte de ces contraintes nécessite la mise en place de solutions nouvelles, s'appuyant notamment sur des techniques d'auto-adaptation. Celles-ci reposent généralement sur un processus bouclé en trois phases : (*i*) le monitoring qui consiste à observer l'état du système, (*ii*) le diagnostic qui analyse les informations relevées pour optimiser le fonctionnement du système, et (*iii*) l'action qui règle les paramètres en conséquence. L'efficacité d'une méthode d'adaptation dépend non seulement de l'algorithme d'optimisation mais aussi de la précision de l'information observée en ligne. Le monitoring est généralement effectué à l'aide d'un ensemble de capteurs intégrés (analogiques ou numériques). Les méthodes industrielles actuelles sont généralement très coûteuses et nécessitent l'insertion d'un grand nombre d'unités pour avoir une information précise sur le comportement du système à une résolution spatiale et temporelle fine. Cette thèse propose une approche innovante qui intervient en amont ; un ensemble de techniques issues du domaine de la fouille de données est mis en œuvre pour l'analyse de données extraites des différents niveaux d'abstractions à partir du flot de conception, ce afin de définir une solution optimale en terme de coût et de précision. Notre méthode permet de dégager de manière systématique l'information pertinente requise pour la mise en œuvre d'un monitoring efficace et dans un contexte où la consommation et la fiabilité apparaissent comme de fortes contraintes, cette thèse s'intéresse plus particulièrement à celui de la puissance et de la température sur puce.

ABSTRACT

Over the last decade, the miniaturization of semiconductor technologies has given rise to design complex systems, including today's several billions of transistors on a single die. As a consequence, the integration density has increased and the power consumption has become significant. This is compounded by the reliability issues represented by the presence of thermal hotspots that can accelerate the degradation of the transistors, and consequently reduce the chip lifetime. In order to face these challenges, new solutions are required, based in particular on self-adaptive systems. These systems are mainly composed of a control loop with three processes : (*i*) the monitoring which is responsible for observing the state of the system, (*ii*) the diagnosis, which analyzes the information collected and make decisions to optimize the behavior of the system, and (*iii*) the action that adjusts the system parameters accordingly. However, effective adaptations depend critically on the monitoring process that should provide an accurate estimation about the system state in a cost-effective way. The monitoring is typically done by using integrated sensors (analog or digital). The industrial methods are usually very expensive, and require a large number of units to produce precise information at a fine-grained resolution. This thesis proposes an innovative and "upstream" approach ; a set of data mining techniques is used to analyze data extracted from various levels of abstractions from the design flow, in order to define the optimum monitoring in terms of cost and accuracy. Our method systematically identifies relevant information required for the implementation of effective monitoring. This thesis mainly focuses on the monitoring of the power and the temperature of the chip.

LISTE DES PUBLICATIONS

Journaux Internationaux

- **Mohamad Najem**, Pascal Benoit, Mohamad El Ahmad, Gilles Sassatelli, Lionel Torres, "*Lightweight Monitoring of the Power Consumption : Data Mining to the Rescue*", IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems (soumis).

Conférences Internationales

- **Mohamad Najem**, Pascal Benoit, Florent Bruguier, Gilles Sassatelli, Lionel Torres, "*Method for Dynamic Power Monitoring on FPGAs*", FPL'2014 : Field Programmable Logic and Applications (2014), Munich, Allemagne.
- **Mohamad Najem**, Pascal Benoit, Gilles Sassatelli, Lionel Torres, "*Cost-Effective System-Level Power-Modeling based on Performance Events*", ARCS'2016 : Architecture of Computing Systems (2016), Nuremberg, Allemagne (soumis).
- Mohamad El Ahmad, **Mohamad Najem**, Pascal Benoit, Gilles Sassatelli, Lionel Torres, "*Adaptive Power Monitoring For Self-Aware Embedded Systems*", NORCAS'2015 : IEEE Nordic Circuits and Systems Conference, 26-28 Octobre 2015, Oslo, Norway.

Worshops

- **Mohamad Najem**, Pascal Benoit, Gilles Sassatelli, "*Exploration of Data Mining techniques for the Monitoring of Systems on Chips*", Journée scientifique pour les chercheurs LIRMM (2015), Montpellier, France.
- **Mohamad Najem**, Pascal Benoit, Gilles Sassatelli, Lionel Torres, "*Monitoring in Embedded Systems*", GDR SoCSiP'2014 : Groupe de recherche pour la communauté SoC-SiP (2014), Paris, France.

- Mohamad El Ahmad, **Mohamad Najem**, Pascal Benoit, Gilles Sassatelli, "*Data Mining for On-Chip Monitoring*", workshops avec INRIA-LEAT (2015), Montpellier, France.

TABLE DES MATIÈRES

LISTE DES ACRONYMES

SoC "*System on Chip*"
PDA "*Personal Digital Assistant*"
ITRS "*International Technology Roadmap for Semiconductors*"
PMU "*Performance Monitoring Unit*"
DVFS "*Dynamic Voltage and Frequency Scaling*"
MPSoC "*Multi-Processor System on Chip*"
NoC "*Network on Chip*"
PLL "*Phase Locked-Loop*"
FLL "*Frequency Locked-Loop*"
VFI "*Voltage and Frequency Island*"
IEM "*Intelligent Energy Manager*"
API "*Application Programming Interface*"
CPM "*Critical Path Monitor*"
PMIC "*Power Management Integrated Circuit*"
EC "*Events Counter*"
RTL "*Register Transfer Level*"
HPC "*High Performance Computing*"
CFS "*Correlation Feature Selection*"
CSE "*Classifier Subset Evaluation*"
MARS "*Multivariate adaptive regression splines*"
NN "*Neural Network*"
LM "*Linear Model*"
TAPoM "*Toggling Activity based Power Monitoring*"
PCC "*Power estimation and Counters Controller*"
PCM "*Performance Counter Monitor*"
ACP "*Analyse en Composantes Principales*"
PCA "*Principal Component Analysis*"
PESel "*Performance Events Selection*"

LISTE DES FIGURES

LISTE DES TABLEAUX

CHAPITRE 1

INTRODUCTION

« Chaque bonne réalisation, grande ou petite, connait ses périodes de corvées et de triomphes ; un début, un combat et une victoire »

Mahatma Gandhi

Sommaire

1.1 Systèmes Embarqués : Contexte et Motivations

Les dernières décennies furent les témoins d'une avancée spectaculaire du domaine des systèmes embarqués et plus particulièrement les systèmes sur puce "*System on Chip*" (SoC). Un ordinateur était jadis une masse encombrante de tubes électroniques et extrêmement complexe à mettre en œuvre pour de simples opérations natives. Un saut dans le temps, et on est passé à des ordinateurs de poche "*Personal Digital Assistant*" (PDA) plus complexes et encore plus performants, capables de gérer un système informatique complet. Cette évolution déterministe est gouvernée par la loi de Gordon Earle Moore annoncée dans les années soixante qui prévoyait le doublement du nombre de transistors à intégrer tous les deux ans. La loi de Moore est toujours d'actualité, les circuits sont de plus en plus petits, avec une densité d'intégration comparable à celle du cerveau humain. La révolution des systèmes embarqués ne s'arrête pas là. Les développements actuels de l'électronique sont souvent présentés suivant deux directions complémentaires (voir Figure 1.1) :

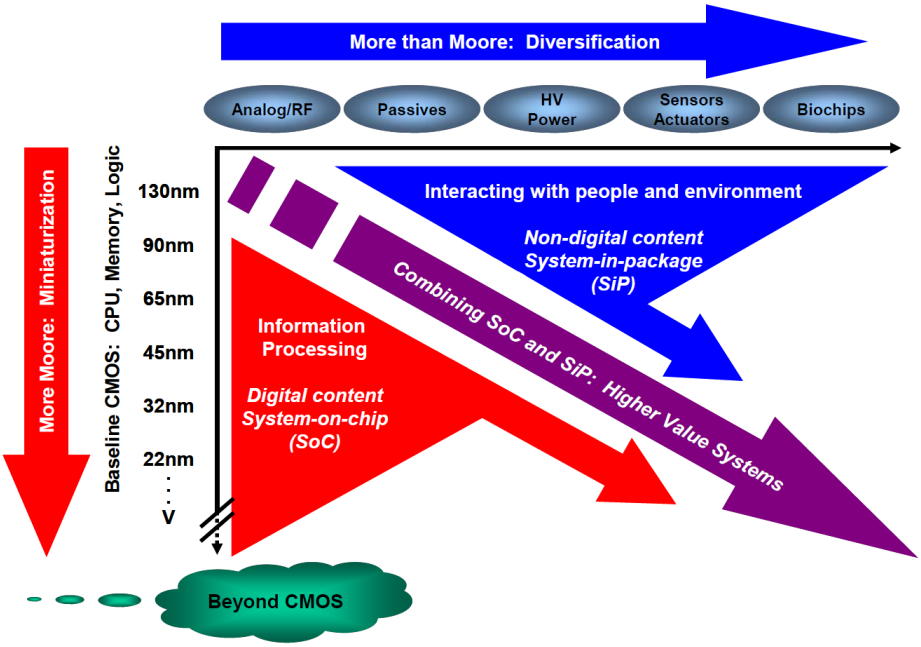


FIGURE 1.1 – La prévision de l'évolution des systèmes embarqués suivants les deux voies "More Moore" et "More than Moore" [ITR13].

- La voie de la miniaturisation "More Moore", qui consiste à continuellement décroitre les dimensions des composants élémentaires (les transistors) ; cette tendance se rapporte, pour une époque donnée, à la finesse de gravure des transistors, ou dimension minimale. Actuellement, l'industrie atteint une finesse de 14 nanomètres.
- La voie de diversification "More than Moore", qui définit la tendance à intégrer de plus en plus de technologie différentes dans une même puce, indépendamment du degré de miniaturisation.

Cependant, cet accroissement de la complexité ne s'est pas fait sans faire apparaitre de nouvelles problématiques. Tout d'abord, la consommation des systèmes ne cesse de doubler tous les 4 ans d'après les études de l'ITRS [ITR13]. D'un autre côté, la technologie des batteries n'est pas capable de suivre le rythme d'évolution des architectures actuelles. De là, est venue la nécessité d'assurer une bonne gestion des ressources qui adapte l'architecture au juste besoin de l'application pour économiser sa consommation (un facteur aussi important que les performances). Ces mécanismes d'adaptation doivent opérer dynamiquement afin de bien manier l'hétérogénéité des applications non maîtrisées au moment de conception du circuit.

En outre, la miniaturisation continue des transistors a provoqué l'apparition de nouvelles problématiques d'une autre nature. La tenue dans le temps d'un circuit est directement liée aux conditions de fonctionnement, en particulier la chaleur dont l'impact s'accentue avec la densité d'intégration. Même dans des conditions d'utilisation dites normales (température ambiante, absence de bruit sur les lignes d'alimentation ou IR drop ...), plusieurs travaux ont révélé un phénomène de vieillissement des transistors qui entraine une dégradation de leur performance. Ainsi, un certain nombre de paramètres tels que la fréquence maximale du circuit, la tension de seuil des transistors, sa consommation nominale, sont susceptibles de changer dans le temps. À cela s'ajoutent également des problèmes de fiabilité, en raison notamment de la densité de puissance élevée qui par effet joule contribue à l'augmentation de la température globale de la puce. De plus, les parties de la puce exécutant des opérations intenses risquent de former des points chauds localisés, dont la température extrême peut contribuer à endommager la puce de façon irréversible. Bien entendu, ces anomalies surviendraient après la phase de fabrication du circuit, au cours de son utilisation. Il conviendrait d'introduire des mécanismes de gestion dynamique dont le rôle sera d'optimiser le rendement du circuit à long terme. Une meilleure gestion de température sera cruciale, tout en veillant à garantir une bonne efficacité énergétique ; le circuit dissipe le minimum d'énergie nécessaire pour exécuter l'application en tenant compte de ses limites de performances.

En résumé, toujours dans le but de garantir une utilisation optimale du circuit, il est nécessaire de prévoir des mécanismes dynamiques qui adaptent le circuit à son contexte de fonctionnement, tant sur un plan applicatif avec les multiples standards à gérer, que technologique. Les systèmes auto-adaptatifs dynamiques mettent en œuvre des capteurs et des actionneurs avec une logique de prise de décision, le tout embarqué au sein du circuit. Un synoptique général de ce schéma est donné dans la Figure 1.2. Intégré sous forme de boucle de contrôle fermée, ce schéma d'adaptation est en mesure d'observer l'état du circuit, ensuite de prendre des décisions pour optimiser son fonctionnement à l'aide des actionneurs qui règlent les paramètres en conséquence. Ainsi, l'efficacité de cette boucle d'adaptation repose en premier lieu sur la phase d'observation en ligne (*monitoring*) qui sera chargée de reporter des informations sur l'état du système. Les méthodes industrielles actuelles consistent à placer un nombre de capteurs par ressource (monitoring statique). À titre d'exemple, l'approche classique pour le monitoring

de la puissance consiste à intégrer un capteur de courant sur chaque rail d'alimentation (*e.g.* ARM Big.LITTLE). Ainsi, ce type de monitoring exige la mise en place de techniques d'adaptations à gros grain. Cependant, avec une information spatiale et temporelle plus détaillée de l'état du système, il est possible de mettre en place des méthodes à grain fin plus performantes. Par exemple, une consommation élevée au niveau d'un cluster nécessite une gestion de la fréquence de fonctionnement de tous les cœurs du cluster. Toutefois, si cette information est disponible en ligne à grain plus fin, à l'échelle du cœur, une gestion des blocs qui consomment le plus suffit pour ajuster cet état. Pour cela, les méthodes classiques sont très coûteuses, et généralement ne sont pas en mesure de produire l'information nécessaire pour une adaptation efficace du système.

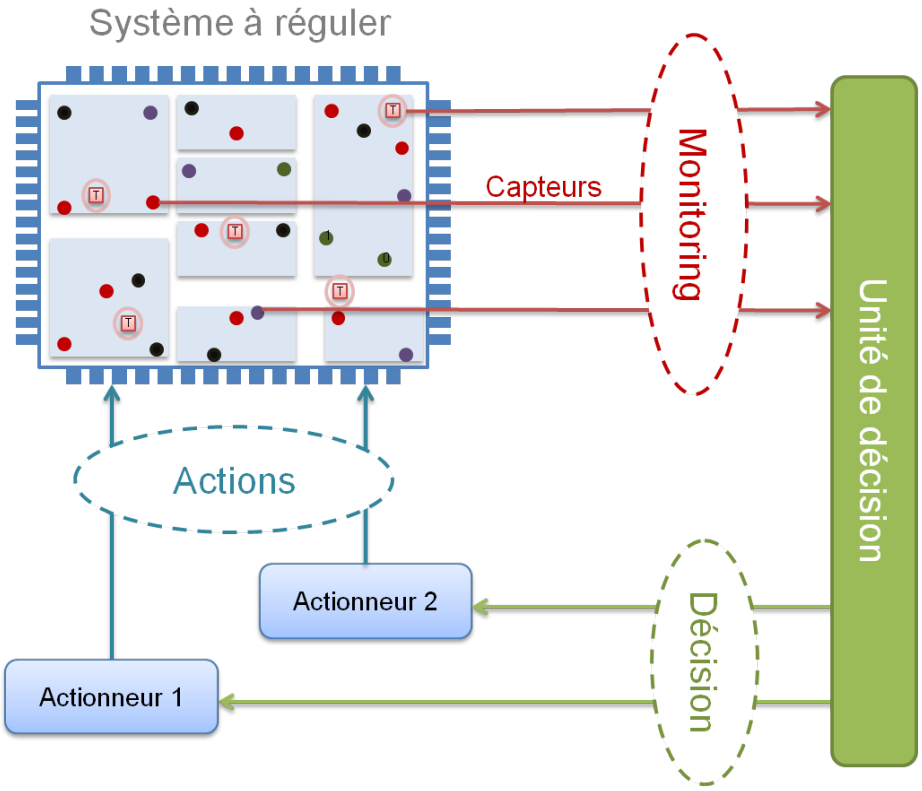


FIGURE 1.2 – Représentation d'une chaine d'adaptation dans un circuit embarqué (adapté de [Bru12]).

1.2 Objectifs et Contributions de la thèse

Dans ce contexte, nous nous somme intéressés au cours de cette thèse au monitoring dans les systèmes intégrés, ce afin de répondre à plusieurs questions en ce sens :

- D'abord, il est important de se demander comment adapter un système pour optimiser son comportement, et a quelle granularité spatiale et temporelle ?

- Il est aussi nécessaire de mener une réflexion sur les méthodes de monitoring dans les systèmes intégrés. Une analyse de l'état de l'art nous permet d'étudier plusieurs solutions mises en place dans ce but. À partir de là, nous pouvons également nous demander quelle est la solution optimale pour observer l'état du système ?
- Le nombre croissant d'unités dans les systèmes complexifie de plus en plus l'instrumentation du flot de conception. Il est ainsi intéressant de se demander comment instrumenter et traiter les grandes quantités de données issues du flot de conception ? comment retrouver l'information pertinente pour un monitoring précis ? et comment identifier l'emplacement optimal des capteurs pour réaliser un monitoring à moindre coût.
- Enfin, nous pouvons également nous demander si on pourra exploiter des ressources existantes dans les processeurs actuels pour réaliser ce monitoring à coût négligeable.

Pour répondre à cette problématique, cette thèse propose une approche innovante qui intervient en amont ; un ensemble de techniques issues du domaine de la fouille de données est mis en œuvre pour l'analyse de données extraites des différents niveaux d'abstractions à partir du flot de conception, ce afin de définir une solution optimale en terme de coût et de précision. Notre méthode permet de dégager de manière systématique l'information pertinente requise pour la mise en œuvre d'un monitoring efficace et dans un contexte où la consommation et la fiabilité apparaissent comme de fortes contraintes, cette thèse s'intéresse plus particulièrement à celui de la puissance et de la température sur puce.

La première contribution (*cf.* Chapitre 3) se concentre au niveau matériel et permet de générer des estimations à grain fin de la puissance. Dans cette optique, nous avons réalisé une instrumentation complète du flot de conception dans le but d'extraire les données à différents niveaux d'abstraction. L'ensemble des méthodes supervisées de la fouille de données sont déployées et comparées dans le but d'identifier l'information pertinente qui corrèle avec la puissance du circuit. Les commutations enregistrées à niveau « transfert de registre » fournissent un indicateur fiable et peuvent être observées facilement en ligne par de simples compteurs d'évènements implémentés sur quelques signaux stratégiques. Un ensemble de modèle est aussi proposé pour l'estimation en ligne de la puissance en fonction de ces évènements.

La plupart des systèmes intégrés disposent de compteurs de performances qui peuvent être exploités efficacement dans le but de réaliser un monitoring de la puissance. La deuxième contribution (*cf.* Chapitre 4) repose sur une optimisation de l'utilisation de ces ressources pour mettre en œuvre une solution à granularité fine. L'analyse des données s'effectue au niveau matériel et logiciel, à l'aide des sondes intégrées (telles que les "*Performance Monitoring Unit*" (PMU)) pour un monitoring hybride à un coût négligeable. Une heuristique inspirée des algorithmes de fouille de données est développée dans ce but, et elle permet de sélectionner automatiquement les évènements ayant le plus d'impact. Des modèles sont proposées afin de suivre les variations de la puissance tenant compte des mécanismes d'adaptation ("*Dynamic*

Voltage and Frequency Scaling" (DVFS)), et sont robustes aux variations de températures.

Une densité de puissance élevée provoque une augmentation de température qui à son tour, augmente la consommation statique, dégrade la fiabilité et accélère le phénomène de vieillissement de la puce. Le monitoring de la température semble donc un élément incontournable dans les systèmes intégrés actuels. La question est de savoir comment déployer un ensemble de capteurs pour pouvoir observer efficacement les variations temporelles et spatiales. Pour cela, nous avons mis en place, dans une troisième contribution (*cf.* Chapitre 5), un ensemble d'outils nous permettant de réaliser des simulations thermiques de nos circuits. Grâce à l'instrumentation des logiciels utilisés, nous pouvons mettre en œuvre des méthodes de segmentation des données afin de définir les régions thermiquement homogènes. À partir de cette information, nous avons proposé une solution de placement des capteurs de température.

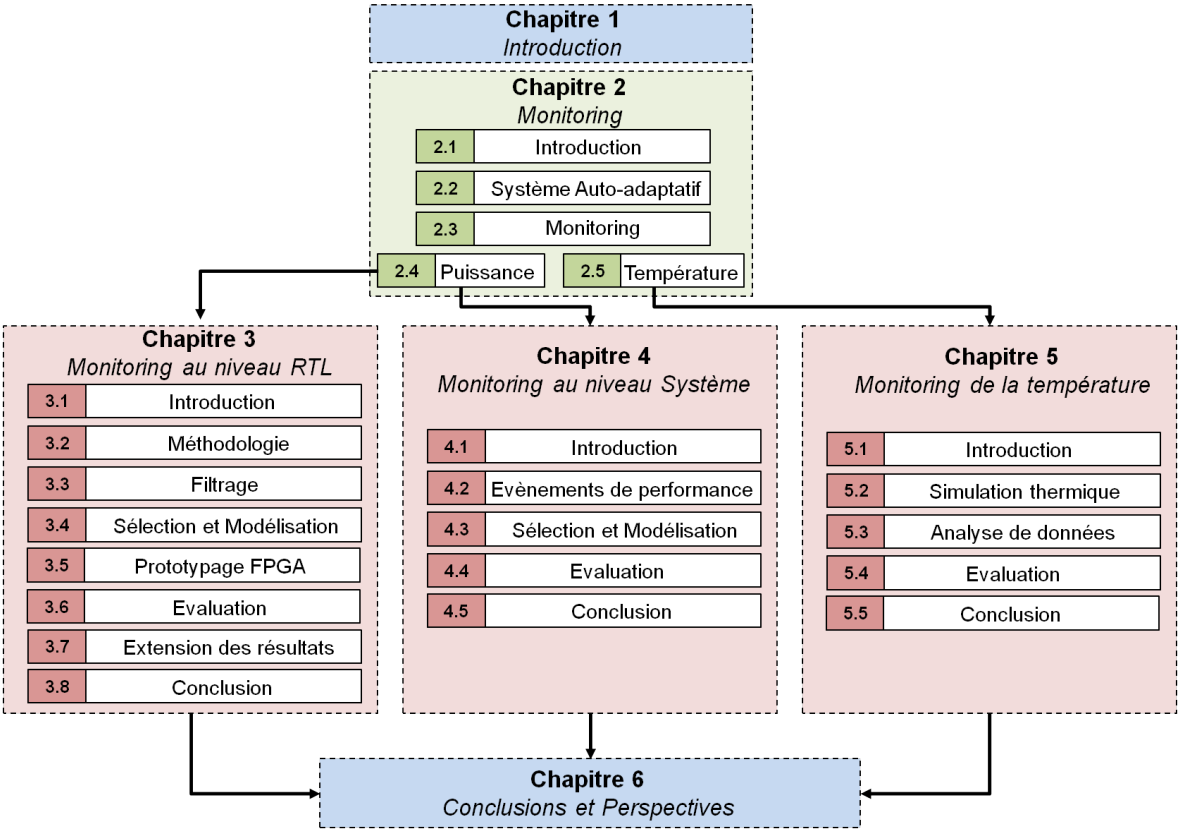


FIGURE 1.3 – Représentation graphique du plan du manuscrit.

1.3 Organisation du manuscrit

Ce manuscrit est divisé en 5 chapitres (Figure 1.3). Dans le Chapitre 2, nous exposons un certain nombre de méthodes d'adaptation existantes. Lorsque dans le Chapitre 3, une méthode de monitoring de la puissance au niveau logique est proposé. Ensuite, dans le Chapitre 4, nous

proposons une approche d'optimisation de l'utilisation des ressources existantes dans les processeurs pour un monitoring efficace à un coût négligeable. Le Chapitre 5 expose une méthode de monitoring de la température. Un outil est développé pour réaliser des simulations thermiques des circuits. Enfin, une conclusion ainsi que les perspectives de ce travail sont proposées dans le Chapitre 6.

CHAPITRE 2

MONITORING DES SYSTÈMES INTÉGRÉS AUTO-ADAPTATIFS

« *Le succès, c'est se promener d'échecs en échecs tout en restant motivé* »

Winston Churchill

Sommaire

2.1 Introduction

Dans ce chapitre, nous nous intéressons à l'état de l'art des méthodes de monitoring dans les systèmes intégrés auto-adaptatifs et plus particulièrement celles qui visent l'observation en ligne de la puissance et la température. Avant de commencer à introduire ces méthodes, il est nécessaire dans un premier temps de bien présenter les notions relatives aux systèmes intégrés auto-adaptatifs.

2.2 Système Auto-adaptatif

2.2.1 Définition

La prise en compte des contraintes liées à la consommation et la fiabilité dues à la complexité grandissante des systèmes intégrés sur puce, nécessite la mise en place des solutions nouvelles s'appuyant notamment sur des circuits auto-adaptatifs, qui sont définis comme des systèmes capables de modifier en ligne certains paramètres de fonctionnement, sur la base d'un objectif d'optimisation.

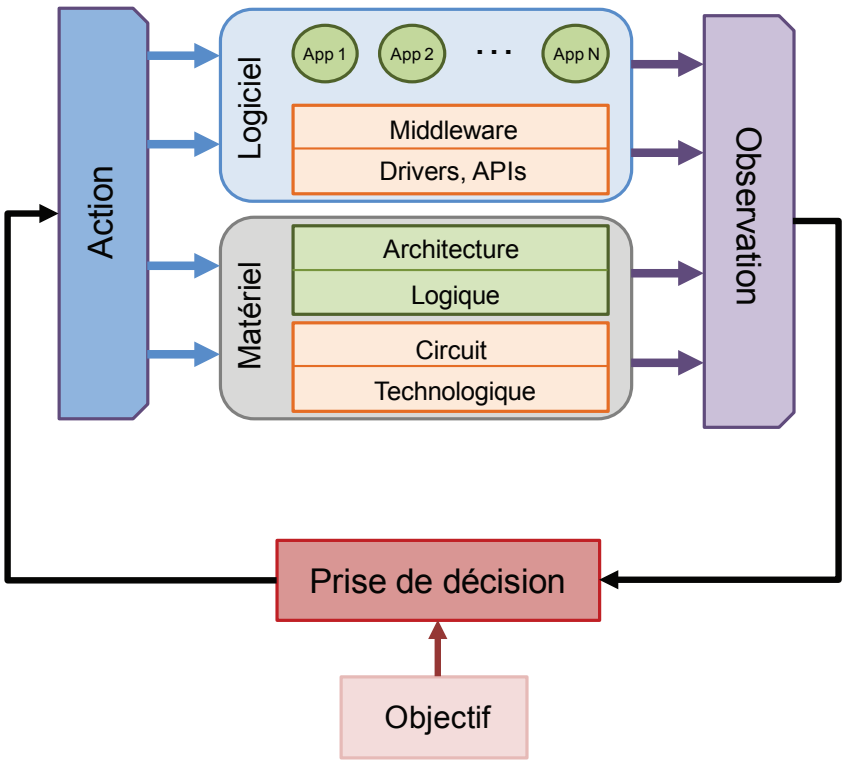


FIGURE 2.1 – Représentation d'un système auto-adaptatif intégré.

Un système auto-adaptatif intègre un processus de contrôle intelligent illustré dans la Fi-

gure 2.1, faisant basculer le système d'une configuration à une autre pour : (*i*) améliorer la performance et la qualité de service par la gestion dynamique des ressources, (*ii*) maximiser l'efficacité énergétique par l'optimisation en ligne de l'utilisation des ressources, ou bien (*iii*) pour améliorer la fiabilité par exemple contre le phénomène de vieillissement accéléré par la présence des points chauds dans le circuit.

2.2.2 Action

L'adaptation est réalisée par la modification en ligne des caractéristiques de fonctionnement à l'aide des actionneurs, pour passer à un état plus adéquat qui satisfait le critère d'optimisation. Le choix d'actionneur se fait en fonction de la configuration système en vue d'être déployé, et également des mécanismes offerts par l'architecture. Sa latence doit être en adéquation avec le temps de contrôle ciblé. Nous pouvons classer ces dispositifs selon les deux niveaux : *Logiciel* si l'action est appliquée sur les programmes ou tâches exécutés par le système, et *Matériel* si elle porte sur un paramètre physique ou logique du circuit.

Logiciel

Au niveau applicatif, il y a en premier lieu les techniques d'ordonnancement (*task scheduling*) qui consistent à remplacer une tâche par une autre sur le même cœur. Cette technique est anticipée dans les méthodes pro-actives pour éviter le surchauffe du circuit [SAR12]. Par ailleurs, la gestion des tâches au niveau "*Multi-Processor System on Chip*" (MPSoC) est effectuée à l'aide d'une unité de gestion globale (*global manager*) qui prend en charge le placement des tâches sur les cœurs, et le basculement en cours d'exécution d'une tâche d'un processeur à un autre (migration). Ces techniques sont fréquemment utilisées pour rééquilibrer la température [MB08], ou dans le cadre de la réallocation des ressources en vue de meilleures performances [SOM+07].

Matériel

En ce qui concerne les actions matérielles, elle peuvent s'effectuer à deux niveaux : architecture ou circuit. La première intègre une boucle de contrôle au niveau des composants de l'architecture, où l'action est réalisée par le biais d'un processus matériel qui réagit sur le fonctionnement du composant. Nous citons, par exemple, la limitation des instructions *fetch* (*fetch throttling*) au niveau du processeur pour une gestion dynamique de la température [SAS02], la re-configuration de la taille et de la politique du cache pour maximiser l'efficacité énergétique [SJT11], le découpage des lignes de cache de données (*sub-banking data cache*) [SD95], et le

routage des paquets pour éviter la congestion sur les liens d'un "*Network on Chip*" (NoC) afin d'améliorer la performance du système multi-cœurs [vdBCGB07].

Au niveau circuit, les actions s'appliquent directement sur les paramètres physiques tels que la fréquence de l'horloge, la tension d'alimentation (V_{dd}) et la tension de polarisation du substrat des transistors (V_{bb} tension de *body-biais* pour les technologies FDSOI). La régulation de l'horloge dans les circuits intégrés est souvent réalisée à l'aide d'un dispositif constitué d'une boucle d'asservissement permettant d'assigner une fréquence de sortie conditionnée par un signal ("*Phase Locked-Loop*" (PLL) [Bes07] ou "*Frequency Locked-Loop*" (FLL) [LPB+11]). En outre, la tension d'alimentation est ajustée à l'aide d'un (ou plusieurs) convertisseur DC-DC qui permet de passer d'un niveau de tension à un autre d'une manière continue ou discrète (V_{dd}-Hopping). Ces actionneurs sont impliqués dans des mécanismes de DVS/DVFS (*Dynamic Voltage/Frequency Scaling*) qui gèrent la fréquence et la tension pour atteindre un compromis adéquat entre la performance du circuit et sa consommation [CH10], et de gestion des cœurs actifs par basculement en mode basse consommation par l'arrêt de l'horloge (*clock-gating*) ou bien de "power switch" permettant l'activation/désactivation de domaine de consommation ("*Voltage and Frequency Island*" (VFI)).

2.2.3 Prise de décision

La gestion des actionneurs précédemment détaillés est effectuée à l'aide d'une unité de prise de décision, généralement associée à un objectif d'optimisation. Dans cette thèse, on s'intéresse plus particulièrement aux méthodes qui visent l'optimisation énergétique/thermique, les contraintes de temps réels, et la fiabilité du système au cours du temps. Les méthodes qui assurent le bon fonctionnement du circuit à travers un mécanisme de débogage ou bien la robustesse contre les fautes et les attaques (sécurité), sortent du cadre de nos travaux. Ainsi, la prise de décision peut être aussi effectuée sur les deux niveaux d'abstraction.

Logiciel

Le fait de décider de la configuration de l'architecture au niveau de la couche applicative se révèle d'un grand intérêt. D'une part, on réutilise les mêmes ressources de l'architecture pour mener le processus de prise de décision. Par conséquent, le coût matériel est fortement diminué si l'on considère uniquement l'empreinte mémoire du contrôle. D'autre part, les concepteurs sont moins contraints au niveau complexité, et la prise de décision peut traiter plusieurs critères avec une vision globale et moyennant des méthodes de résolution avancées.

Il existe dans la littérature de nombreuses méthodes de prise de décision à ce niveau. Nous citons ici, par exemple, l'ensemble des services OS ("*Intelligent Energy Manager*" (IEM)) [ARM10b]

dans l'architecture ARM qui permettent de classer les phases d'applications en fonction de leur charge de travail pour prédire les performances exigées afin d'ajuster la tension et la fréquence, et les politiques d'ordonnancement dynamique des tâches orientées vers le contrôle des architectures mono- et multi-cœurs qui décident l'emplacement et la durée d'exécution des tâches selon la métrique à optimiser (énergie ou performance) [CK07].

Matériel

La prise de décision au niveau matériel est constituée d'un composant intégrant des mécanismes plus ou moins élaborés. Si l'architecture dispose d'un nombre limité de configurations (par exemple la profondeur du pipeline, structure et hiérarchie du cache, fenêtre d'instructions, les niveaux de tension *etc.*), une simple évaluation de chaque mode ou une combinaison de modes permet de positionner le système sur la configuration la plus adéquate aux besoins du contexte [HRYT00]. Certes, cette approche requiert plus d'attention si le nombre de configurations est élevé, dans le sens ou la durée d'évaluation de tous les choix possibles doit rester raisonnable. En outre, quand la configuration requise est une valeur numérique d'un paramètre donné ou bien le nombre de configurations est relativement grand, les mécanismes les plus avancés ont recours à (*i*) des approches analytiques se basant sur des modèles tels que la politique d'optimisation de la performance du processeur sous contrainte de température qui exprime analytiquement la fréquence en fonction de la température [RVCC06], ou bien (*ii*) à des heuristiques basées sur la théorie du contrôle telles que la théorie des jeux ou de consensus pour configurer, par exemple, le couple tension-fréquence [WJMC04].

2.2.4 Synthèse

La politique d'adaptation dépend principalement du type d'actionneurs et de l'objectif d'optimisation, qui impose une analyse :

- Au niveau temporel : la **granularité temporel** (ou latence) reflète la durée pendant laquelle la boucle d'adaptation sera impliquée (temps entre le moment de l'observation et l'action). Ce paramètre dépend de la criticité de l'adaptation, *i.e.* s'il y a une urgence d'action ou pas. Par exemple, la gestion de la fiabilité contre les surchauffes du circuit est réalisée par des mécanismes relativement rapides tels que l'arrêt du processeur (*power-off*) ou bien la régulation de la fréquence/tension (DVFS), où la latence est de quelques dizaines de cycles d'horloge (centaines de microsecondes). D'autres méthodes d'adaptation ciblent par exemple l'amélioration de la performance, où la latence est plus élevée (mais moins critique).
- Au niveau spatiale : la **granularité spatiale** permet d'identifier à quel niveau est réalisé

l'adaptation. Afin d'illustrer ce point, nous allons prendre l'exemple de l'amélioration de la performance. Pour ce faire, il est possible de régler la fréquence de toute la puce, ou bien de réagir à grain plus fin en distribuant la charge au niveau de chaque cœur du circuit.

La Figure 2.2 illustre l'ensemble du cycle d'adaptation. La prise de décision au niveau logiciel concerne plutôt des mécanismes qui sont moins critiques adressant la gestion du comportement du système à travers des actionneurs logiciels tels que l'ordonnancement des tâches, ou bien matériels tels que la gestion du couple tension-fréquence par le *gorvernor* d'un système d'exploitation. En fait, le passage de message d'une couche logicielle à une autre matérielle nécessite l'exécution d'un ensemble d'instructions représenté par des APIs au niveau du *kernel* système où la latence peut atteindre une centaine de microsecondes. Dans le cas où la criticité du contrôle est majeure, la prise de décision et l'action se situent généralement au niveau matériel pour réduire cette latence. On se retrouve ainsi avec un contrôle qui réagit en quelques cycles d'horloge.

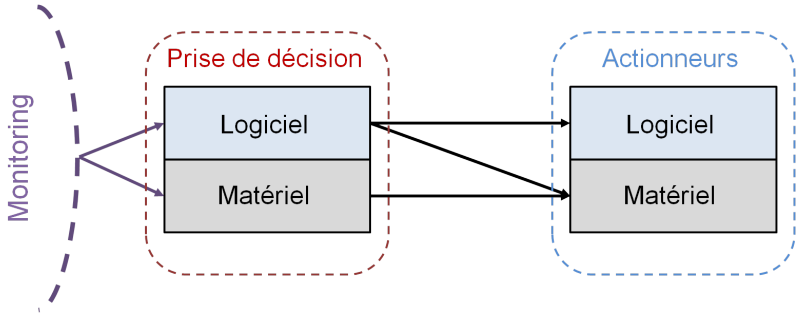


FIGURE 2.2 – Illustration abstraite du cycle d'adaptation.

Sous ces contraintes, la mise en œuvre d'une boucle d'adaptation nécessite en premier lieu la spécification d'une phase d'observation en ligne (*monitoring*) qui sera chargée de reporter des informations sur l'état du système à une granularité spatiale et temporelle compatibles avec le type d'actionneur et l'objectif d'optimisation.

2.3 Monitoring dans les systèmes auto-adaptatifs

Nous avons vu précédemment l'action et la prise de décision dans les systèmes adaptatifs. Dans cette section, nous allons présenter le processus de monitoring qui complète la boucle de contrôle dans les systèmes auto-adaptatifs.

2.3.1 Définition

Nous commençons cette section par quelques définitions nécessaires à la bonne compréhension de la suite :

- Le **mesurage** est l'opération ou l'ensemble d'opérations permettant l'obtention d'une valeur qui représente un état du système.
- La **mesurande** est l'information soumise au mesurage, et qui peut être une grandeur physique ou bien une information logique.
- Le **mesure** est une représentation quantifiée d'une mesurande.
- L'**État** est un ensemble de mesures caractéristiques.

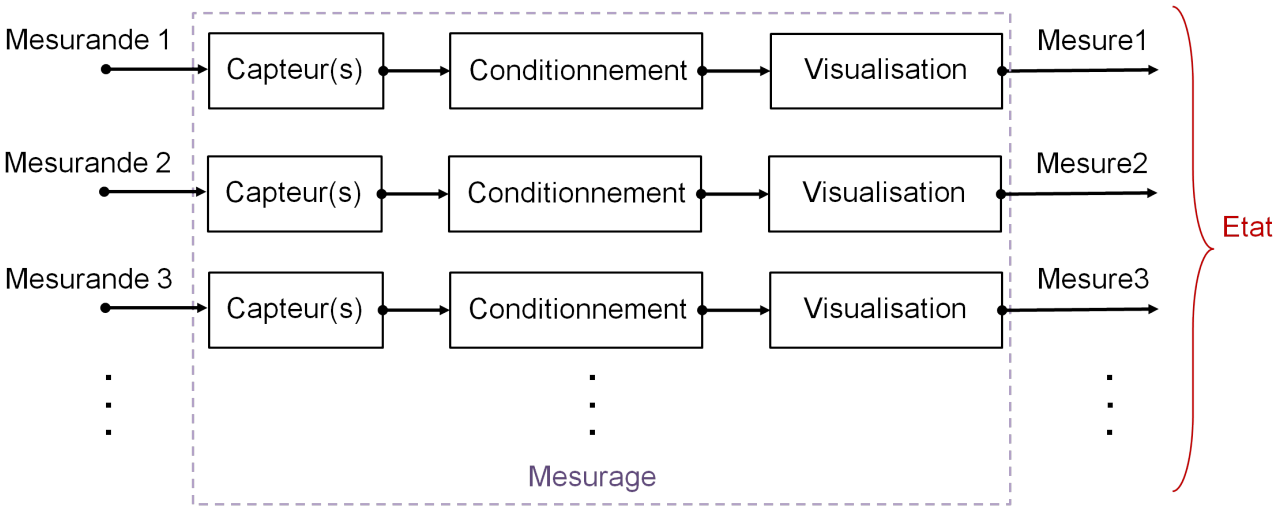


FIGURE 2.3 – Structure d'une chaine d'acquisition.

Connaissant ces éléments, nous pouvons passer à la description du monitoring. Le monitoring est le processus d'observation en ligne de l'état du système dans la boucle de contrôle. Elle est constituée d'une (ou plusieurs) chaine d'acquisition qui contient au minimum trois étages différents comme représentés sur la Figure 2.3.

Le premier élément de cette chaine est le capteur. Son rôle est d'être sensible aux variations d'une mesurande que nous voulons étudier et de délivrer une autre information image de la première compréhensible par la suite de la chaine.

Le second élément de cette chaine, le conditionneur, a pour rôle : d'amplifier, d'enregistrer et d'agréger, les valeurs délivrées par le capteur pour fournir une information compatible avec l'unité de visualisation ou d'utilisation. Enfin, l'unité de visualisation et/ou d'utilisation permet de rendre la valeur de l'état du système exploitable.

Dans notre contexte où l'adaptation cible la gestion du comportement afin d'optimiser les caractéristiques de fonctionnement telles que la performance et l'énergie, l'état du système

peut être représenté par : (*i*) la performance, (*ii*) la puissance, ou (*iii*) la température du circuit. La structure présentée ici se retrouve dans toutes les méthodes de monitoring quelque soit la grandeur que l'on souhaite mesurer.

2.3.2 Méthodes de monitoring

Le monitoring peut s'effectuer à différents niveaux d'abstractions, ceci dépend de la mesurande que l'on cherche à observer. Une charge de calcul ou plus généralement un mode d'application se trouve au niveau logiciel, lorsque les informations liées à la consommation et la température se trouvent au niveau matériel.

2.3.3 Méthodes logicielles

Dans la plupart des cas, le monitoring au niveau système sert au profilage des applications décrivant ainsi la performance. Dans ce cas, le capteur est représenté par un processus d'instrumentation du code. Nous pouvons ainsi distinguer deux types de monitoring : (*i*) l'instrumentation qui est basée sur un ensemble d'instructions ajoutées au niveau binaire de l'application durant la compilation, ou bien (*ii*) un processus séparé qui capte les transitions des processus ou bien l'activité de l'ordonnanceur [dM10]. Le conditionnement correspond ainsi à un compteur qui représente le nombre de changements par intervalle de temps pour ensuite exprimer la performance comme une valeur numérique correspondant à l'activité au niveau système.

Les systèmes d'exploitation offrent de leur côté, une interface d'observation vis-à-vis du matériel. Des threads spéciaux récupèrent l'identifiant de la tâche en cours ainsi que le contenu des registres d'activité, nous pouvons parler ici d'une méthode hybride puisque le capteur et le conditionnement dans ce cas se situent au niveau matériel, tandis que la visualisation se passe au niveau logiciel. Nous pouvons citer, par exemple, les outils perf [dM10] et Oprofile [Lev04], qui fournissent un ensemble des routines pour accéder à des registres dans l'architecture matérielle.

Les processus introduits pour le monitoring engendrent l'interruption momentanée de l'exécution de l'application courante. Cette perturbation correspond au coût en performance du monitoring. En outre, le temps de réponse est un facteur de la fréquence de fonctionnement, les politiques de l'ordonnanceur des tâches et le temps requis pour exécuter le code dédié au monitoring.

2.3.4 Méthodes matérielles

L'information captée au niveau matériel est localisée au niveau du circuit, ou bien au niveau architecture. Dans le premier cas, le capteur est une structure analogique ou numérique qui cible des phénomènes liés aux variations du processus de fabrication, la dégradation en cours du temps de la technologie utilisée, *etc.* Afin de mieux représenter le monitoring à ce niveau, on considère un moniteur de chemin critique "*Critical Path Monitor*" (CPM) tel que le Razor [EDL+04], développé afin de détecter les fautes temporelles dans le circuit, et donc d'évaluer ses performances réelles après fabrication. Le monitoring du chemin critique est basé sur la même chaine d'acquisition : (*i*) un capteur permettant de capter le retard sur un signal de test après duplication, (*ii*) ce retard est numérisé ensuite à partir d'un comparateur et/ou un convertisseur, (*iii*) la visualisation indique ainsi s'il y a violation de temps de propagation.

Contrairement aux grandeurs physiques, l'information logique est plus synthétique et peut-être confinée plus longtemps au sein de la structure de monitoring. Cette notion d'historique est souvent indispensable pour caractériser l'application en cours vis-à-vis de l'architecture mise en œuvre. Cette fonction est assurée par des capteurs d'activité spécifiques. Il s'agit donc d'intégrer des dispositifs caractérisant l'activité engendrée au sein de l'architecture au cours de l'exécution d'une application quelconque. Cette activité peut être captée sur des signaux internes du système pour caractériser l'activité physique au niveau des composants [OMM08], ou bien elle peut correspondre à un ensemble d'états caractérisant ainsi la performance de l'architecture d'où l'appellation fréquente de capteurs de performance (on parle ici de performances de l'application une fois qu'elle est mappée sur l'architecture, et non pas des performances de timing caractéristiques du silicium). Actuellement, la plupart des processeurs modernes sont dotés de ce type de capteurs intégré : ARM CortexA [ARM10a], Intel Sandy Bridge [Wil12], IBM Power6 [Lia09], *etc.*

Les informations logiques captées sont généralement agrégées pendant des périodes de temps régulières à l'aide d'un compteur (ou conditionneur). Ainsi, la performance sera représentée comme une valeur numérique telle que le nombre d'instructions par intervalle de temps, le nombre de cycles de processeur, *etc.* La visualisation de ces informations au niveau système est utile dans certains cas pour l'optimisation algorithmique, le débogage, ou bien même pour l'auto-adaptation. C'est la raison pour laquelle, les compteurs de performance sont souvent connectés comme des périphériques sur le bus système dans les processeurs actuels, pour qu'ils soient accessibles par un processus logiciel. Dans ce cas, on parle aussi d'une approche de monitoring hybride, où le capteur et le conditionnement sont des dispositifs matériels et la visualisation est un processus logiciel.

Dans tous les cas, l'intégration de dispositifs matériels va engendrer un surcout représenté par la surface silicium et l'énergie consommée par ces dispositifs. Cependant, le temps de ré-

ponse est plus rapide que le monitoring logiciel, et dépend de la période d'alimentation du compteur.

2.3.5 Bilan

Nous avons vu que les deux paramètres suivants influent sur les caractéristiques de la méthode de monitoring :

- La latence qui correspond au temps de réponse, et qui doit être compatible avec le niveau de réactivité du processus de prise de décision.
- Le **coût** en performance, surface silicium et énergie engendré par le processus de monitoring.

Comme nous l'avons expliqué, le monitoring logiciel va entrainer des changements au niveau du comportement de l'application (période, temps de réponse, *etc.*), qui à leur tour provoquent la non maîtrise des contraintes temps réel. Il se peut que l'information récupérée ne reflète plus l'état actuel avec la **précision** souhaitée, un exemple qui s'applique notamment lors de l'observation des phénomènes physiques tel que la variation de la puissance ou bien la température. D'autre part, le monitoring doit répondre au besoin de la prise de décision qui peut s'appliquer à une granularité fine.

Le tableau 2.1 compare les caractéristiques principales d'une méthode de monitoring. Le monitoring au niveau matériel permet d'avoir une information précise sur les performances du système avec une résolution temporelle fine. Cependant, le coût en silicium et énergie est généralement important ce qui pose la question de scalabilité de l'approche pour observer un système complexe. Au niveau logiciel, l'information peut être moins précise, mais le surcout est représenté par le coût en énergie et performance (y compris l'espace mémoire nécessaire pour la mise en œuvre du monitoring), permettant ainsi l'observation en ligne des circuits multi-cœurs complexe. Le monitoring hybride combine les avantages des deux méthodes puisque la chaîne d'acquisition est distribué sur les deux niveaux d'abstractions.

TABLEAU 2.1 – Comparaison des approches de monitoring.

Niveau d'abstraction	**Précision**	**Résolution temporelle**	**Coût**		
			performance	**silicium**	**énergie**
Matériel	élevée	élevée	-	élevé	élevé
Hybride	moyenne	moyen	moyen	moyen	moyen
logiciel	-	faible	élevé	-	faible

Une prise de décision efficace repose surtout sur un monitoring fiable. Il est ainsi question de comment définir la solution optimale qui permet de produire l'information la plus précise

de l'état du système à faible coût en satisfaisant les contraintes de la prise de décision liées à la granularité spatiale et à la résolution temporelle. Dans cette thèse, nous avons proposé une approche innovante qui intervient en amont (au moment de la conception du circuit) afin de retrouver cette solution pour le monitoring de la puissance et de la température qui constituent deux informations fondamentales pour l'optimisation énergétique et l'amélioration de la fiabilité du circuit contre le vieillissement. C'est donc naturellement que, dans les prochaines sections, nous nous pencherons sur l'étude de l'état de l'art du monitoring de la puissance et de la température.

2.4 Monitoring de la puissance

Dans cette section, nous allons présenter en premier lieu le modèle physique qui exprime la puissance en fonction d'un certain nombre de paramètres et de variables. Ensuite, nous nous penchons sur la description des différentes méthodes qui visent le monitoring de la puissance.

2.4.1 Modèle physique

La puissance est une grandeur physique qui représente la consommation en énergie du circuit pendant un intervalle de temps. Elle peut s'exprimer par :

$$\mathrm{P}_{totale} = \mathrm{V}_{dd} \times \mathrm{I} = \mathrm{P}_{dyn} + \mathrm{P}_{cc} + \mathrm{P}_f = \frac{1}{2}.\alpha.\mathrm{C}.\mathrm{V}^2{}_{dd}.f + \mathrm{V}_{dd}.\mathrm{I}_{cc} + \mathrm{V}_{dd}.\mathrm{I}_f \tag{2.1}$$

La puissance totale (P_{totale} en Watts) est le produit de la tension d'alimentation V_{dd} (en Volts) par l'intensité de courant I (en Ampères). Elle peut être aussi exprimé comme étant la somme d'une composante dynamique P_{dyn} qui dépend de l'activité physique interne α au niveau des transistors, une statique P_f qui est proportionnel au courant de fuite dans les transistors I_f (ceci est une fonction de la température), et une dernière P_{cc} qui est dû au courant de court-circuit I_{cc} et qui est d'ailleurs souvent négligé. Dans l'équation (2.1), la puissance dynamique P_{dyn} est aussi exprimée par $\frac{1}{2}.\alpha.\mathrm{C}.\mathrm{V}^2{}_{dd}.f$, où C est la capacité équivalente. f et v_{dd} sont la fréquence et la tension d'alimentation, et correspondent aux paramètres réglés par les actionneurs dans la boucle de contrôle ; l'information qui représente ces deux paramètres est souvent disponible dans le système. L'activité α et la température sont les deux variables du modèle qui nécessitent l'insertion de capteurs pour la collection en ligne de ces informations.

D'après ce modèle, nous pouvons classer les méthodes de monitoring en deux catégories : (*i*) les méthodes directes qui utilisent des capteurs pour mesurer le courant consommé, et (*ii*) les méthodes indirectes qui estiment la puissance en fonction des paramètres de fonctionne-

ment du système et des informations sur l'utilisation des ressources.

2.4.2 Mesure directe

La mesure directe de la puissance consommée par le circuit est généralement basée sur l'intégration de capteurs de courant à l'intérieur ou bien à l'extérieur de la puce. L'approche industrielle consiste à rajouter dans les téléphones portables modernes, des puces externes "*Power Management Integrated Circuit*" (PMIC) intégrant ces capteurs (tel qu'on trouve dans le Samsung Galaxy-S avec le processeur ARM Big.LITTLE [LIM13]). Le PMIC est situé entre la batterie et certains composants (processeur, écran, *etc*) tel qu'illustré dans la Figure 2.4. Son rôle principal est de mesurer le courant consommé sur chaque rail d'alimentation à l'aide des capteurs intégrés. Ayant une interface de communication I2C, il est aussi accessible par des routines au niveau système d'exploitation. Auparavant, le PMIC jouait le rôle d'observation de l'état de la batterie pour une visualisation à l'écran, où les services accèdent en lecture aux registres internes du PMIC pour retrouver les valeurs numériques qui reflètent la quantité de charge restante (et donc la consommation). Son rôle a évolué récemment avec les options rajoutées par les concepteurs dans les processeurs modernes. Le PMIC intègre des mécanismes de gestion de la tension d'alimentation (DVS), et reçoit des commandes du processeur pour basculer en mode basse consommation (par exemple le MAXIM MAX77686 dans les processeurs Exynos). Ces modes sont généralement spécifiés au niveau de l'OS par un ensemble d'états (C-States, P-States et T-States) qui définissent le mode de fonctionnement de chaque composant (éteint/allumé, tension, fréquence, *etc.*).

L'utilisation d'un circuit intégré pour le monitoring de la puissance peut être intéressant dans certains cas, notamment lorsqu'il intègre des mécanismes de gestion de la consommation du circuit. Cependant, ces circuits ne produisent qu'une information de la puissance consommée sur les lignes d'alimentation. La mise en œuvre d'une méthode d'adaptation nécessite parfois une information à une granularité plus fine (par exemple au niveau de chaque cœur). Dans le but d'améliorer la granularité de cette approche, l'alimentation des différents modules est séparée dans le processeur ARM Big.LITTLE permettant ainsi le monitoring de la puissance consommée par bloc (petit cœur, grand cœur, mémoire externe, entrées sorties du système, *etc.*).

Dans ces approches, la mesurande est souvent le courant qui traverse une ligne d'alimentation. À titre d'exemple, nous allons nous pencher sur le capteur de puissance proposé dans [BJ12a]. Le circuit équivalent est montré dans la Figure 2.5, où le capteur est conçu pour une technologie SOI 45-nm avec un transistor de veille (*sleep*) pour réduire la puissance consommée par le capteur lorsque le circuit est inactif. Le but principal est de mesurer le courant I_{load} consommé par la charge (*Load*). Ce courant traverse le transistor de vieille et provoque une

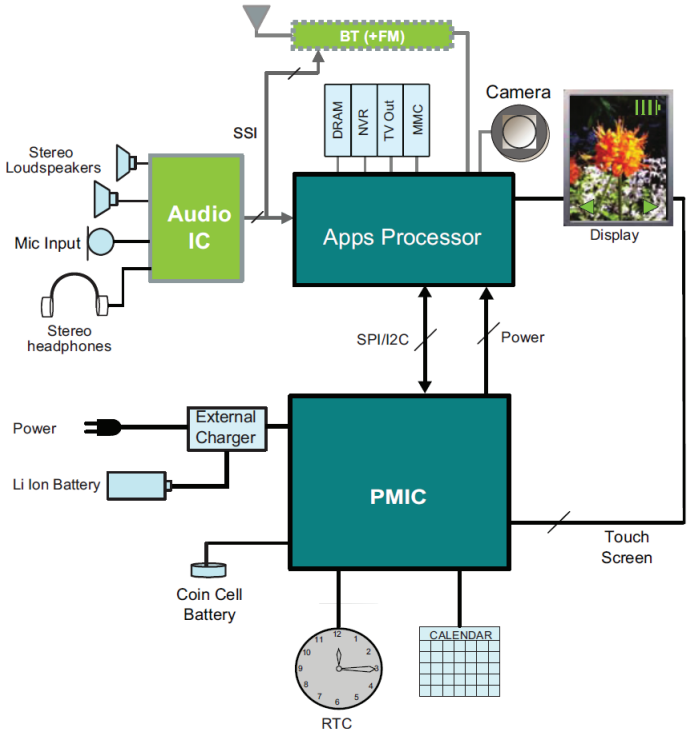


FIGURE 2.4 – Représentation simplifiée de l'intégration d'un PMIC dans un téléphone portable (adapté de [Sem13]).

chute de tension à cause de la résistance de veille équivalente (R_{sleep}). Cette chute de tension est retenue par le transistor M2, qui à son tour, le transmet à un autre transistor M3 avec une capacité de charge C_a . Un comparateur par rapport à une tension minimale de variation est ensuite utilisé pour décharger cette capacité. Le conditionnement du signal est réalisé par la chaine d'inverseurs qui transforme en pulse la tension au niveau de la capacité. De ce fait, le courant qui charge la capacité et la fréquence de commutation de la chaine d'inverseurs sont proportionnels au courant consommé I_{load} pour une température constante. Un circuit logique (*TFF*) est enfin utilisé pour représenter la puissance consommée par la charge (*Load*) par une valeur numérique qui correspond au nombre d'impulsions en sortie de la chaine d'inverseurs.

Les avantages principaux des mesures directes sont la haute résolution temporelle et la précision élevée en mesurant la puissance totale. Puisque les trois éléments de la chaine d'acquisition sont constitués d'éléments matériels, le coût du monitoring correspond donc à la surface silicium et l'énergie consommée par le capteur. L'intégration des capteurs matériels ne dégrade pas la performance de l'application. Le capteur de courant présenté précédemment, par exemple, permet de mesurer la puissance pour une résolution de 0.5µs à une précision d'environ 3%, et occupe 140µm × 140µm en technologie 45-nm.

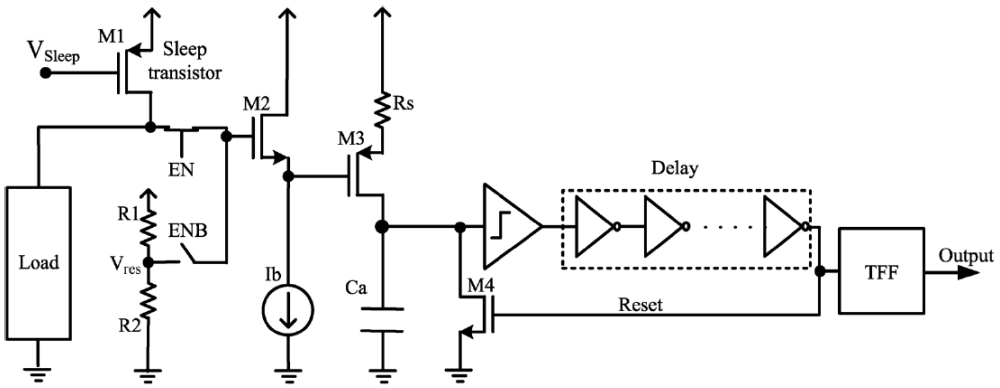


FIGURE 2.5 – Circuit équivalent d'un capteur de courant sur puce [BJ12a].

2.4.3 Estimation indirecte

La deuxième approche pour le monitoring de la puissance consiste à estimer la valeur de puissance en utilisant les variables qui correspondent à l'activité et à la température, avec les deux paramètres ajustables : la fréquence et la tension d'alimentation. La chaine d'acquisition pour le monitoring de la puissance est représenté dans la Figure 2.6.

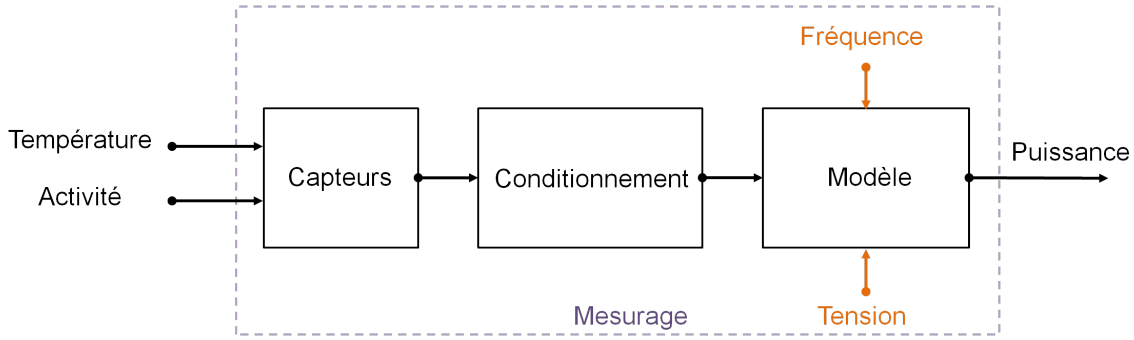


FIGURE 2.6 – Structure de la chaine d'acquisition pour l'estimation indirecte de la puissance.

Estimation de l'activité

En général, les applications sont constituées d'un ensemble d'instructions qui activent le système différemment. Par exemple, les applications dédiées au traitement de données ont tendance à provoquer des instructions d'accès aux caches et à la mémoire, alors que les applications de type "communication" provoquent des appels systèmes permettant aux modules de communication de transmettre ou bien de recevoir des paquets à travers des interfaces dédiés. En outre, les composants du système consomment différemment de l'énergie selon leur charge de travail. L'activité qui est proportionnelle aux milliards d'opérations logiques, provoque ainsi une variation de la puissance dynamique consommée.

Plusieurs méthodes ont été proposées dans la littérature pour le monitoring de la puissance au niveau système en fonction de l'activité, en supposant les autres paramètres tels que la fréquence de fonctionnement et la température de la puce comme constants. Dans plusieurs méthodes, l'activité est captée par les compteurs de performance au niveau de l'architecture. Dans [CKS+09], un modèle linéaire simple est proposé pour estimer la puissance consommée par la plateforme ARM11 MPCore en fonction de cinq évènements de performance : le nombre d'instructions exécutées, le nombre d'accès au cache L1 des données, le nombre d'accès au cache L2, le nombre de cycles de processeur et le nombre de transactions conséquentes au protocole de cohérence des caches. Dans [BWeWK03], la puissance est exprimée linéairement en fonction de l'énergie consommée par évènement interne du processeur tel que l'exécution d'une instruction de prédiction. Une approche différente est proposée dans [IM03, PMV13], où un facteur qui représente sa contribution à la consommation en puissance est attribué à chaque composant du système. Ces facteurs estimés à l'aide du taux d'accès basé sur les événements de performance : le facteur pour le bus est par exemple proportionnel au nombre de transactions de lectures et d'écritures effectuées.

L'observation en ligne des évènements de performance repose généralement sur une méthode hybride qui intègre des compteurs matériels configurables pour compter l'occurrence des évènements au niveau du processeur et des périphériques, et des compteurs logiciels qui prennent en charge le monitoring de l'activité au niveau du système d'exploitation et au niveau des tâches. Il existe cependant des milliers d'évènements au niveau système [SGG+08] : la sélection des évènements pertinents pour caractériser l'activité interne du système est un problème essentiel qui n'est pas adressé d'une manière systématique par les méthodes présentées, qui reposent toutes sur l'estimation de la puissance en fonction des évènements de performance. L'activité a été approximée par quelques événements sélectionnés sur la base des observations, qui rend ces méthodes très spécifiques à l'architecture matérielle.

En principe, la puissance est exprimée en fonction de l'activité physique correspondant aux taux de commutation des transistors. Le fait de remonter au niveau architecture pour approximer cette activité par un ensemble d'évènements de performance, peut avoir un effet négatif sur la précision des estimations. Une approche différente est proposé dans [MBTC13], où l'activité est représentée par un ensemble d'évènements de commutation sur des signaux internes physiques. Les auteurs ont présenté une méthode pour le monitoring de l'énergie basée sur les chaines de Markov. La puissance est représentée par un ensemble fini de modes de consommation. Ils utilisent un compteur d'activité 32-bits pour compter l'occurrence d'un évènement de commutation sur un signal spécifique du circuit. Ce compteur est configuré par un signal différent pour chaque mode de consommation. La sélection des signaux pertinents est basée sur l'heuristique *stepwise* qui est une technique développé par les statisticiens afin de repérer la variable la plus intéressante pour une régression multiple. Cette approche est intéressante mais suppose un ensemble fini de modes. Elle convient à de petits composants tels que les mé-

moires, mais elle est trop limitée pour les composants complexes tels que les processeurs et les systèmes multi-cœurs.

Modélisation macroscopique

Plusieurs méthodes existantes font l'hypothèse d'une activité constante par application (activité moyenne). Selon cette hypothèse, les auteurs de [SKUY15] expriment la puissance consommée par les clusters dans la plateforme ARM Big.LITTLE sous :

$$\mathrm{P}_{totale} = \alpha.\mathrm{C}.\mathrm{V}^2{}_{dd}.f + \mathrm{V}_{dd}.(c_1.\mathrm{T}^2.e^{\frac{c_2}{\mathrm{T}}} + \mathrm{I}_{gate}) \tag{2.2}$$

où, T est la température de la puce, et c_1, c_2 et I_{gate} sont des constantes obtenues en faisant faisant varier dans chaque expérimentation une des variables de l'équation. Le courant de fuite I_f est approximé comme étant une fonction exponentielle de la température de la puce T.

D'autres méthodes négligent aussi l'impact de la température. La puissance peut s'exprimer ainsi seulement en fonction de paramètres, la fréquence et la tension d'alimentation. L'équation équivalente proposée dans [EFH09] est montrée dans (2.3), où k_1 et k_2 sont deux constantes qui déterminent le budget énergétique de chaque application.

$$\mathrm{P}_{totale} = k_1.\mathrm{V}^2{}_{dd}.f + k_2.\mathrm{V}_{dd} \tag{2.3}$$

Dans une adaptation par DVFS, la fréquence et la tension d'alimentation sont modifié en même temps. L'impact de ce couple sur les variations de la puissance est souvent approximé selon f^β, avec β compris entre 2 et 3 [Gha12]. L'équation de la puissance (2.3) peut être ainsi représenté par (2.4), où β est l'exposant de la fréquence et d, l et m sont des constantes. La méthode proposée dans [WMW09, SII13] suppose que β est égale à 1 pour approximer la puissance par une équation linéaire qui va alimenter des méthodes d'adaptation. La constante ainsi factorisée de d et l , appelée α_i dans [WMW09] et U_i dans [SII13], est approximée pour chaque application comme étant l'activité moyenne à une fréquence f.

$$\mathrm{P}_{totale} = d.f^\beta + l.f + m \tag{2.4}$$

La partie statique de la puissance varie exponentiellement en fonction de la température. Une approximation linéaire en utilisant la série de Taylor premier ordre est proposée dans [Gha12] pour inclure ce paramètre à la modélisation de la puissance dans l'équation (2.4). Dans toutes ces méthodes, l'activité est considérée constante. Pour cela, ces modèles peuvent avoir une bonne précision dans certains cas, notamment pour l'estimation de la puissance moyenne, mais restent très limités pour suivre les variations fines de la puissance.

2.4.4 Bilan

Nous avons présenté les différentes méthodes existantes pour le monitoring de la puissance. Le tableau 2.2 résume ces méthodes qui reposent sur des mesures directes, ou bien des estimations indirectes.

Les sondes de courant et de puissance permettent un monitoring plus précis de la puissance (erreur <3%) pour une résolution temporelle très fine (<1ms). Cette approche prend en compte tous les paramètres qui influent sur la variation de la puissance. Par contre, elle est très coûteuse et ne permet qu'une estimation spatiale "gros grain". Dans l'idéal, la méthode de monitoring doit être aussi précise qu'une sonde de courant avec un coût très faible permettant une meilleure observation à une granularité spatiale et temporelle fine.

Les méthodes indirectes sont moins précises (erreur 3-7%) pour des résolutions temporelles plus élevées (1-500ms) par rapport aux mesures directes. Nous pouvons constater qu'une partie de ces approches considèrent une activité moyenne constante prenant en compte seulement les variations de la fréquence et la température [BWeWK03, IM03, PMV13, BJ12b], tandis que l'autre partie suppose les paramètres du circuit constants en surveillant la puissance en fonction de l'activité représentée par un ensemble d'évènements de performance [Gha12, SKUY15]. Le coût matériel en surface et énergie de ces méthodes dépend essentiellement du coût des compteurs de performance. Le coût en performance est relativement faible, et correspond au temps requis par les processus au niveau logiciel.

L'approche proposée dans [MBTC13], atteint une précision élevée (erreur 3-10%) pour une résolution temporelle proche de celle des sondes de courant. Cette méthode estime la puissance en fonction seulement de l'activité qui est représentée par des évènements de commutation observées à l'aide des compteurs d'évènements. L'évaluation du coût de cette approche n'est pas évoqué par les auteurs.

TABLEAU 2.2 – Comparaison des méthodes de monitoring de la puissance et énergie consommée dans les systèmes embarqués.

Publication	**Approche**	**Précision**	**Résolution temporelle**	**Coût en surface**	**Coût en puissance**	**Coût en performance**	**Capteurs**
[BJ12a]	Directe	3.5% (f=100MH->1GHz)	0.5µs	140x140µm	120µW (V_{dd}=1.2V et T=26.7°C)	-	Sonde de courant
[WRF+10]	Directe	2-3%	1ms	Non précisé	Non précisé	-	IBM Sonde de courant/tension
[BJ12a]	Directe	3.5% (f=100MH->1GHz)	0.5µs	140x140µm	120µW (V_{dd}=1.2V et T=26.7°C)	-	Sonde de courant
[BWeWK03]	Indirecte	<7%	1ms	Non précisé (coût d'un PMU)	Non précisé (coût d'un PMU)	<1%	Compteurs de performance
[IM03]	Indirecte	6-12%	440ms	Non précisé (coût d'un PMU)	Non précisé (coût d'un PMU)	Non précisé	Compteurs de performance
[PMV13]	Indirecte	8-10%	500ms	Non précisé (coût d'un PMU)	Non précisé (coût d'un PMU)	<0.5%	Compteurs de performance
[BJ12b]	Indirecte	<9%	1s	Non précisé (coût d'un PMU)	Non précisé (coût d'un PMU)	<1%	Compteurs de performance
[MBTC13]	Indirecte	3-10%	100µs	Non précisé	Non précisé	NULL	Compteur d'activité
[SKUY15]	Indirecte	<4%	1s	Non précisé	Négligeable	Négligeable	Compteurs de performance + Température
[Gha12]	Indirecte	5%	10ms	Non précisé	Non précisé	Négligeable	Température

2.5 Monitoring de la température de la puce

Dans la section précédente, nous avons pu voir que la température est un élément principal dans l'estimation de la puissance. À cela s'ajoute, l'ensemble des méthodes de gestion de la dissipation thermique qui se basent sur l'information de la température afin d'améliorer la fiabilité du circuit contre le phénomène de vieillissement accéléré par la présence des points chauds dans le circuit. Dans cette section, nous présentons en premier lieu un modèle physique de la température, ensuite nous nous penchons sur l'ensemble de méthodes de monitoring existantes. Enfin, nous aborderons la problématique de placement des capteurs dans les systèmes intégrés.

2.5.1 Modèle physique

En régime permanent, la température peut s'exprimer selon la loi d'Ohm par :

$$T = T_a + \theta \times P_{tot} \tag{2.5}$$

L'analogie avec la loi d'Ohm électrique est la suivante : la puissance thermique circule comme un courant, et comme le matériau conducteur s'oppose à cette circulation (résistance thermique), on observe une différence de températures entre deux points situés sur le chemin de circulation de la puissance thermique. θ est la résistance thermique qui dépend des caractéristiques thermiques du matériel et de la conductivité thermique des différentes couches (résistivité et conductivité). En régime transitoire, une croissance dans la partie dynamique de la puissance va augmenter la température selon l'équation (2.5). Cette dernière aura pour impact une hausse de la partie statique de la puissance qui par conséquent, augmente aussi la température. Lorsque l'activité devient stable, ce phénomène converge vers une température constante (régime permanent). Cette relation entre la puissance et la température est souvent appelée chaine de réaction positif.

Le monitoring de la température peut s'effectuer soit par des mesures directes en utilisant des capteurs, ou bien par une estimation indirecte en se basant sur l'information de la puissance. Ces deux approches sont abordées en détail dans les deux sections suivantes.

2.5.2 Mesure directe

Les capteurs de température embarqués sur les puces sont généralement regroupés en deux catégories : (*i*) analogiques et (*ii*) numériques [LMMM08]. Les premières produisent une ten-

sion ou un courant proportionnel à la température, en tirant profit du fait que plusieurs paramètres électriques des dispositifs semi-conducteurs sont sensibles aux variations de la température. Les capteurs numériques utilisent un principe similaire sur des portes logiques qui transforment l'information liée à la température en une information logique.

Capteurs Analogiques

Il existe trois structures principales pour réaliser des capteurs de température analogiques. La première d'entre elles est basée sur l'utilisation d'une diode "thermique" [SGH+07]. Ces diodes possèdent l'avantage suivant : une variation δT_j de la température de la jonction PN entraîne une variation de tension proportionnelle aux bornes de celle-ci. Nous pouvons donc écrire : $\delta T_j = K.V_{dd}$ avec K un coefficient de proportionnalité généralement exprimé en °C/mV et typiquement compris entre 0.4 et 0.8°C/mV .

La seconde solution consiste à utiliser les caractéristiques du transistor MOS pour mesurer la température. Un capteur typique est proposé dans [SMKR97]. La variation du courant en sortie du transistor est exprimée par :

$$\frac{\delta I_{out}}{I_{out}} = (\frac{1}{\beta}.\frac{d\beta}{dT} + \frac{2}{V_T}.\frac{dV_T}{dT}).\delta T \tag{2.6}$$

V_T est la tension de seuil (en Volts), β est le facteur de gain du transistor, et δT est la variation de la température.

La dernière solution est basée sur l'utilisation de transistors bipolaires pour l'estimation de la température [BH99, Mak10, SBM+10]. Un schéma typique de capteur à base de transistor bipolaire est montré dans la Figure 2.7. L'idée principale est basée sur l'utilisation de deux transistors PNP verticale avec deux courants collecteurs différents. Selon [Mak10], la différence de potentiel au niveau des deux transistors δV_{BE} peut être exprimée par :

$$\delta V_{BE} = \frac{k.T}{q}\log(p) \tag{2.7}$$

où k est la constante de Boltzmann, q est la constante de charge. p est aussi une constante spécifiée lors de la conception du capteur et correspond au facteur de différence entre les courants qui alimentent les deux transistors (voir I et pI dans la Figure 2.7). La différence de potentiel δV_{BE} sera ensuite combinée avec un facteur de gain α pour produire une tension V_{PTAT} qui sera directement proportionnelle à la température ($V_{PTAT} = \alpha.\delta V_{BE}$). Cette tension est comparée à une tension de référence V_{REF} avant d'être convertie en code numérique μ à l'aide du convertisseur analogique-numérique sigma-delta.

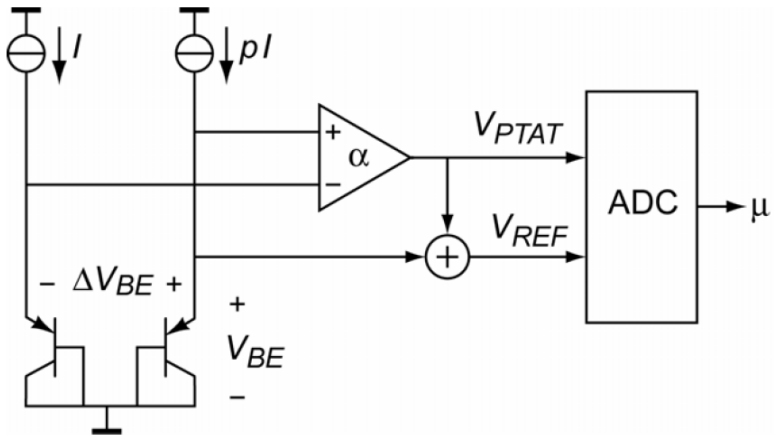


FIGURE 2.7 – Capteur de température à base de transistor bipolaire PNP [Mak10].

Capteur Numérique

En ce qui concerne les structures numériques, le capteur le plus connu est basé sur l'utilisation d'oscillateurs en anneau [RAHP12, FBCP10, MX09]. À titre d'exemple, le capteur de température montré dans la Figure 2.8 est proposé dans [RAHP12]. Le composant basique est un ensemble d'inverseurs. La sortie du dernier inverseur est bouclée à l'entrée du premier. La sortie Q oscille entre 0 et 1 à une fréquence f qui dépend du délai de propagation selon l'équation (2.8), où N est le nombre d'inverseurs et t_d est le délai qui dépend principalement de la tension d'alimentation et de la température. En supposant la tension d'alimentation du circuit constante, une augmentation de la température implique une augmentation du délai et ainsi une réduction dans la fréquence d'oscillation. Des composants supplémentaires sont généralement ajoutés pour activer/désactiver le capteur et pour transformer la fréquence d'oscillation en information logique (par exemple à l'aide d'un compteur).

$$f = \frac{1}{2.\mathrm{N}.t_d} \tag{2.8}$$

Une autre idée de capteur de température consiste à propager un signal à travers une interconnexion relativement longue et à mesurer le délai mis pour traverser une telle structure [DB07]. De plus, les auteurs dans [KX04] ont proposé aussi un capteur de température (4T-Decay) basé sur la relation entre la puissance statique et la température. L'idée principale est de charger des cellules mémoires par un courant statique qui dépend de la puissance statique et ensuite de mesurer le temps de décharge de ces cellules qui va donc dépendre de la température.

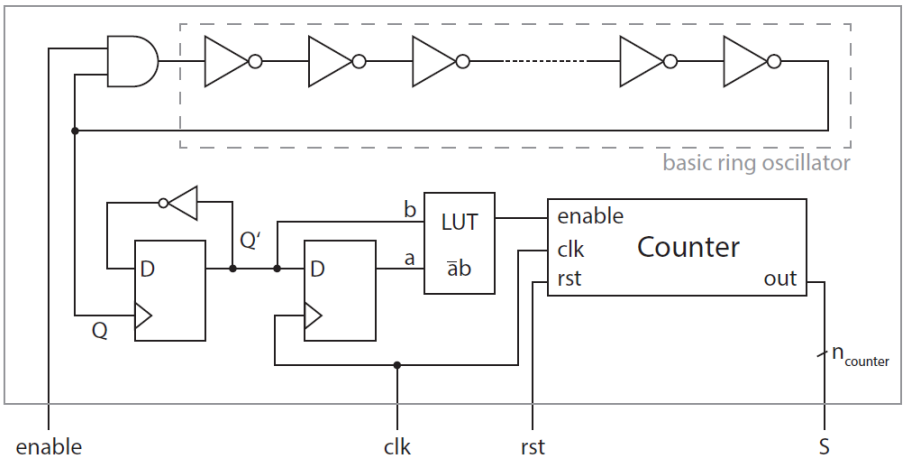


FIGURE 2.8 – Capteur de température à base d'un oscillateur en anneau [RAHP12].

Caractéristiques et limitation

Nous avons présenté quelques structures de capteurs qui permettent la surveillance de la température. Dans cette approche, les trois éléments de la chaîne d'acquisition (capteur, conditionnement et visualisation), correspondent à des modules matériels. Le tableau 2.3 résume les principales caractéristiques de ces capteurs. Nous pouvons constater que la plupart des capteurs sont capables d'observer une variation de température avec une bonne précision (< 3°C) pour un bon compromis entre la taille et la complexité du capteur.

Afin d'avoir une bonne observabilité de la dissipation thermique dans les puces, il est nécessaire d'intégrer plusieurs capteurs en différents points du circuit. Même s'il y a une tendance claire à l'augmentation du nombre de capteurs dans les micro-processeurs, l'augmentation arbitraire impliquera un coût non négligeable. À titre d'exemple, la méthode considérée dans le processeur AMD quad-core Opteron conçue à 65nm SOI CMOS, consiste à intégrer jusqu'à 38 capteurs de type diode dans la puce pour surveiller la température [LMMM08]. En supposant que le capteur a des caractéristiques identiques à celles du capteur proposé dans [SBM+10] (voir tableau 2.3), le coût en silicium occupé par les capteurs serait d'environ 3 mm^2. En outre, la puissance consommée par ces capteurs est aussi d'environ 1mW. En rajoutant la puissance consommée par l'interconnexion et les modules logiques supplémentaires (compteurs, registres comparateurs, *etc.*) qui gèrent la communication à travers ces capteurs, le coût peut atteindre facilement quelques centaines de milliwatts [LMMM08]. C'est la raison pour laquelle le monitoring de la température doit se baser sur un nombre minimal de capteurs pour réduire ce coût.

TABLEAU 2.3 – Comparaison des caractéristiques des capteurs de température existants.

Publication	Type du capteur	Intervalle de Temp.	Précision	Résolution temporelle	Coût en surface / processus de fab.	Coût en puissance/courant
[SGH+07]	Analogique (Diode)	22 → 550°C	0.5-1% (0.2-5°C)	Pas mentionnée	5µm x 5µm /1µm	6µA 50µW à 85°C
[SMKR97]	Analogique (MOS)	0 → 150°C	2°C	2ms	$0.028mm^2$ /1µm	20mW @ 85°C
[SBM+10]	Analogique (BJT)	-70 → 125°C	0.2-0.5°C	Pas mentionnée	$0.1mm^2$ /65nm	9.96µW à 27°C
[MX09]	Numérique (Oscillateur)	-40 → 120°C	<1°C	1sec	$0.0036mm^2$ /130nm	0.09µW à 27°C
[DB07]	Numérique (Propagation)	0 → 200°C	<1.5% (<3°C)	Pas mentionnée	$250\mu m^2$ /45nm	33µW à 80°C
[KX04]	Numérique (4T-Decay)	20 → 150°C	<2% (<3°C)	500µs	$1.7\mu m^2$ /180nm	221µW à 85°C

2.5.3 Estimation indirecte

La deuxième catégorie des méthodes de monitoring consiste à estimer la température en utilisant l'information qui représente la puissance. En fait, la température à un instant donné et dans un point précis du circuit, dépend de la quantité de puissance consommée dans ce point et de l'état thermique à des instants précédents. On parle souvent de prédiction, puisqu'on se base sur l'historique pour prédire la température à des instants futurs.

Dans [SKUY15], la prédiction de la température $T(k+1)$ de chaque ressource de la plateforme ARM Big.LITTLE pendant la période d'ordonnancement suivante $k+1$, est exprimé en fonction des valeurs de la puissance et la température pendant la période k selon l'équation suivante :

$$T(k+1) = A_s \times T(k) + B_s \times P(k) \tag{2.9}$$

où A_s et B_s sont les paramètres déterminés pour chaque ressource à partir des simulations thermiques en utilisant l'outil Hotspot [HGV+06]. Les valeurs de la puissance sont produites par les sondes de courants disponibles dans la plateforme ARM.Big.LITTE.

Une modélisation linéaire de la future température est aussi proposé dans [SAR12] prenant en compte l'historique de la puissance et la température pour la plateforme XScale. Dans cette méthode, la puissance est approximé pour chaque composant du système en utilisant les outils CACTI [CAC11] et Wattch [BTM00]. Une approche différente est présenté dans [WMW09] basé sur l'estimation de la variation de la température $\delta_T(k)$ selon l'équation suivante :

$$\delta_T(k) = A \times \delta_T(k-1) + B \times \delta_f(k) \tag{2.10}$$

où A et B sont constantes. $\delta_T(k-1)$ et $\delta_f(k-1)$ sont les variations de la température et de la fréquence à l'instant précédente $k-1$.

Les méthodes indirectes sont souvent utilisées pour alimenter des méthodes de prise de décision pro-actives qui anticipent des actions afin d'éviter le surchauffe du circuit. L'ensemble de méthodes présenté ci-dessus ne fournit pas une information fiable à grain fin de l'état thermique de la puce, puisque les informations de la puissance sont retrouvées soit à l'aide de quelques capteurs intégrés ou bien en utilisant des modèles issus des simulations. Le coût matériel de ce type de monitoring est essentiellement basé sur le coût du monitoring de la puissance, alors que le coût en performances est relativement faible puisqu'il dépend de la complexité du modèle linéaire (quelques cycles d'horloge pour réaliser le calcul de deux multiplications et d'une addition).

2.5.4 Placement des capteurs de température

Nous avons pu voir que l'approche indirecte n'est pas suffisante pour avoir une information précise du comportement thermique du circuit, alors que l'approche directe permet d'avoir cette information à un coût relativement élevé notamment pour une granularité spatiale fine. À cet effet, le placement des capteurs de température dans le circuit a été abordé dans la littérature afin de réduire ce coût tout en ayant une information précise de la température. Les principales méthodes existantes peuvent être regroupées en 2 catégories : (*i*) uniforme, qui permet de surveiller la température en utilisant une matrice de capteurs, et (*ii*) non-uniforme, où l'insertion de capteurs est basée sur des informations extraites auparavant sur le comportement du circuit.

Uniforme

Il existe plusieurs méthodes dans la littérature qui proposent une approche de quadrillage uniforme [MMNL08, LMMM08, NCR10]. Les auteurs de [LMMM08] proposent une distribution uniforme sous forme de matrice de capteurs. Ils constatent que cette approche n'est pas assez précise et nécessite un grand nombre de capteurs pour avoir un monitoring fiable de la température. Pour cette raison, les auteurs proposent deux autres méthodes montrées dans la Figure 2.9. La première méthode consiste à utiliser un nombre minimal de capteurs et d'estimer la température dans un point en fonction de la température mesurée par les capteurs voisins. Alors que la deuxième solution nécessite un grand nombre de capteurs gérés hiérarchiquement. Tout d'abord, un ensemble de capteurs reste toujours actifs afin de surveiller la température par zone. Ensuite, lorsqu'une augmentation remarquable de la température dans une zone est détectée, les capteurs intégrés dans cette zone sont activés afin d'avoir une vision plus fine du point chaud.

En outre, les auteurs de [NCR10] proposent une méthode récursive de découpage de la surface géométrique de la puce illustrée dans la Figure 2.10. Durant le découpage récursif, l'allocation du capteur peut être réalisée selon deux possibilités : (*i*) le capteur est placé dans le centre géométrique de la zone, ou bien (*ii*) il est placé au barycentre thermique.

Pour récapituler, les méthodes uniformes sont généralement basées sur la décomposition de la surface géométrique de la puce sans aucun recul sur le comportement thermique. Ces méthodes nécessitent l'insertion d'un grand nombre de capteurs pour avoir une vision précise de la dissipation thermique sur toute la puce.

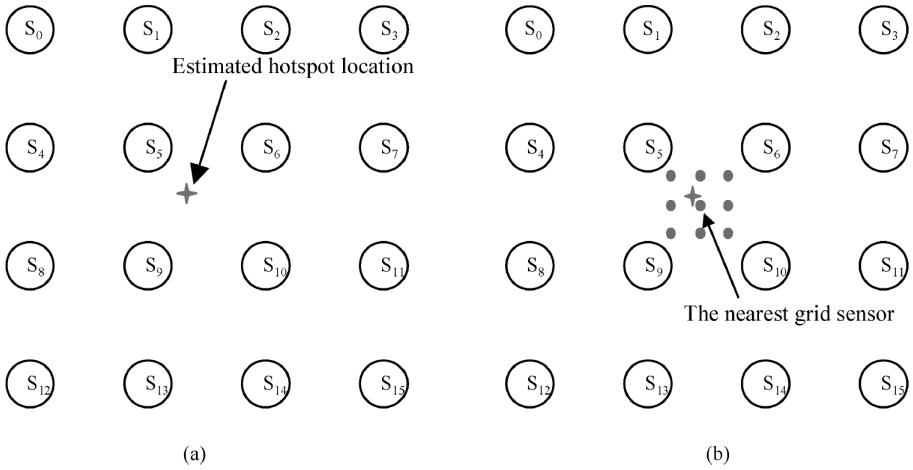


FIGURE 2.9 – Méthode uniforme de monitoring de la température ; (a) interpolation de la température en fonction des capteurs voisins, (b) Gestion hiérarchique de capteurs par activation/désactivation et estimation plus proche en fonction de ces capteurs [LMMM08].

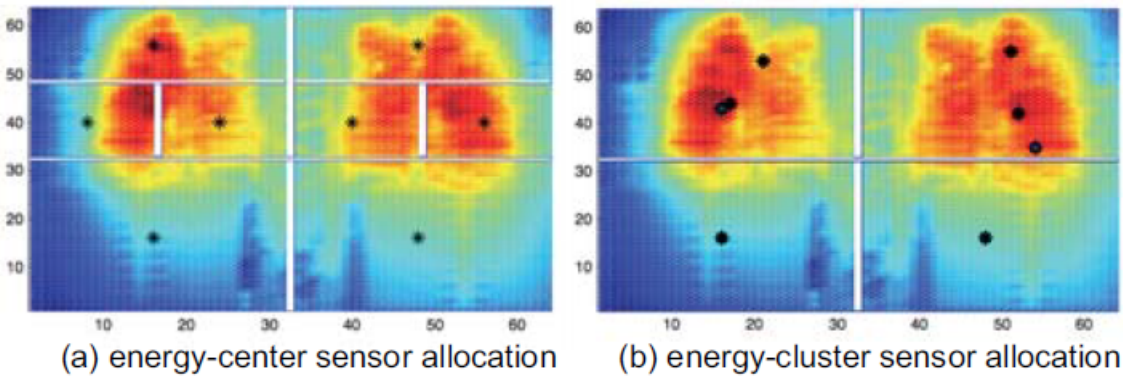


FIGURE 2.10 – Méthode récursive de découpage de la surface géométrique de la puce ; Le capteur est placé : (a) au milieu de la zone, ou bien (b) sur le barycentre thermique [NCR10].

Non-uniforme

L'approche non-uniforme du placement de capteurs est abordée dans plusieurs travaux afin de trouver le bon compromis entre la précision de l'observation et le coût représenté par le nombre de capteurs intégrés sur puce. Dans [LSH05], les auteurs présentent une méthode pour le monitoring de la variation maximale de la température en proposant un modèle analytique pour le placement des capteurs. Ce modèle est montré dans l'équation (2.11), où $T_{src}(r)$ est la température dans un point à une distance r du point chaud à surveiller ($T_{src-max}$). La constante k dépend de l'épaisseur de la puce, de la résistivité thermique de l'interface matérielle et des caractéristiques du silicium.

$$T_{src}(r) = T_{src-max} \times (1 - e^{\frac{-2r}{k}}) \tag{2.11}$$

Dans cette étude, les informations liées à la température maximale dans un point chaud et l'emplacement du capteur sont utilisés pour déterminer la marge de sécurité pour le placement du capteur, et aussi pour déterminer le nombre optimal de capteurs nécessaires à la surveillance de tous les points chauds. Nous pouvons constater que la différence de la température entre le point le plus chaud et l'emplacement du capteur augmente lorsque la distance de la source augmente, et donc la précision des mesures décroît. À cet effet, dans [SR10], les auteurs créent des zones d'observabilité pour chaque point chaud où la différence de température est toujours inférieure à une valeur maximale tolérable. Ensuite, ils cherchent un emplacement optimal des capteurs basé sur cette information en tenant compte du capteur dans la zone d'observabilité de chaque point chaud. Cette méthode peut être efficace pour surveiller quelques points chauds dans la puce, mais reste limitée pour la surveillance de tout le profil thermique de la puce. En outre, dans la réalité plusieurs sources d'échauffement peuvent être présentes en même temps affectant indépendamment la température de la puce.

Une autre méthode pour le placement de capteurs pour le diagnostic des fautes dans les circuits intégrés est proposée dans [BK01]. La méthode consiste à relier un capteur de température à chaque module de puissance tel qu'illustré dans la Figure 2.11. Cependant, les auteurs de [LSH05] démontrent que les zones chaudes se déplacent en fonction des applications exécutées par le processeur.

D'autres approches utilisent les algorithmes d'apprentissage pour extraire des informations pertinentes pour un placement optimale des capteurs. Les auteurs de [MMNL08] présentent une méthode basée sur l'algorithme *k-moyenne*. L'idée principale est de regrouper l'ensemble des nœuds thermiques en k sous-ensembles contenant chacun un ensemble de nœuds. Le capteur est ensuite placé dans le barycentre du sous-ensemble.

En général, le comportement thermique varie d'une application à une autre, notamment

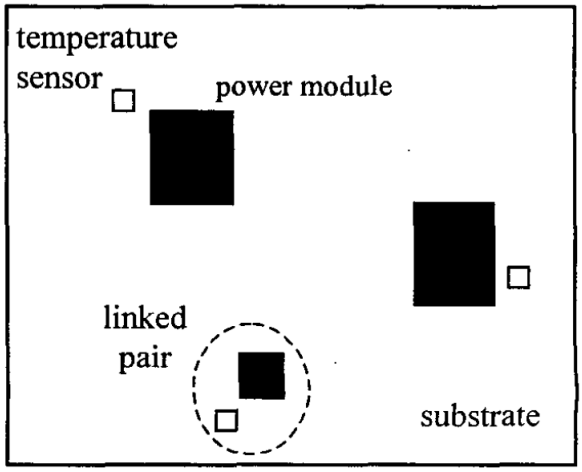


FIGURE 2.11 – Monitoring de la température par module [BK01].

les endroits des points chauds du circuit [MMNL08]. Les auteurs de [LRLZ13] ont cherché à combler cette lacune en proposant un double regroupement (dual clustering). Leur méthode consiste à utiliser la segmentation *k-moyenne* sur plusieurs données thermiques extraites de plusieurs applications. Un deuxième regroupement spatial est basé sur le diagramme de *Voronoi* [Bhattacharya et Gavrilova 2007] montré, à titre d'exemple, dans la Figure 2.12. Les zones qui contiennent des points chauds (par ex : H1 et H2) et qui partagent une interface unique (*Voronoi edge*) sont considérées comme zones voisines. Ensuite, l'idée principale de la segmentation est de fusionner les zones voisines dont la distance qui sépare leurs points chauds est inférieure à un seuil prédéfini.

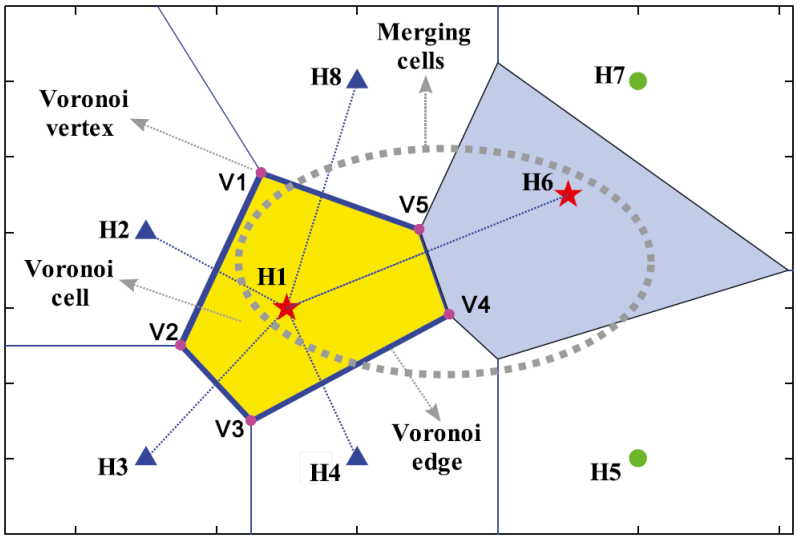


FIGURE 2.12 – Diagramme de *Voronoi* pour le regroupement des points chauds [LRLZ13].

Une autre approche pour le placement des capteurs est présentée dans [RCN11]. Les auteurs proposent un algorithme itératif constitué de deux phases. L'objectif de la première phase est de trouver une allocation initiale basée sur la distribution des capteurs sur les points chauds

prenant en compte des zones avec des variations de température indépendantes. La deuxième phase consiste ensuite à optimiser l'allocation de chaque capteur par rapport aux autres capteurs.

2.5.5 Bilan

Dans cette section, nous avons présenté des méthodes qui permettent d'observer en ligne la température. Les mesures directes à l'aide de capteurs de température sont souvent plus précisent que l'approche d'estimation indirecte. Nous avons aussi abordé la problématique de placement de capteurs dans le circuit afin d'avoir une vision spatiale de la dissipation thermique. Les différentes méthodes de placement de capteurs de température sont regroupées en deux catégories principales : (*i*) uniforme, et (*ii*) non-uniforme.

Le tableau 2.4 compare ces méthodes de placement de capteurs. Nous pouvons conclure que l'erreur des méthodes uniformes est plus élevée que l'approche non-uniforme et nécessite aussi un grand nombre de capteurs. Nous avons pu voir qu'un placement statique des capteurs n'est pas suffisant pour le monitoring du profil thermique. En outre, le placement de capteurs basé sur la décomposition uniforme de la surface de la puce peut répondre à ce problème. Cependant, cette approche nécessite l'insertion d'un grand nombre de capteurs pour avoir une estimation précise de la température dans n'importe quel endroit de la puce. D'autre part, les méthodes non-uniformes proposent une insertion de capteurs basée sur l'étude du comportement thermique des différentes zones de la puce. Cette méthode est la moins couteuse ; elle permet un monitoring spatial de la température, mais nécessite une phase d'apprentissage hors-ligne. Cet apprentissage est généralement conduit sur des données extraites à partir des simulations au moment de la conception du circuit. Dans cette thèse, nous proposons un ensemble d'outils qui nous permettent de réaliser des simulations thermiques du circuit, ce afin de réaliser les analyses de données. Nous proposons aussi une méthode systématique pour le placement des capteurs de température dans chaque zone de la puce.

TABLEAU 2.4 – Comparaison des différentes méthodes existantes pour le placement des capteurs de température sur puce.

Publication	**Approche**	**Plateforme**	**Nombre de capteurs**	**précision**	**type de surveillance**
[SR10]	Non-uniforme	MPcore	$4 \rightarrow 7$	$7 \rightarrow 1$ °C	8 points d'intérêt.
[NCR10]	Uniforme	Dual-core	$4 \rightarrow 32$	$32 \rightarrow 13\%$	Cartographie thermique complète.
[MMNL08]	Non-uniforme (K-moyenne)	Dual-core	$1 \rightarrow 6$	$7 \rightarrow 1.5$°C	Cartographie thermique complète.
[RCN11]	Non-uniforme (itérative)	Dual-core	$1 \rightarrow 6$	$6 \rightarrow 0.6$°C	Cartographie thermique complète.
[RCN11]	Uniforme (matrice)	Dual-core	$1 \rightarrow 6$	$10 \rightarrow 4$°C	Cartographie thermique complète.

2.6 Conclusion

Dans ce chapitre, nous avons abordé, dans un premier temps, les politiques de prises de décisions avec l'ensemble d'actionneurs existant dans les systèmes auto-adaptatifs. Ce travail nous a aidés à mieux appréhender les spécificités du contrôle des systèmes embarqués, ce afin de définir les principaux caractéristiques d'une méthode de monitoring efficace.

Dans un second temps, nous nous sommes intéressés aux méthodes permettant de surveiller la puissance consommée par les puces. Nous avons commencé à présenter les méthodes directes de mesure à l'aide des capteurs de puissance. Nous avons ensuite fait un tour d'horizon des méthodes indirectes qui observent les paramètres de fonctionnement du système et l'activité qui correspond à l'utilisation des ressources pour estimer la puissance. Nous avons pu remarquer que les mesures directes permettent une observation précise de la puissance à une résolution temporelle très fine. Les méthodes indirectes sont moins coûteuses moins précises à une résolution temporelle moins élevée. Nous avons pu constater aussi que les méthodes indirectes existantes ne prennent pas en compte tous les paramètres qui affectent la puissance.

Nous nous sommes intéressés par la suite aux méthodes de surveillance de la température dans les puces. Nous avons d'abord présenté l'ensemble des capteurs de températures existants. Ensuite, nous avons expliqué l'ensemble des méthodes de placement des capteurs de température dans la puce afin d'avoir une vision globale sur la dissipation thermique. Nous avons conclu que le placement statique des capteurs ne permet pas une observation du profil thermique. Tandis que, le placement de capteurs basé sur la décomposition uniforme de la surface de la puce est très couteux. En outre, les méthodes non-uniformes qui proposent une insertion basée sur l'extraction de la connaissance du comportement thermique sont moins coûteuses, mais nécessitent une phase d'apprentissage au moment de la conception du circuit.

CHAPITRE 3

NOUVELLE MÉTHODE DE MONITORING DE LA PUISSANCE DES SOCS AU NIVEAU RTL

« Si vous ne pouvez pas faire de grandes choses, faites de petites choses de façon grandioses »

Napoleon Hill

Sommaire

3.1 Introduction

Nous avons pu voir dans le chapitre précédent que l'information liée à la puissance consommée par le circuit est indispensable dans plusieurs méthodes d'optimisation, notamment les pro-actives qui essaient d'anticiper des actions pour éviter des états indésirables tels que le dépassement d'un seuil thermique. De plus, le monitoring de la puissance à une résolution spatiale et temporelle fine permet de mettre en place des méthodes d'adaptation à grain fin efficaces. Nous citons, par exemple, la méthode de gestion de tâches introduite dans [ZXD+08], qui est basée sur la prédiction de la température par cœur à partir des estimations de la puissance à une résolution temporelle fine (chaque 8 ms) afin d'éviter les points chauds dans les plateformes multi-cœurs.

Les méthodes standard de monitoring de la puissance sont généralement basées sur des mesures directes à l'aide de capteurs de courant et de tension. En plus de la précision élevée des mesures, l'avantage de ces méthodes réside dans le fait que ces mesures sont effectuées à une résolution temporelle fine. Cependant, la résolution spatiale est faible puisque très peu de capteurs peuvent être intégrés dans la puce à cause de leurs coûts élevés. Ainsi, la deuxième solution repose sur l'estimation indirecte de la puissance en fonction des paramètres technologiques et de fonctionnement du circuit. Cette solution est généralement moins précise, mais aussi moins coûteuse, permettant ainsi un monitoring à une résolution spatiale plus importante.

En effet, la puissance consommée dans un circuit numérique est due à l'activité correspondant à la charge et la décharge des transistors (puissance dynamique), et aux courants de court-circuit et de fuite (puissance statique) qui dépendent de plusieurs paramètres tels que la température de la jonction, le procédé technologique, le vieillissement *etc.* L'activité interne est une des variables qui varie en fonction des applications dans les SoCs et qui affectent directement consommation. Elle correspond, au niveau physique, au nombre de transistors qui commutent à chaque cycle d'horloge. Suivant les tâches à réaliser et les données manipulées, cette activité peut varier et entraine donc des changements significatifs de la consommation. C'est la raison pour laquelle le monitoring de l'activité d'un système est l'un des éléments importants qui entre dans l'estimation précise de la puissance. Nous avons pu voir dans le chapitre précédent que la plupart des méthodes négligent l'impact de l'activité sur les variations de la puissance, alors que la seule méthode présenté dans [MBTC13] propose une solution systématique mais très limité pour le monitoring dans les circuits complexes.

Ce chapitre présente une nouvelle approche de monitoring de la puissance pour satisfaire les trois critères suivants : (*i*) une précision élevée, (*ii*) une résolution fine, et (*iii*) un coût de monitoring faible. La puissance est estimée en fonction de l'activité au niveau matériel, obtenue à partir d'un ensemble de méthodes d'analyse de données issues du flot de conception. La

technique envisagée par notre méthode est illustrée dans la Figure 3.1. La méthode consiste à intégrer des compteurs d'évènements "*Events Counter*" (EC) dans le circuit. L'unité d'adaptation collecte périodiquement les informations des compteurs, et à partir d'un modèle, estime en ligne la puissance consommée par le circuit. Afin de construire un sous-système de monitoring précis et peu coûteux, il faut donc d'une part, sélectionner un sous-ensemble de signaux stratégiques (qui déterminera le nombre de compteurs à intégrer) pour réduire le coût et d'autre part, définir un modèle qui produit des estimations précises de la puissance.

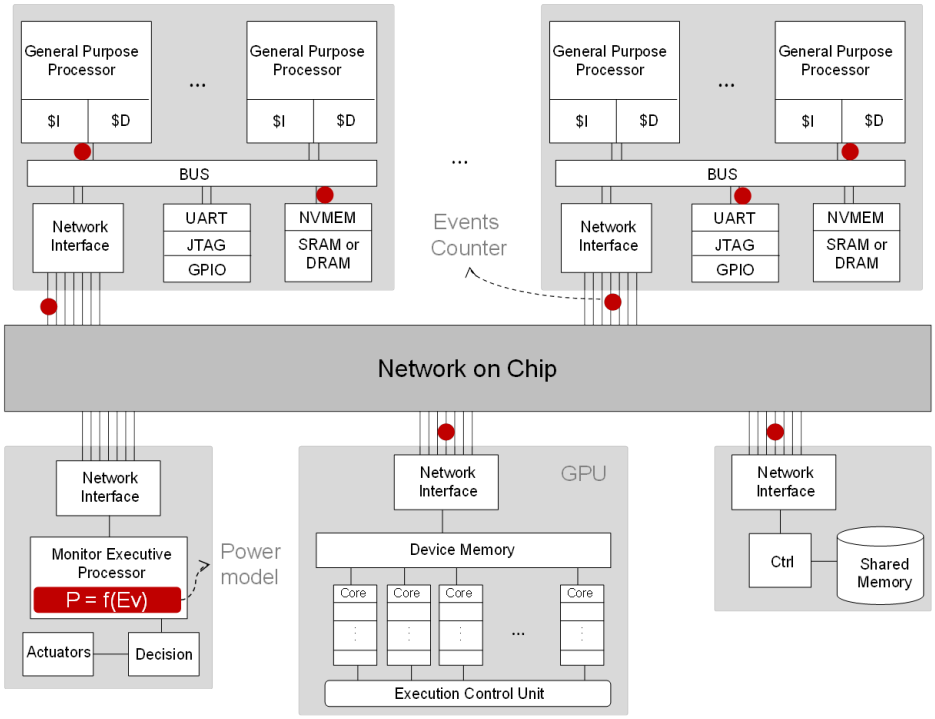


FIGURE 3.1 – Exemple de monitoring de la puissance dynamique d'un système hétérogène multi-cœur.

Dans ce qui suit, nous commencerons par présenter une méthodologie générique qui permet d'identifier l'information pertinente pour la modélisation de la puissance. Puis, nous proposerons une instrumentation complète du flot de conception dans le but d'extraire les données à différents niveaux d'abstraction. Ensuite, nous nous pencherons sur l'explication des différentes méthodes de sélection et les modèles utilisés pour construire notre sous-système de monitoring. Nous clôturerons ce chapitre avec un cas d'étude pour valider notre approche et évaluer son efficacité.

3.2 Méthodologie : vue d'ensemble et généralités

En général, l'information précise qui caractérise l'activité d'un système est représentée par la commutation des transistors. Nous cherchons à caractériser cette activité à un niveau d'abstraction élevé (au niveau transfert de registre) pour mettre en place une méthode de monitoring

générique. Pour ce faire, nous proposons une analyse des données extraites au moment de la conception du circuit. L'idée principale est basée sur l'analyse des données issus des différents niveaux d'abstraction du flot de conception, et sur l'utilisation des méthodes de fouille de données pour la sélection des signaux stratégiques qui reflètent l'information pertinente pour une modélisation précise de la puissance. En effet, la quantité de données extraites à partir du flot de conception s'accroit avec la complexité des systèmes intégrés. Un système intégré moderne tel que la Samsung Exynos 7420 intègre jusqu'à 8 cœurs ARM avec des bus d'interconnexions de type AMBA avec une largeur du bus de données de 64 bits [DT14]. Un simple calcul nous permet de constater que l'on est face à un grand nombre de signaux potentiels (supérieur à 1000 au niveau "*Register Transfer Level*" (RTL)). Ce nombre peut atteindre une centaine de milliers dans les systèmes de calcul haute performance "*High Performance Computing*" (HPC) qui connectent des centaines d'unités de calculs. La recherche dans cette quantité de données devient ainsi une difficulté majeure. C'est la raison pour laquelle, nous avons introduit l'utilisation des méthodes de fouille de données dans notre analyse.

Une telle méthode repose sur la définition des éléments suivants :

- N_s et T sont le nombre total de signaux internes au niveau RTL d'un circuit et le temps de simulation.
- $E\nu_i(t)$ est l'occurrence de l'évènement de commutation sur un signal S_i. $E\nu$ est donc l'ensemble des évènements disponibles, $E\nu = \{E\nu_i\}_{i=1:N_s}$.
- $P_{dyn}(t)$ est la puissance dynamique instantanée.

On cherche à modéliser une caractéristique du système (la puissance) qui est une variable de l'état (défini par plusieurs variables) : pour cela on s'appuie sur les méthodes supervisées de fouille de données. La base de données à construire doit contenir : (*i*) les évènements de commutation au niveau RTL ($E\nu$), et (*ii*) la puissance dynamique instantanée (P_{dyn}). En temps de conception, il faut des applications de tests simulées qui permettent de faire varier l'activité interne du circuit et d'estimer la consommation dynamique de puissance afin de construire cette base de données. Le flot de conception/simulation est illustré dans la Figure 3.2 ; il permet de collecter ces informations pour un circuit donné en se basant sur un ensemble d'outils classiques et des bibliothèques technologiques.

La première phase de la construction de la base de données consiste à retrouver la puissance dynamique instantanée $P_{dyn}(t)$. Il existe plusieurs outils qui produisent ces estimations, par exemple PrimeTime-PX de Cadence [SYN15]. Ces outils nécessitent des simulations de la description du circuit après placement et routage afin de produire des estimations précises de la puissance consommée. À cet effet, le premier processus du flot consiste à synthétiser, placer et router le circuit décrit en VHDL ou Verilog pour une technologie donnée pour générer la *netlist* après placement et routage. Une simulation de cette *netlist* sera effectuée par des outils de

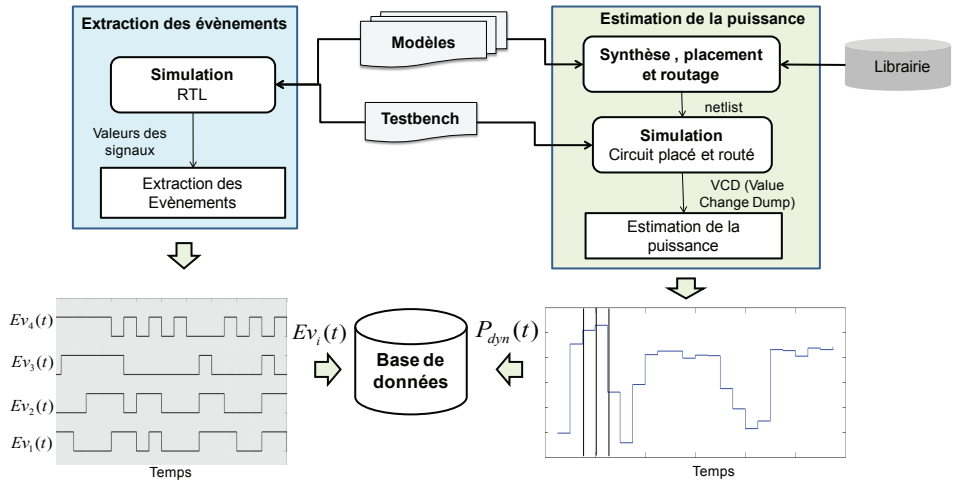


FIGURE 3.2 – Méthode générale de génération de la base de données par les outils de conceptions des circuits.

simulation tels que ModelSim qui produit généralement un fichier contenant les informations de transition au niveau portes logiques (VCD). Ce fichier est ensuite utilisé par l'outil d'estimation de puissance afin de produire les valeurs de P_{dyn} qui correspondent dans tous les cas à une suite d'estimations de la puissance moyenne consommée pendant des intervalles de temps réguliers t_i.

La deuxième phase consiste à extraire les activités de commutation de la simulation du circuit au niveau RTL. Cette simulation est effectivement plus rapide que celle après placement et routage, en utilisant le même *testbench* ainsi que la durée de simulation T dans les deux cas pour construire une base de données représentative. Celle-ci est représentée dans la Figure 3.3. Chaque ligne correspond à une instance et contient le nombre de commutations (passage de 0 à 1 ou de 1 à 0) sur tous les signaux internes $\{S_1...S_{Ns}\}$ du circuit décrit au niveau RTL avec la puissance estimée correspondant $P_{dyn}(t_i)$ pendant l'intervalle de temps t_i.

Une fois que la base de données est préparée, la méthode d'analyse de données consiste en premier lieu à filtrer les signaux qui ne contribuent pas aux variations de la puissance. Ensuite, une phase de sélection de signaux stratégiques permet de définir l'ensemble de signaux à utiliser pour placer les compteurs d'évènements. Ces deux phases sont réalisées à l'aide des méthodes de fouille de données pour la sélection des attributs qui sont généralement regroupés en deux approches [BL97] : (*i*) *Filter* qui utilise des critères statistiques tels que la corrélation, ou bien (*ii*) *Wrapper* qui prend en compte la dépendance d'un attribut par rapport à un modèle spécifique. Dans un premier temps, nous allons éliminer les attributs redondants en nous basant sur l'évaluation selon l'approche *filter*. Ensuite, nous introduirons l'approche *wrapper* dans la section 3.4 afin de sélectionner les signaux stratégiques. Enfin, une modélisation en fonction de l'activité de commutation sur les signaux sélectionnés en utilisant les méthodes de classification permettra d'estimer la puissance consommée.

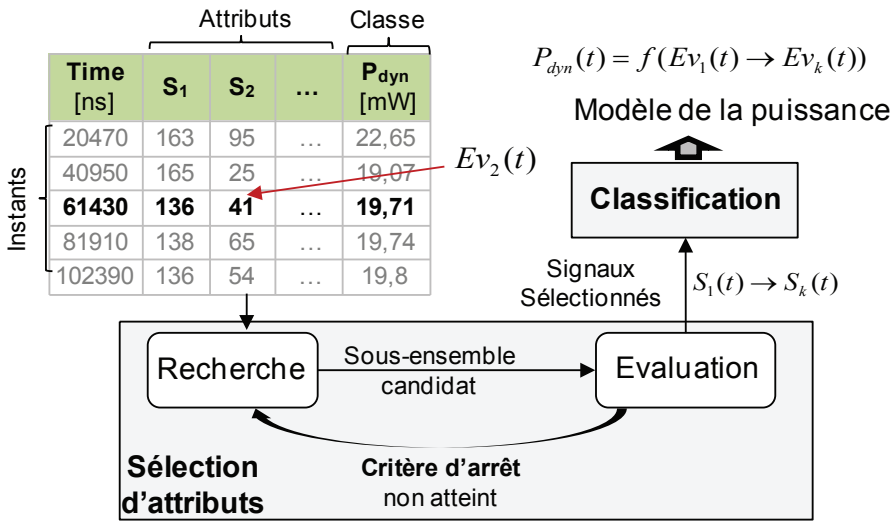


FIGURE 3.3 – Illustration de la méthode générale de sélection de signaux stratégiques et de la modélisation de la puissance.

3.3 Filtrage des attributs redondants

Les bases de données obtenues par le biais du flot précédemment décrit contiennent des centaines (voire des milliers) d'attributs et un certain nombre d'instances (t_i). Le traitement statistique classique consiste à calculer la corrélation qui donne une mesure synthétique de l'intensité de la relation entre un attribut S_i et la puissance P_{dyn}. Généralement, la mesure de la corrélation linéaire entre deux variables est effectuée avec le coefficient de corrélation de Bravais-Pearson, au besoin en utilisant un changement de variable si la liaison n'est pas linéaire. Ce calcul ne constitue qu'une première étape dans l'analyse de la relation entre les attributs et la puissance. Il ne renseigne pas le degré de significativité d'une relation car celle-ci dépend également du nombre d'instances t_i. Pour cela, il faut procéder à un test d'hypothèse pour déterminer si une relation est significative. En général, on fixe l'hypothèse H_0 : il n'y a pas de relation entre les deux variables. Ensuite, on calcule la probabilité que cette hypothèse soit vraie (notée *p-value*). À chaque calcul de corrélation entre S_i et P_{dyn}, on retrouve la valeur de *p-value*. Si cette valeur est inférieure à un seuil α (généralement 5%) l'hypothèse est rejetée. La relation entre S_i et P_{dyn} est ainsi considérée comme significative.

Dans notre cas, on peut trouver un grand nombre de signaux S_i qui ont des corrélations significatives plus ou moins fortes. Une simple analyse statistique ne permet pas de tirer des conclusions pour sélectionner quelques signaux stratégiques. C'est pour cela que l'utilisation des algorithmes de fouille de données peut présenter une solution intéressante, notamment les méthodes de sélection d'attributs qui automatisent efficacement les techniques d'analyses statistiques avec l'aspect algorithmique de l'informatique. C'est un ensemble d'algorithmes qui cherchent dans l'espace d'état à sélectionner k attributs parmi les N_s attributs de départ. Une

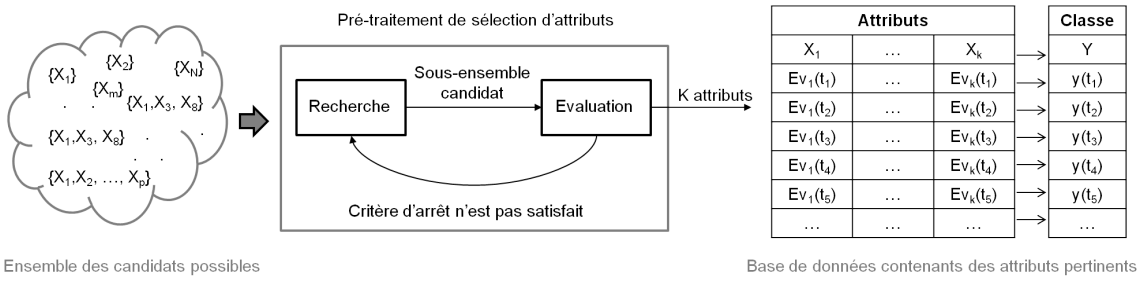


FIGURE 3.4 – Méthode de sélection des attributs.

méthode de sélection est principalement constituée d'une procédure de recherche et d'une procédure d'évaluation (Figure 3.4).

3.3.1 Procédure d'évaluation *Filter*

L'évaluation selon l'approche *filter* se base essentiellement sur des dépendances statistiques telles que la corrélation pour évaluer un sous-ensemble d'attributs. Dans la littérature, il existe plusieurs méthodes d'évaluation, mais la plupart d'entre elles opèrent sur des données nominales et binaires telles que Relief-F, one-R, Chi-squared, *etc.* L'algorithme "*Correlation Feature Selection*" (CFS) [Hal98] est connu dans l'apprentissage automatique, et propose une heuristique qui traite les données numériques [ZLSC10, PHW+08]. Le principe de cet algorithme est basé sur l'hypothèse suivante : un bon sous-ensemble contient des attributs fortement corrélés avec la classe et faiblement corrélés entre eux. L'équation (3.1) illustre cette heuristique qui évalue un candidat (ensemble d'attributs), où p est le nombre total d'attributs dans le sous-ensemble candidat, $\bar{R}_{ac}$ est la moyenne des coefficients de corrélations calculés entre chaque attribut du candidat et la classe, et $\bar{R}_{aa}$ est la moyenne des coefficients de corrélations calculés entre tous les attributs du candidat. Le candidat ayant la plus grande valeur de mérite est celui à considérer comme solution.

$$M = \frac{p \times \bar{R}_{ac}}{\sqrt{p + p \times (p-1) \times \bar{R}_{aa}}} \tag{3.1}$$

3.3.2 Procédure de recherche

Le candidat est généralement choisi par un algorithme de recherche associé à la procédure d'évaluation. La méthode de recherche exhaustive tend à rechercher la solution optimale en évaluant tous les cas possibles en considérant 2^{N_s} candidats pour N_s signaux. Par ailleurs, il existe les algorithmes itératifs qui cherchent une solution satisfaisante en réduisant largement le nombre de candidats. Tous ces algorithmes construisent leur solution de manière itérative :

(*i*) progressive en partant d'un ensemble vide et en rajoutant un attribut à chaque itération, (*ii*) ou bien dégressive en partant de l'ensemble total et en éliminant un attribut à chaque itération.

```
   input  : X ensemble d'attributs et eval() procédure d'évaluation
   output : S sous-ensemble d'attributs sélectionnés
 1 S ← {} ; C ← X ; best_eval = 0 ;                          // Initialisation des variables
 2 while 1 do
 3     /* Itérations de l'algorithme                                                    */
 4     best_x ← {} ;                                  // Initialisation du variable de la boucle
 5     for x in C do
 6         subset_candidate ← S ∪ {x} ;                          // Sous-ensemble candidate
 7         nouvelle_eval = eval(subset_candidate) ;               // Évaluation du candidat
 8         /* Recherche du meilleur attribut                                           */
 9         if nouvelle_eval > best_eval then
10             best_eval = nouvelle_eval ;
11             best_x ← {x} ;
12         end
13     end
14     /* Condition d'arrêt                                                            */
15     if best_x != {} then
16         S ← S ∪ {x} ;                          // L'attribut est rajouté à la solution S
17         C ← C \ {x} ;
18     else
19         return S ;                                                // Fin de la recherche
20     end
21 end
```

Algorithm 1: Illustration de l'approche progressive de l'algorithme *Greedy Stepwise* [CF94]

En général, les méthodes itératives comme le Greedy Stepwise [WM14] se basent sur un algorithme glouton (*Greedy*) dont l'approche progressive est illustrée dans l'Algorithme 1. En partant de l'ensemble d'attributs X qui contient N_s éléments (signaux dans notre cas) et une fonction d'évaluation *eval* (par exemple le CFS), il construit d'une manière itérative un candidat à évaluer afin de trouver une solution satisfaisante. La solution S est d'abord considérée vide et l'ensemble de départ C est égal à l'ensemble des attributs X en entrée de l'algorithme. À chaque itération, il évalue l'union de la solution S et chaque attribut x restant dans C. Il choisit ensuite ce qui produit la meilleure évaluation, et met à jour la solution S et l'ensemble C. La recherche se termine lorsqu'aucun des attributs restants dans C n'améliore le mérite de la solution.

Il existe une deuxième approche de recherche itérative, nommée *BestFirst* [XYC88], qui autorise un retour en arrière (*backtracking*) selon un paramètre prédéfini. Celui-ci permet de remplacer les attributs sélectionnés dans S qui n'ont qu'un faible impact, par d'autres plus pertinents. Après un certain temps, le *BestFirst* tend à évaluer tous les cas possibles. À cet effet, plusieurs critères d'arrêt permettent d'assurer la convergence de cet algorithme (nombre de *backtracking*). Une optimisation du temps de recherche de l'algorithme *BestFirst* est aussi pro-

posée dans l'algorithme *LinearForwardSelection* [GFHK09] qui spécifie au départ la taille de la solution S en nombre d'attributs.

L'algorithme génétique est une recherche méta-heuristique basée sur le principe de sélection des gènes en biologie [PeP+95]. Il crée une population de solutions formée de sous-ensembles d'attributs, où la solution est recherchée en parallèle dans toute la population qui évolue au cours du temps pour converger vers une solution optimale. À chaque génération de ce processus itératif, une nouvelle population est générée après un ensemble d'opérations telles que la mutation et la permutation. Pendant la mutation, un échange d'attributs est effectué entre les éléments de la population, lorsque la permutation fusionne deux éléments de la solution en groupant les meilleurs attributs selon une procédure d'évaluation afin de créer deux nouveaux sous-ensembles. Les meilleurs attributs ont une plus grande chance d'être présents dans la nouvelle population. C'est dans ce sens que l'on dit que la population évolue au cours du temps.

3.3.3 Bilan

Les deux procédures présentées ci-dessus constituent une méthode de sélection qui permet de filtrer les attributs redondants de la base de données. Ceci va permettre dans notre cas d'éliminer les signaux qui n'apportent pas assez d'information à la modélisation de la puissance. La complexité de cette méthode dépend essentiellement de la procédure de recherche, puisque le temps d'évaluation d'un candidat par CFS est négligeable (il est égal au temps de calcul de l'équation du mérite (3.1)). Ainsi, le choix de la procédure de recherche va dépendre du temps d'exécution et de la qualité de la solution retrouvée selon cette métrique d'évaluation. Nous présenterons ultérieurement dans la section 3.6 une évaluation détaillée de toutes ces procédures sur un cas d'étude qui nous permettra de tirer des conclusions sur le choix de l'algorithme pour la résolution de notre problème.

3.4 Sélection des signaux stratégiques et Modélisation de la puissance

La méthode de sélection selon l'approche *filter* permet un filtrage des signaux redondants. En d'autres termes, les signaux supprimés n'apportent pas suffisamment d'information pour expliquer la variation de la puissance. Un modèle de la puissance en fonction des évènements sur les signaux sélectionnés peut être ainsi construit. Toutefois, les signaux sélectionnés n'ont pas tous un impact significatif sur la précision du modèle. La procédure d'évaluation *wrapper* permet de mettre cela en évidence. L'ensemble de modèles de puissance sera aussi présenté

dans cette section.

3.4.1 Sélection par approche *Wrapper*

La sélection *wrapper* est représentée dans la Figure 3.5. La méthode consiste à utiliser un algorithme de recherche parmi ceux présentés ci-dessus avec une procédure d'évaluation telle que le "*Classifier Subset Evaluation*" (CSE) [DL97] qui se base sur la rétroaction du modèle de la puissance pour évaluer un sous-ensemble d'attributs. Le CSE calcule l'erreur initiale en considérant l'ensemble de départ avec tous les attributs. Il évalue ensuite les candidats par la différence d'erreur introduite par la suppression d'un attribut de cet ensemble. On peut ainsi définir un critère d'arrêt qui peut être, soit un seuil d'erreur à ne pas dépasser, soit un nombre donné d'attributs. À ce stade, c'est au concepteur de fixer le meilleur compromis entre la précision et le nombre de signaux à utiliser.

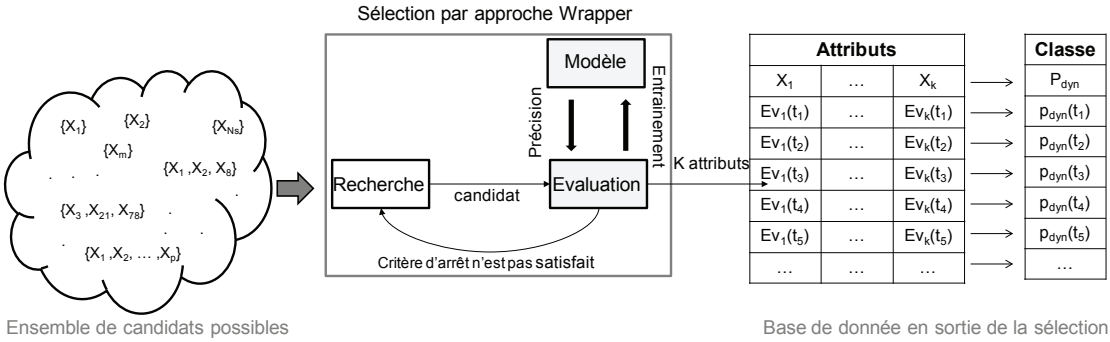


FIGURE 3.5 – Méthode de sélection des attributs selon l'approche *wrapper*.

3.4.2 Modélisation de la puissance

Dans cette partie, la modélisation de la puissance dynamique se base seulement sur l'activité de commutation au niveau RTL en supposant constants tous les autres paramètres du circuit. Une fois que les signaux stratégiques qui produisent une modélisation précise de la puissance ont été retrouvés, les modèles : linéaire, réseau de neurones et "*Multivariate adaptive regression splines*" (MARS), peuvent être entrainés sur une base de données pour qu'ils soient implémentés ultérieurement en ligne. Ensuite, il est question de la précision et de la complexité qui déterminent le choix d'un des trois modèles.

Soit k le nombre de signaux sélectionnés par les méthodes de sélection. Les trois modèles introduits pour la modélisation de la puissance sont présentés en détail dans les sections suivantes.

Modèle linéaire

Ce modèle est obtenu par régression en exprimant la valeur de $\hat{P}_{dyn}$ en fonction des évènements sur les k signaux sélectionnés selon l'équation (3.2). Les c_i sont des coefficients obtenus à partir de la base de données d'apprentissage. c_0 correspond à la composante constante du modèle, et c_i représente le poids attribué à chaque attribut i, avec $i = \{1..k\}$. Ces coefficients sont obtenus systématiquement par la méthode des moindres carrés qui consiste à ajuster les c_i pour minimiser le carré de l'erreur entre l'estimation $\hat{P}_{dyn}$ et la valeur réelle P_{dyn}.

$$\hat{P}_{dyn} = c_0 + \sum_{i=1}^{k} (c_i \times Ev_i) \tag{3.2}$$

Régressions multivariées par spline adaptative : MARS

MARS [Fri91] est une technique de régression non paramétrique pouvant être vue comme une extension des régressions linéaires qui modélisent automatiquement des interactions et des non-linéarités. L'équation (3.3) illustre la construction de ce modèle qui exprime $\hat{P}_{dyn}$ comme une somme pondérée de fonctions de bases B_j (voir équation (3.4)). Une fonction de base est associée à un évènement Ev_i parmi l'ensemble des évènements collectés sur les k signaux. Le choix du nombre M de fonctions est un paramètre spécifié généralement par espionnage, permettant d'ajuster la précision du modèle mais aussi sa complexité. En outre, les α_j et les c_j sont les paramètres ajustés par la méthode des moindres carrés pour avoir des estimations précises.

$$\hat{P}_{dyn} = \alpha_0 + \sum_{j=1}^{M} (\alpha_j \times B_j(Ev_i)) \tag{3.3}$$

$$B_j(Ev_i) = max(0, c_j - Ev_i), i \in \{1..k\} \tag{3.4}$$

Réseaux de neurones

Les modèles à base de réseaux de neurones, inspirés du fonctionnement des neurones biologiques humains, sont largement connus dans le domaine de la fouille de données pour modéliser les interactions non linéaires. On s'intéresse dans la classification supervisée aux réseaux non bouclés où la configuration la plus classique est appelée *perceptron multicouche*. Dans cette architecture, les neurones sont organisés en couches comme le montre la Figure 3.6 : une couche intermédiaire entre les entrées et les sorties appelée couche cachée et un neurone (ou une couche de neurones) en sortie. Les connexions se font d'une couche à la suivante sans qu'il

y ait de connexion entre les couches non adjacentes. Cette architecture est également appelée réseau à deux couches puisqu'il y a deux couches de poids ajustables : celle qui relie les entrées aux neurones cachés et celle qui relie les neurones cachés au neurone de sortie. Une fois que l'architecture est choisie, il faut fixer le nombre de neurones cachés M. Plus ce nombre est élevé, plus la fonction modélisée par le réseau de neurones peut être complexe.

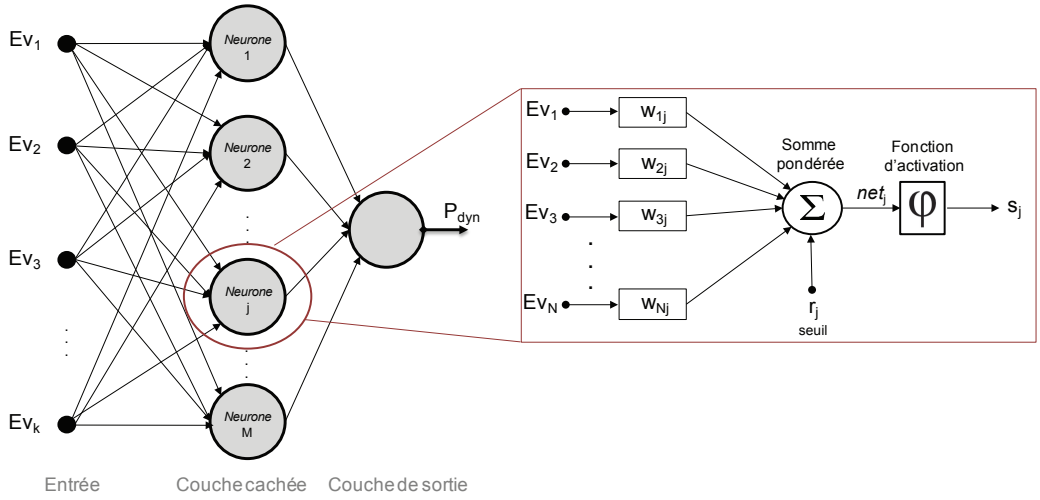


FIGURE 3.6 – Modèle de puissance à base de réseau de neurones *perceptron* multicouche.

Un neurone est l'association de deux opérations : une linéaire qui réalise une somme pondérée des entrées multipliées par des coefficients et la deuxième non-linéaire qui active la sortie du neurone. Pour mieux comprendre le modèle, on considère le neurone j dans la Figure 3.6 qui tout d'abord associe à ses entrées (a_i dans l'équation 3.5) les poids correspondants $w_{\{ij\}}$. Vu que le neurone j se trouve dans la couche cachée, les entrées du neurone sont égales aux entrées du réseau ($a_i = \mathrm{E}v_i, i = \{1..k\}$). Dans le cas où le neurone appartient à la couche de sortie, les entrées du neurone sont égales aux sorties des neurones dans la couche cachée ($i = \{1..\mathrm{M}\}$). L'équation (3.5) montre la fonction linéaire du neurone où w_{ij} et r_j sont les paramètres du modèle pour le neurone j. La fonction d'activation dans l'équation (3.6) correspond à la fonction *sigmoid* qui produit un résultat entre -1 et 1, selon le résultat de la fonction linéaire net_j. En effet, le choix de la fonction d'activation affecte le temps d'apprentissage du modèle, et a peu d'impact sur la précision du modèle. Selon [SJS13], si un réseau de neurones a bien appris avec une fonction particulière sur des données représentatives, il y a une forte probabilité que d'autres fonctions produisent aussi des résultats proches. Ainsi, l'apprentissage d'un réseau de neurones est généralement réalisé à partir de la technique de rétropropagation du gradient [Dev82] pour déterminer les paramètres du modèle. Cette technique initialise aléatoirement ces paramètres, et ensuite les ajuste d'une manière itérative pour minimiser l'erreur entre l'estimation et la valeur réelle.

$$net_j = r_j + \sum_{i=1}^{k} (w_{ij} \times a_j) \tag{3.5}$$

$$s_j = \frac{1}{1 + e^{-net_j}} \tag{3.6}$$

3.5 Prototypage sur FPGA

Les circuits FPGA peuvent être envisagés comme une alternative aux circuits dédiés (faibles volumes, notamment) ou bien utilisés en phase de prototypage. Dans les deux cas se pose la question de la mise en œuvre du monitoring de la consommation tel que nous l'avons défini. L'estimation de la consommation est un sujet abordé par les auteurs de [CCF03], où la macro-modélisation à base de pré-caractérisation est utilisée pour relever la consommation moyenne par accès aux LUTs (*Look Up Tables*) et aux registres dans le FPGA. Une autre approche proposée dans [NMH12], utilise un modèle linéaire pour estimer la puissance dynamique en fonction des activités de commutation au niveau RTL. Le modèle est validé sur de petits composants tels qu'un additionneur et un multiplieur. Par ailleurs, des modèles de la consommation pour les systèmes multi-processeurs sur FPGA ont été proposés dans [PP12]. La technique est basée sur la construction en temps de conception de modèles en fonction d'un ensemble d'évènements (nommé "signature") tels que les instructions d'accès mémoire (Load/Store). Les caractérisations sont conduites pour des implémentations FPGA, mais les modèles créés sont par la suite intégrés dans des simulateurs du type SystemC. La contribution se limite à une estimation de la puissance dynamique sans proposer une intégration qui permettrait une observation en-ligne.

Notre méthode cible une implémentation matérielle du monitoring sur FPGA et offre de nombreux avantages, y compris la flexibilité du reconfigurable, et la validation des méthodes de gestion de la puissance sur les FPGAs. Pour cela, nous proposons un flot complet pour générer des bases de données pour les circuits implémentés sur des technologies FPGA. La Figure 3.7 illustre ce flot.

En général, la puissance dynamique dépend de la technologie cible, de la fréquence de fonctionnement et aussi du nombre de ressources occupées par le circuit. La première phase de l'estimation de la consommation instantanée P_{dyn} nécessite la synthèse, le placement et le routage du circuit pour une technologie FPGA donnée. Ensuite, la simulation du circuit placé/routé produit des fichiers qui décrivent l'activité à bas niveau qui sont utilisés par un outil d'estimation de la puissance sur FPGA tel que Xpower de Xilinx. Ces outils qui ne permettent pas d'estimer directement la puissance instantanée (ils ne produisent qu'une estimation de la moyenne consommée pendant toute la durée simulée T) ont nécessité quelques adaptations : la simulation est découpée en N_i intervalles de temps de durées t_i selon l'équation (3.7), avec t_0 correspondant au temps nécessaire à l'initialisation du système en début de la simulation. ModelSim fournit des outils efficaces pour réaliser cette opération tels que *wlfman* pour découper la simulation et *wlf2vcd* pour générer le fichier qui contient l'activité (VCD). Il suffit par la suite de

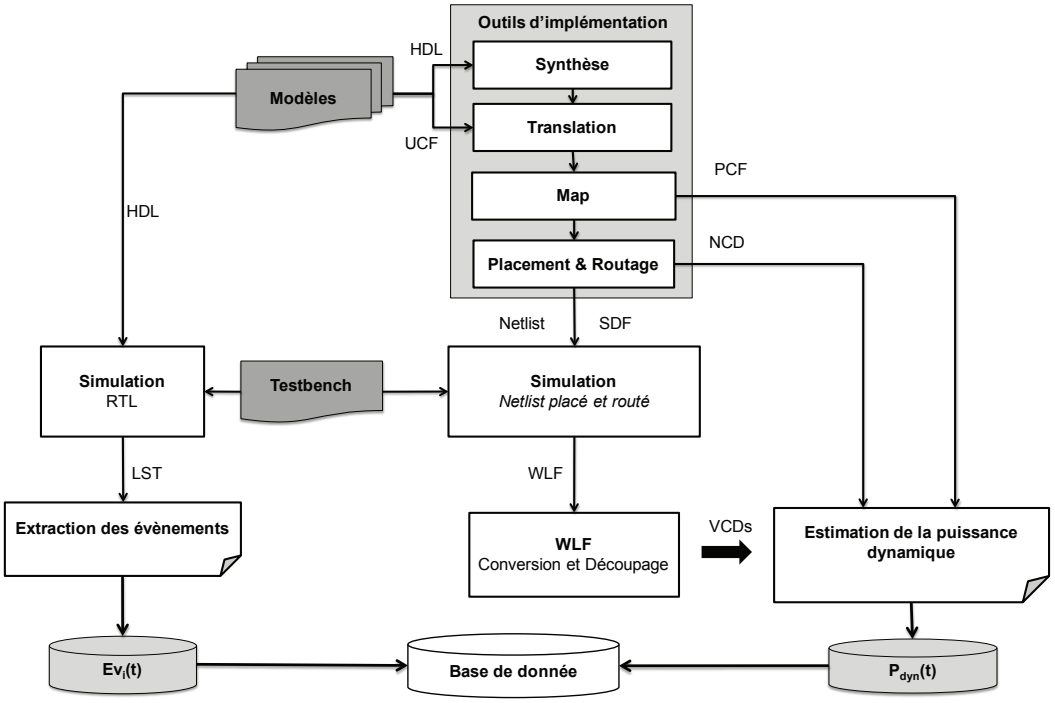


FIGURE 3.7 – Flot complet de génération des bases de données pour les circuits implémentés sur des technologies FPGA.

lancer l'outil d'estimation de la puissance pour récupérer N_i rapports à analyser pour extraire les valeurs de la puissance dynamique instantanée.

$$T = t_0 + N_i * t_i \tag{3.7}$$

Lorsqu'on se place au niveau RTL, l'activité de commutation ne dépend que du modèle de l'architecture et du *testbench*. L'extraction des évènements est ainsi indépendante de la technologie et dans tous les cas nécessite une simulation du circuit au niveau RTL. Une fois la période d'échantillonnage de la puissance t_i spécifiée, le nombre de commutations sur tous les signaux internes du circuit est enregistré pour construire la base de données telle qu'elle est décrite dans la Figure 3.3.

Ce flot peut être adapté pour considérer des valeurs de puissance mesurées sur carte comme alternative aux estimations (voir Figure 3.8). Une instrumentation est ainsi nécessaire pour retrouver ces valeurs, notamment à une résolution temporelle fine. Il suffit ensuite de synchroniser les mesures avec les évènements extraits au niveau RTL pour construire la base de données. L'ensemble des méthodes de sélection d'attributs et de modélisation de puissance peuvent être ensuite appliqués pour définir l'emplacement des compteurs d'évènements et estimer en ligne la puissance.

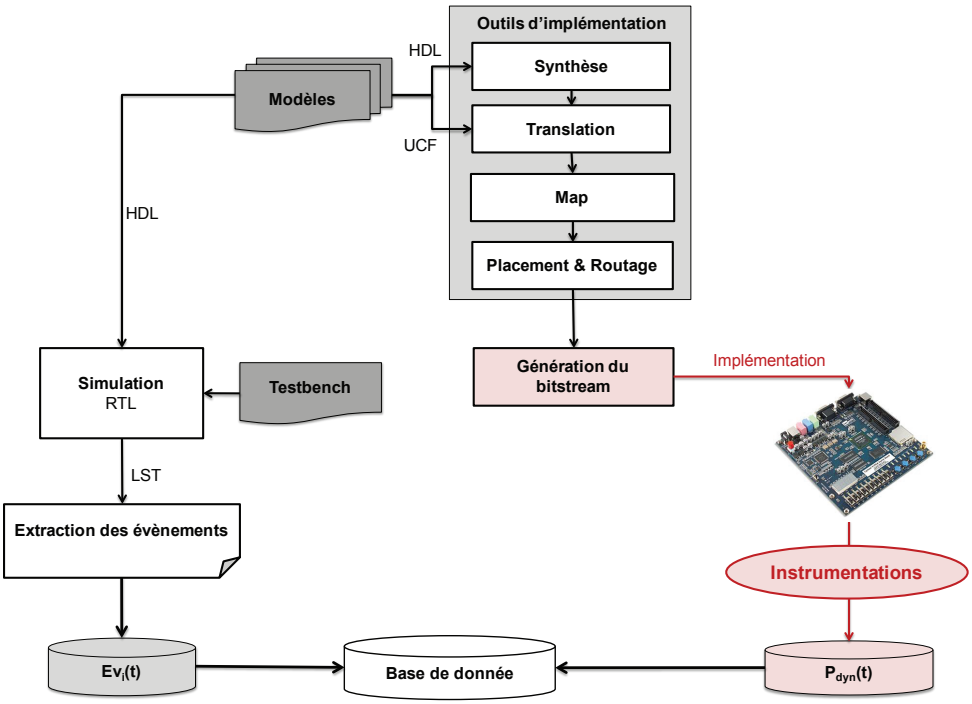


FIGURE 3.8 – Flot complet de génération des bases de données à partir des mesures de la puissance sur des technologies FPGA.

3.6 Évaluation

Dans cette section, nous présentons un cas d'étude permettant d'évaluer notre méthode de monitoring de la puissance sur FPGA, en exposant d'abord le sous-système de monitoring à base de compteur d'évènements et un système sur puce comme circuit de test. Ensuite, nous nous penchons sur l'évaluation des méthodes proposées pour la sélection de signaux. Puis, différents compromis entre la précision des estimations et le coût du système sont discutés. Enfin, une discussion sur la portabilité des résultats sur d'autres technologies est présentée.

3.6.1 Description du cas d'étude

TAPoM : Sous-système de monitoring

Le sous-système de monitoring de la puissance considéré comme cas d'étude dans nos expérimentations est appelé "*Toggling Activity based Power Monitoring*" (TAPoM). Ce sous-système (Figure 3.9) est principalement basé sur des compteurs d'évènements (EC : "*Event Counter*") et une unité de contrôle "*Power estimation and Counters Controller*" (PCC). EC est conçu à partir d'un simple registre qui indique l'occurrence d'un front montant ou descendant produit sur un signal en entrée ; PCC, quant à elle, estime les valeurs de la puissance consom-

mée par le circuit à l'aide d'un modèle. Elle prend en charge aussi la synchronisation des compteurs par rapport à une fréquence d'échantillonnage spécifiée au moment de la conception du circuit.

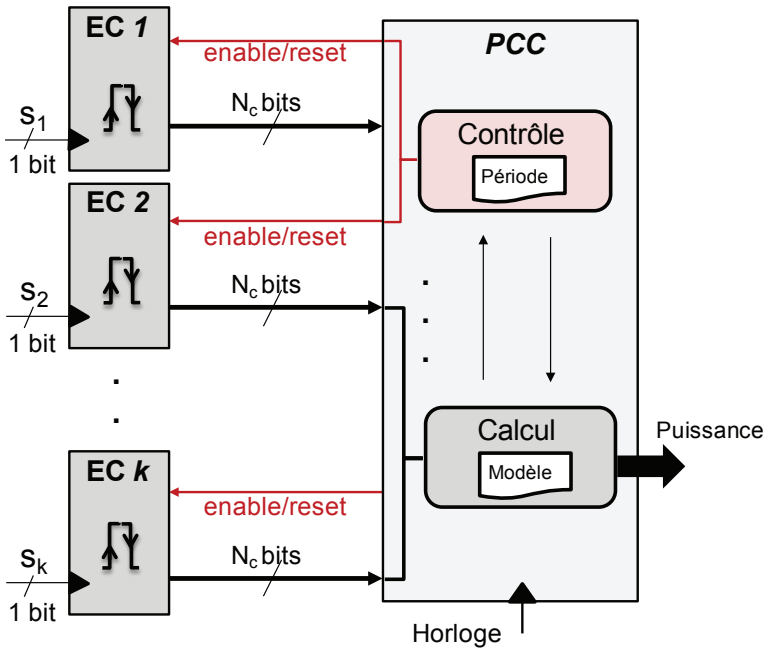


FIGURE 3.9 – Représentation du sous-système de monitoring *TAPoM*.

La justification du choix de conception de TAPoM repose sur ses avantages par rapport aux autres architectures. D'abord, les compteurs ECs ne sont pas connectés à l'horloge, ce qui limite déjà les contraintes liées à la distribution de l'horloge dans le circuit. De plus, l'architecture de TAPoM permet une gestion énergétique plus efficace des compteurs en les activant au moment du monitoring afin de réduire la puissance consommée par les ECs.

Il est impératif d'intégrer les compteurs d'évènements et de les connecter à des signaux au niveau RTL au moment de la conception du circuit. En outre, l'unité de calcul et de contrôle PCC peut être conçue sous forme d'une couche logicielle où le processeur prend en charge la collection des valeurs des compteurs et l'estimation en-ligne de la puissance.

System On Chip : SecretBlaze

La plateforme matérielle utilisée est constituée principalement d'un microprocesseur SecretBlaze [BCBT11] et d'un bus fonctionnant selon le standard Wishbone v4 [HO02]. Les communications entre le processeur, la mémoire interne ainsi que les autres éléments périphériques, s'effectuent à travers un bus Whishbone. La Figure 3.10 illustre l'architecture simplifiée de ce système.

Le SecretBlaze est un microprocesseur Open Source du type Harvard. Il possède une archi-

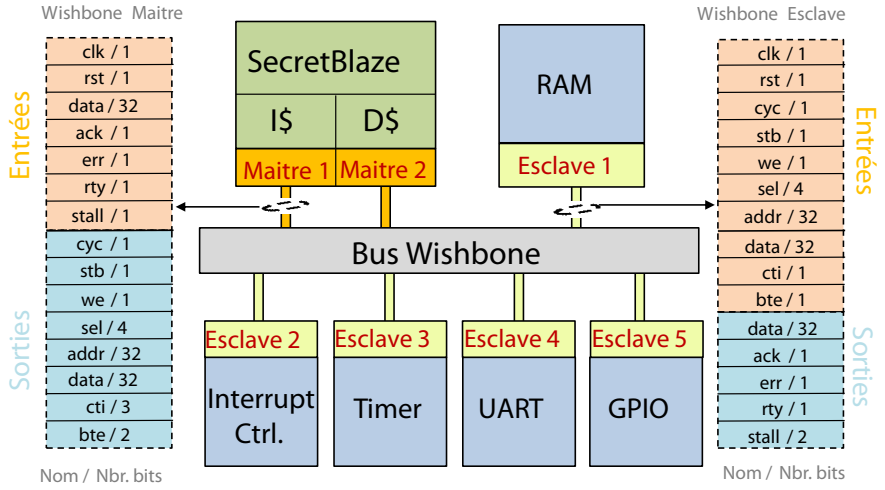


FIGURE 3.10 – Représentation du SoC SecretBlaze.

tecture RISC avec 5 étages de pipeline et utilise le jeu d'instructions du processeur MicroBlaze [Xila]. De plus, ce processeur possède un cache d'instruction et un cache de données entièrement configurable en terme de taille totale, de taille de ligne ou encore de politique de remplacement (écriture directe ou différée). Le contrôleur d'interruption permet, quant à lui, de gérer jusqu'à 8 interruptions en parallèle avec gestion des priorités, masquage et armement. De son côté, le timer est constitué d'un compteur 32 bits permettant de générer une interruption après une durée paramétrable, lorsque l'UART gère la communication RS232 série avec l'extérieur.

Outils et Benchmarks

Nous avons choisi d'expérimenter notre méthode de monitoring sur le circuit reconfigurable Xilinx Spartan-6 LX45 FPGA (45-nm). La carte Atlys qui intègre le Spartan-6 LX45 contient en particulier un capteur de puissance qui mesure en ligne la puissance consommée sur les différents rails d'alimentation. À l'aide de cette plateforme, nous pouvons aussi comparer les valeurs estimées par notre méthode de monitoring aux mesures réelles.

La variation de la puissance est une conséquence de la variation de l'activité dans le système et qui dépend essentiellement de l'application. Afin d'évaluer des activités diverses et variées pour couvrir une large gamme de cas d'utilisation, nous avons expérimenté notre méthode en utilisant l'ensemble des applications *benchmarks* présenté dans le tableau 3.1. La Figure 3.11 montre le profil de consommation extrait à partir de notre flot présenté dans la Section 3.5 qui produit des estimations pseudo-instantanées de la puissance consommée par le processeur SecretBlaze lors de l'exécution de ces applications à une fréquence de 25 Mhz. Ces estimations correspondent à un échantillonnage de la puissance pour une période très fine égale à 100µs. L'outil ISE v13.1 de Xilinx a été utilisé pour générer la *netlist* placée et routée, ainsi que ModelSim 10.1d qui réalise les simulations de la *netlist* générée et de la description RTL du circuit.

TABLEAU 3.1 – Ensemble d'applications considérés dans nos expérimentations.

Application	Référence	# Cycles	Structure et Opérations	Description
Nsichneu	[GBEL10]	2762500	> 250 conditions (if-else)	Simulateur de réseaux de pétri
MJPEG	[MJP93]	87500	Boucles, matrices, entier	Compression/décompression vidéo
MXM	[MxM10]	5682500	Boucles, Matrices, entiers	Multiplication matricielles
compress	[GBEL10]	25000	Boucles, matrices, opérations binaires	SPEC95 compression de données
statemat	[GBEL10]	97500	Boucles	Simulation d'une FSM pour véhicules
qurt	[GBEL10]	37500	Boucles, matrices, virgule flottantes	Opérations exponentielles
whetstone	[CW76]	4250000	Matrices, boucles, conditions, entier, trigonométriques, appel système	Programme de teste de performance

Nous pouvons remarquer une certaine diversité dans le profil de consommation : la puissance consommée par *nsichneu* est quasi constante alors que *MJPEG* provoque une discontinuité brutale, la puissance consommée lors de l'exécution de *MXM* varie autour d'une moyenne avec de petites amplitudes, lorsque *compress*, *statemat* et *qurt* ont des amplitudes plus importantes. En outre, nous pouvons constater les différents niveaux de consommation de *Whetstone* liés à des fonctions différentes : initialisation de matrices, sauts conditionnels, appel système, interruptions et fonctions arithmétiques.

3.6.2 Méthodes de sélections

Le processeur SecretBlaze contient 212 signaux au niveau RTL, ce qui correspond en fait à 1531 connexions de 1 bit. La première étape consiste à définir le sous-ensemble qui caractérise l'activité globale du système. Pour ce faire, nous allons implémenter dans cette section l'ensemble de méthodes de fouille de données pour la sélection d'attributs.

Nous rappelons que notre objectif est de surveiller la puissance à une granularité fine. C'est la raison pour laquelle, nous allons évaluer les méthodes de sélection sur une base de données extraites pour une période d'échantillonnage égale à 100 µs ce qui correspond à 2500 cycles à 25 MHz. Nous démontrerons ultérieurement (dans la section 3.6.5) que les résultats conduits restent valables pour des résolutions moins importantes.

De plus, l'activité est directement proportionnelle au nombre d'opérations réalisées par intervalle de temps. Afin de maximiser l'impact de l'activité sur les variations de la puissance, nous avons fixé la fréquence du SecretBlaze à la fréquence maximale de fonctionnement (25 MHz) lors de l'exécution des applications présentées ci-dessus. La base de données ainsi pro-

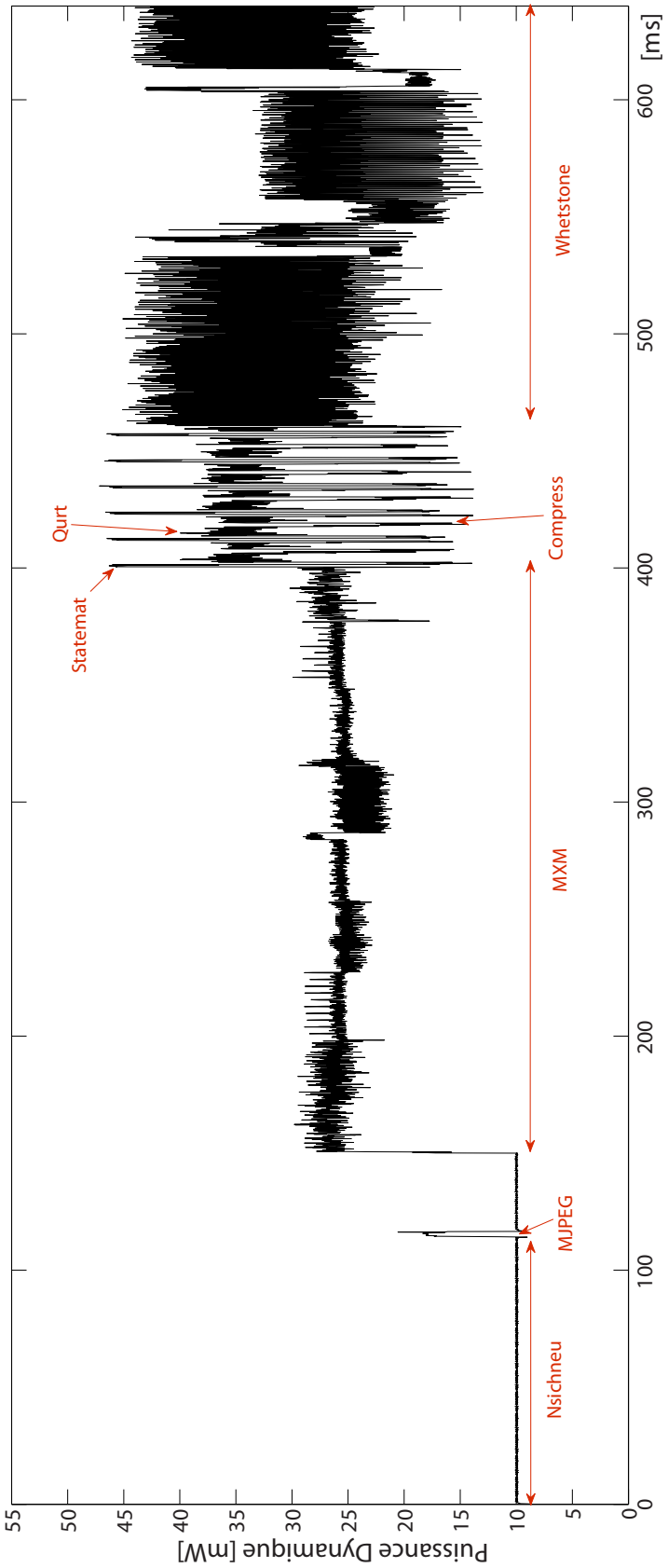


FIGURE 3.11 – Profile de consommation du SecretBlaze a 25 MHz exécutant les différentes benchmarks.

duite contient 1531 attributs (ou signaux) et 6500 instances (voir Figure 3.3 pour la représentation de la base de données).

Corrélations et tests d'hypothèses

Nous rappelons que la méthode standard en statistique consiste à mesurer la corrélation entre chaque attribut et la puissance. Pour tester la significativité d'une corrélation, les statisticiens procèdent à un test d'hypothèse qui permet d'assigner un indice numérique connu sous le nom "*p-value*". En pratique, pour la prise en compte ou le rejet d'une hypothèse on considère un seuil de significativité conventionnellement fixé à 5% [GP75]. C'est-à-dire que si une corrélation est testée avec "*p-value*"<0.05, cette corrélation peut être considérée comme significative avec un risque d'erreur de 5%.

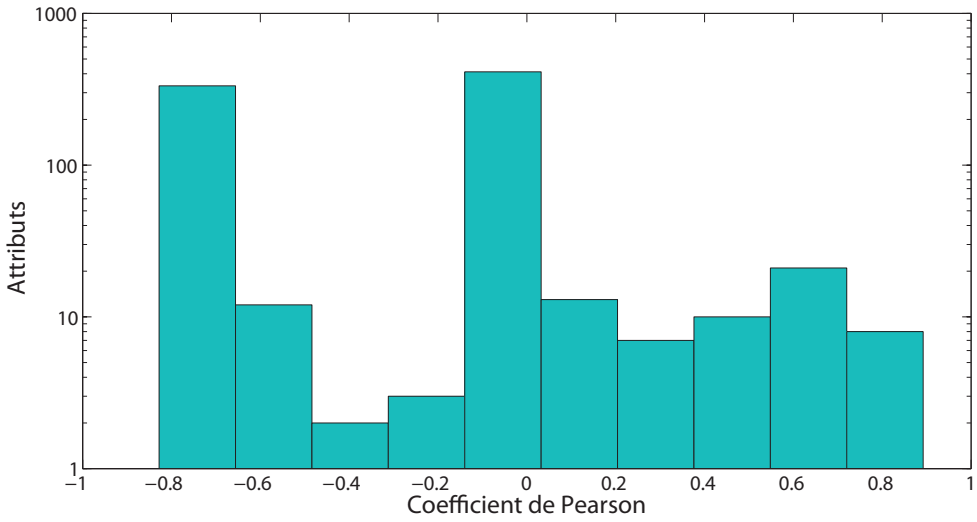


FIGURE 3.12 – Histogramme des coefficients de corrélation des attributs.

Afin d'évaluer cette méthode dans notre cas d'étude, nous avons calculé la corrélation de Pearson entre chaque attribut et la puissance. La sélection selon le test d'hypothèse nous permet de retenir 821 attributs parmi les 1531. L'histogramme des coefficients de corrélation de Pearson est montré dans la Figure 3.12. D'après cette figure, nous pouvons retrouver une diversité de corrélations : les attributs fortement corrélés ($|R| > 0.7$), moyennement corrélés ($0.7<|R|<0.2$) et faiblement corrélés ($|R| < 0.2$) avec la puissance. Cependant, les attributs fortement corrélés avec la puissance peuvent être redondants dans le sens où aucune information supplémentaire peut être rajoutée en les gardant tous. En outre, deux attributs faiblement corrélés avec la puissance peuvent être utiles en les combinant [GE03].

Sélection par les méthodes de fouille de données

D'après l'étude présentée dans la section précédente, il est clair que la sélection selon la méthode standard en statistique n'est pas suffisante pour résoudre notre problème. Ceci justifie notre choix d'explorer d'autres méthodes qui analysent la relation entre les attributs et la puissance, notamment les méthodes de fouille de données pour la sélection d'attributs présentées précédemment dans les sections 3.4 et 3.3. Ces méthodes ont été conçues pour traiter de grandes quantités de données tout en cherchant à retrouver un sous-ensemble d'attributs suffisant pour la modélisation.

Dans cette expérimentation, nous allons étudier et comparer plusieurs techniques issues des méthodes de sélection d'attributs. Le tableau 3.2 montre les différentes combinaisons possibles entre les approches d'évaluation (*filter* et *wrapper*) et les trois types de recherche : (*i*) Greedy Stepwise, (*ii*) BestFirst, et (*iii*) Génétique.

Les deux algorithmes de recherche GreedyStepwise et BestFirst correspondent à deux familles d'algorithmes itératifs, alors que l'algorithme génétique est une méta-heuristique. Dans la littérature, la taille recommandée de la population dans un algorithme génétique est entre 50 et 200 pour une meilleure sélection [RFP13], tandis que le nombre d'itérations (ou génération) doit être largement grand pour assurer la convergence [Gol89]. Pour cela, nous avons fixé dans notre implémentation la taille de la population et le nombre de générations à 100 et 200 respectivement. Cependant, nous avons exclu la recherche exhaustive des expérimentations puisqu'elle est très lente et n'a jamais abouti à une solution après plus de 170 heures d'exécution.

TABLEAU 3.2 – Les différentes combinaisons considérées dans cette expérimentation.

Méthodes de Sélection	**processus de recherche**	**Processus d'évaluation**	
M1	Greedy Stepwise	CFS	
M2	BestFirst	CFS	*Filter*
M3	Genetique	CFS	
M4	Greedy Stepwise	CSE + LM	
M5	BestFirst	CSE + LM	
M6	Genetique	CSE + LM	
M7	Greedy Stepwise	CSE + NN	*Wrappers*
M8	BestFirst	CSE + NN	
M9	Genetique	CSE + NN	
M10	Greedy Stepwise	CSE + MARS	

Nous rappelons que l'évaluation *wrapper* CSE est principalement basée sur la rétroaction d'un modèle. Les modèles ainsi expérimentés avec CSE correspondent à ceux présentés dans la section 3.4 : le modèle linéaire ("*Linear Model*" (LM)), réseau de neurones ("*Neural Network*" (NN)) et MARS. En pratique, le nombre de neurones dans la couche cachée du NN a un impact

sur la précision du modèle et sur coût de calcul. Dans notre cas, au-delà de 3 neurones, l'erreur de NN est maintenue mais avec une complexité croissante. Pour cela, ce paramètre est fixé dans notre expérimentation au plus petit nombre permettant de générer un modèle précis (égal à 3).

L'outil open-source WEKA est utilisé pour l'application des méthodes de sélection. La toolbox Areslab de Matlab vient compléter WEKA pour l'implémentation du modèle MARS. À noter que la méthode de sélection dans l'implémentation de MARS dans Areslab est une méthode itérative du type Greedy qui utilise aussi le critère de RMSE (*Root Mean Squared Error*) pour l'évaluation.

3.6.3 Évaluation des méthodes de sélection

Nous allons évaluer à ce stade les différentes méthodes de sélection présentées dans la section précédente. Dans notre contexte, la méthode la plus efficace est celle qui répond aux critères suivants :

- Nombre d'attributs faibles : chaque attribut correspond en fait à un signal sur 1 bit au niveau RTL. La sélection d'un attribut impliquera l'insertion d'un capteur EC pour la surveillance des évènements de commutation, et donc augmentera le coût du monitoring. Afin de minimiser ce coût, la méthode optimale doit sélectionner le petit nombre d'attributs pour une précision ciblée.
- Temps de réponse le plus petit : le nombre de signaux au niveau RTL a tendance à augmenter avec la croissance de la complexité dans les systèmes intégrés. Pour cela, la méthode doit être capable de produire une solution en cherchant dans de grandes quantités de données. Le temps de réponse de l'algorithme est un critère qui permet d'approximer cette scalabilité ; la méthode la plus rapide correspond à la méthode la plus scalable.
- Précision la plus élevée : le sous-ensemble d'attributs sélectionné par la méthode doit être capable d'estimer à l'aide d'un modèle la puissance avec une précision élevée.

En se basant sur ces trois critères, nous allons appliquer les méthodes de sélection présentées dans le tableau 3.2 sur la base de données initiales comportant 1531 attributs et 6500 instances. Pour ce faire, nous avons utilisé un processeur d'un serveur d'Intel 64 bits fonctionnant à 2.2 GHz. Une comparaison est conduite dans les sections suivantes entre :(*i*) les méthodes *filters*, (*ii*) les *wrappers*, (*iii*) et entre celles du *filter* et *wrapper*.

Comparaison par approche filter

L'évaluation de type *filter* CFS calcule la métrique représentée dans l'équation (3.1) pour évaluer un sous-ensemble d'attributs. Cette métrique est un critère numérique qui reflète la

corrélation entre le sous-ensemble et la puissance. Le but de ce type de sélection est de retrouver le sous-ensemble qui contient des attributs fortement corrélés avec la puissance et faiblement corrélés entre eux ; ce critère correspond au mérite le plus élevé.

TABLEAU 3.3 – Résultats de l'application des méthodes *filter* de sélection.

Critère d'éval.	**Approches *Filter***		
	M1	*M2*	*M3*
Taille (# attributs)	72	53	262
Mérite (CFS)	0.985	0.985	0.931
Temps (min)	1.33	2.5	12.1

Le tableau 3.3 représente les résultats de l'application des trois méthodes M1, M2 et M3 qui correspondent aux trois procédures de recherche : Greedy stepwise, BestFirst et génétique. Une première réflexion permet de conclure que ces trois méthodes produisent des solutions en peu de temps (<15 min), en sélectionnant un nombre élevé d'attributs ; les trois méthodes ont retrouvé plus de 50 attributs, ce qui n'est pas négligeable en terme de coût pour le monitoring. En outre, le temps de réponse de ces méthodes dépend seulement du nombre de candidats à évaluer puisque le calcul de l'équation (3.1) pour l'évaluation est presque instantané.

D'autre part, nous pouvons constater que l'algorithme génétique (M3) est 5 fois plus lent et sélectionne un nombre plus élevé d'attributs par rapport aux méthodes heuristiques (M1 et M2). Ces dernières, quant à elles, produisent des solutions comparables (mérite : M1 = M2). Le critère d'arrêt de ces recherches est la seule différence ; Greedy Stepwise continue la recherche tant qu'il y a un attribut qui améliore ou maintient la valeur du mérite, alors que BestFirst arrête la recherche lorsqu'aucun attribut restant n'améliore le mérite.

Comparaison par approche wrapper

La procédure *wrapper* CSE évalue un sous-ensemble par sa précision par rapport à un modèle donné en utilisant la caractéristique de mesure de l'erreur RMSE (*Root Mean Squared Error*). L'objectif de cette sélection est de retrouver le sous-ensemble qui produit le modèle le plus précis (RMSE le plus petit). La complexité de ces méthodes dépend principalement du nombre de candidats à évaluer, mais aussi du temps d'entrainement des modèles. Quel que soit le processus de recherche, ces méthodes nécessitent la création et l'apprentissage du modèle à chaque itération de l'algorithme. Puisque le temps d'apprentissage des modèles NN et MARS n'est généralement pas négligeable (une dizaine de minutes), le temps de recherche aug-

mente exponentiellement en fonction du nombre de candidats à évaluer. Pour cette raison, les approches *wrappers* associées à ces deux modèles (M7, M8, M9 et M10) sont très lentes dans notre cas ; elles ne produisent pas une solution sous 96 heures.

TABLEAU 3.4 – Résultats de l'application des méthodes *wrappers* de sélection.

Critère d'éval.	**Approche *Wrapper***		
	M4	*M5*	*M6*
Taille (# attributs)	97	84	273
RMSE	0.934	0.934	0.95
Temps (min)	930.15	660.2	5040.86

Le tableau 3.4 résume les principaux résultats de l'application des méthodes restantes M4, M5 et M6. Ces résultats rejoignent les comparaisons conduites dans la section précédente concernant les algorithmes de recherches (M6 par rapport à M4/M5 et M4 par rapport à M5). Cependant, malgré les efforts de ces méthodes pour retrouver un sous-ensemble produisant l'estimation la plus précise, le nombre de signaux sélectionnés reste élevé. En outre, la complexité de ces méthodes est aussi un problème qui nous empêche d'approximer la scalabilité de ces méthodes pour des systèmes plus complexes.

Comparaison wrapper-filter

Dans les deux sections précédentes, nous avons présenté les résultats de l'implémentation des méthodes de sélection selon les deux approches *filter* et *wrapper*. Une comparaison plus générale nous permet de vérifier que les méthodes *filter* sont plus rapides que les méthodes *wrappers*, puisque le temps pris pour calculer le mérite CFS est beaucoup plus court que le temps d'apprentissage d'un modèle. En outre, les contraintes d'implémentation représentées par le temps de réponse des méthodes *wrapper* ne permettent pas d'obtenir une solution avec les deux modèles NN et MARS. De plus, toutes les méthodes sélectionnent un grand nombre d'attributs qui aura un impact non négligeable sur le coût du monitoring.

3.6.4 Méthode de sélection proposée

Nous rappelons que l'objectif est de proposer une méthode qui répond aux trois critères précédemment définis : (*i*) un petit nombre d'attributs, (*ii*) un faible temps de réponse et (*iii*) un sous-ensemble produisant un modèle avec la précision la plus élevée. Ayant des objectifs de sélection différents, aucune des méthodes évaluées ci-dessus ne satisfait ces critères ; les

méthodes *filter* essayent d'optimiser le temps de recherche (critère *ii*), alors que les *wrappers* recherchent la solution la plus précise (critère *iii*).

Afin de combler cette lacune, nous avons proposé précédemment une méthode de sélection en deux phases, illustrée ici dans la Figure 3.13. Dans la première phase (détaillée dans la Section 3.3), nous profitons de la rapidité des méthodes *filter* pour supprimer les données redondantes de la base de données initiale à l'aide de l'évaluation CFS avec une procédure de recherche heuristique (par exemple à l'aide de la méthode M1). La deuxième phase (présentée dans la Section 3.4) consiste à rechercher la solution dans la base de données réduite selon l'approche *wrapper* à l'aide de CSE et d'un modèle spécifié.

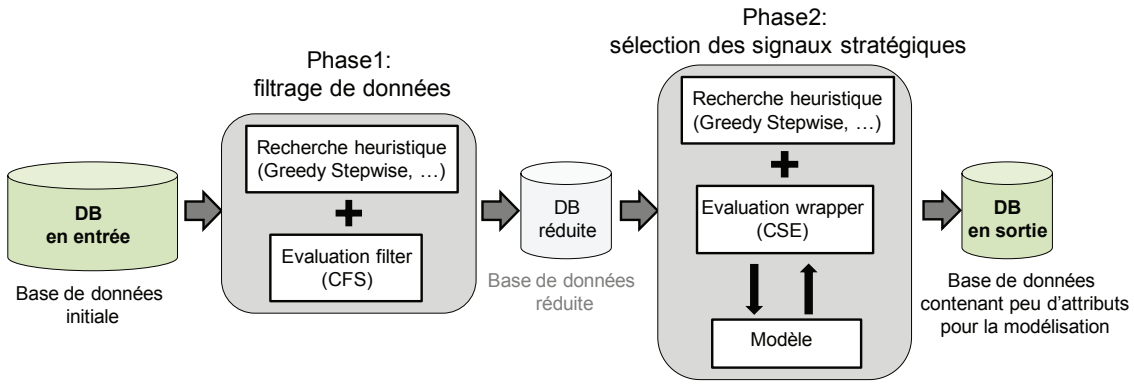


FIGURE 3.13 – La méthode proposée en deux phases pour la sélection d'attributs.

Évaluation de la méthode de sélection

Afin d'évaluer notre méthode, nous allons l'appliquer à la base de données de départ. La première phase correspond à la méthode M1 présentée dans le tableau 3.2. Tandis que, dans la deuxième phase, nous avons expérimenté les trois modèles avec l'évaluation *wrapper* CSE. Les trois expériences seront ainsi nommées dans les sections suivantes par : "B1" pour M1 puis M4 (LM), "B2" pour M1 puis M7 (NN) et "B3" pour M1 puis M10 (MARS).

TABLEAU 3.5 – Résultats de l'application des méthodes *wrappers* de sélection.

Critère d'éval.	**Approche *Wrapper***		
	B1	*B2*	*B3*
Taille (# attributs)	8	6	6
RMSE	1.506	1.392	1.016
Temps (min)	0.52	6.05	7.15

La complexité de notre méthode correspond au temps de traitement en deux phases. Pendant un temps de réponse d'environ 2 minutes, la première a éliminé 1459 attributs redondants parmi les 1531, en comparant à la méthode statistique classique qui élimine 710 attributs. Le tableau 3.5 résume les principaux résultats des trois expérimentations. Après le filtrage de données, la sélection selon l'approche *wrapper* avec les deux modèles NN (B2) et MARS (B3) devient réalisable avec un temps raisonnable (< 10 min). En comparant la solution produite par B1 avec M4 dans le tableau 3.4, nous pouvons conclure que le filtrage avec B1 permet de réduire 1000 fois le temps de recherche. En outre, grâce à notre méthode, il devient possible de sélectionner très peu de signaux (<10) pour la modélisation de la puissance. Une étude approfondie sur la précision des modèles construits à partir des signaux sélectionnés est conduite ultérieurement dans la Section 3.6.5.

Signaux sélectionnés

Les signaux sélectionnés par les trois méthodes B1, B2 et B3, sont représentés dans le tableau 3.6. La plupart de ces signaux correspondent à quelques bits du bus de données et d'adresses de l'interface Wishbone (voir Figure 3.10). Il faut cependant noter que les informations transmises à travers les bus de données et d'adresses passent à travers d'un bus partagé à l'intérieur de l'interconnexion Whishbone (voir la spécification illustrée dans la Figure 3.14). Un contrôleur dédié gère ainsi la communication et sélectionne pour chaque requête une cible à travers les signaux de contrôle. Par conséquent, toutes les requêtes émises ou bien reçues par un maître ou un esclave, vont forcément changer les bits des bus de données ou d'adresses. La surveillance de quelques signaux sur ces bus est suffisante pour approximer l'activité globale du système. Dans tous les cas, cette information ne peut pas être retrouvée en se basant sur quelques observations ou une pré-connaissance du circuit. Grace à notre méthode de sélection, il devient possible de retrouver ces signaux d'une manière automatisée pour alimenter le sous-système de monitoring TAPoM.

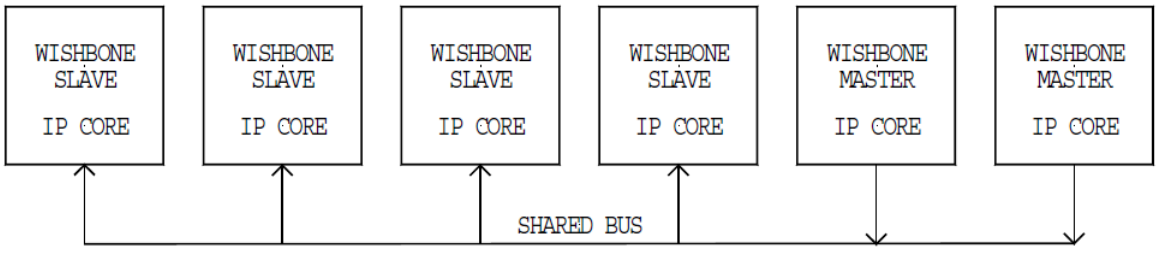


FIGURE 3.14 – La spécification du bus partagée à l'intérieur de l'interconnexion [Ope02].

TABLEAU 3.6 – L'ensemble de signaux sélectionner par les trois méthodes B1, B2 et B3.

Numéro	Méthodes de sélection	Interface M-E	Description	Corrélation de Pearson
1	B1	Maitre 2	addr(10) : bit 10 du bus d'adresse en sortie.	0.8921
2	B1, B2, B3	Maitre 1	addr(7) : bit 7 du bus d'adresse en sortie.	0.805
3	B1	Maitre 1	data(18) : bit 18 du bus de données en entrée.	-0.8021
4	B1, B2, B3	Maitre 1	addr(7) : bit 7 du bus d'adresse en sortie.	-0.0289
5	B1, B2, B3	Maitre 1	addr(8) : bit 8 du bus d'adresse en sortie.	0.7579
6	B1	Esclave 1	data(21) : bit 21 du bus de données en entrée.	-0.0364
7	B1	Maitre 2	data(20) : bit 20 du bus de données en sortie.	0.7055
8	B1	Maitre 1	data(8) : bit 8 du bus de données en sortie.	-0.0289
9	B2, B3	Maitre 1	data(10) : bit 10 du bus de données en entrée.	-0.8132
10	B2, B3	Maitre 2	addr(14) : bit 14 du bus d'adresse en sortie.	0.6411
11	B2	Esclave 2	clk : horloge du bus wishbone en entrée.	-0.0283
12	B3	Maitre 2	addr(26) : bit 26 du bus d'adresse en sortie.	-0.0407

3.6.5 Précision des modèles et suivi de la puissance dynamique

Dans cette expérimentation, nous souhaitons évaluer la précision des trois modèles présentés dans la Section 3.4 construite à partir des signaux sélectionnés. Pour ce faire, nous considérons les trois bases de données sélectionnées précédemment (voir le tableau 3.6) : B1 avec 8 attributs sélectionnés par la méthode M4 (LM), B2 avec 6 attributs extraits de la méthode M7 (NN), et B3 avec 6 attributs à partir de M10 (MARS). Tel que cela se pratique classiquement dans la fouille de données, ces bases sont découpées aléatoirement selon les instances en deux parties : (*i*) 70% pour l'entrainement du modèle et (*ii*) 30% pour la validation et l'évaluation de l'erreur. La précision des modèles est exprimée par l'erreur relative et par le coefficient de détermination (nommé R^2) qui détermine à quel point le modèle est adapté pour décrire la référence (plus la valeur de R^2 est proche de 1, mieux c'est).

La Figure 3.15 montre le suivi des modèles de la puissance dynamique par les différents modèles pour une période d'échantillonnage minimale égale à 100µs. Trois zooms sur des intervalles de temps exposant des propriétés intéressantes sont proposés. La première lorsqu'une discontinuité est présente a cause de MJPEG : les deux modèles LM et NN ne sont pas en mesure de suivre la référence, alors que MARS est le seul modèle qui la suit avec une bonne précision. Dans les autres zones (MXM et Whetstone), nous constatons que les trois modèles sont capables de suivre les variations de la puissance. En principe, une bonne estimation est représentée par un coefficient R^2 supérieur à 0.9. Ce coefficient est mesuré à 0.9817 pour LM, 0.9833 pour NN et 0.9889 pour MARS. D'autre part, l'erreur relative des trois modèles est illustrée dans la Figure 3.15-e ; LM estime 66% du temps la puissance avec une erreur inférieure à 5%, alors que NN et MARS sont plus précis à cette granularité temporelle et atteignent 78% et 87% respectivement.

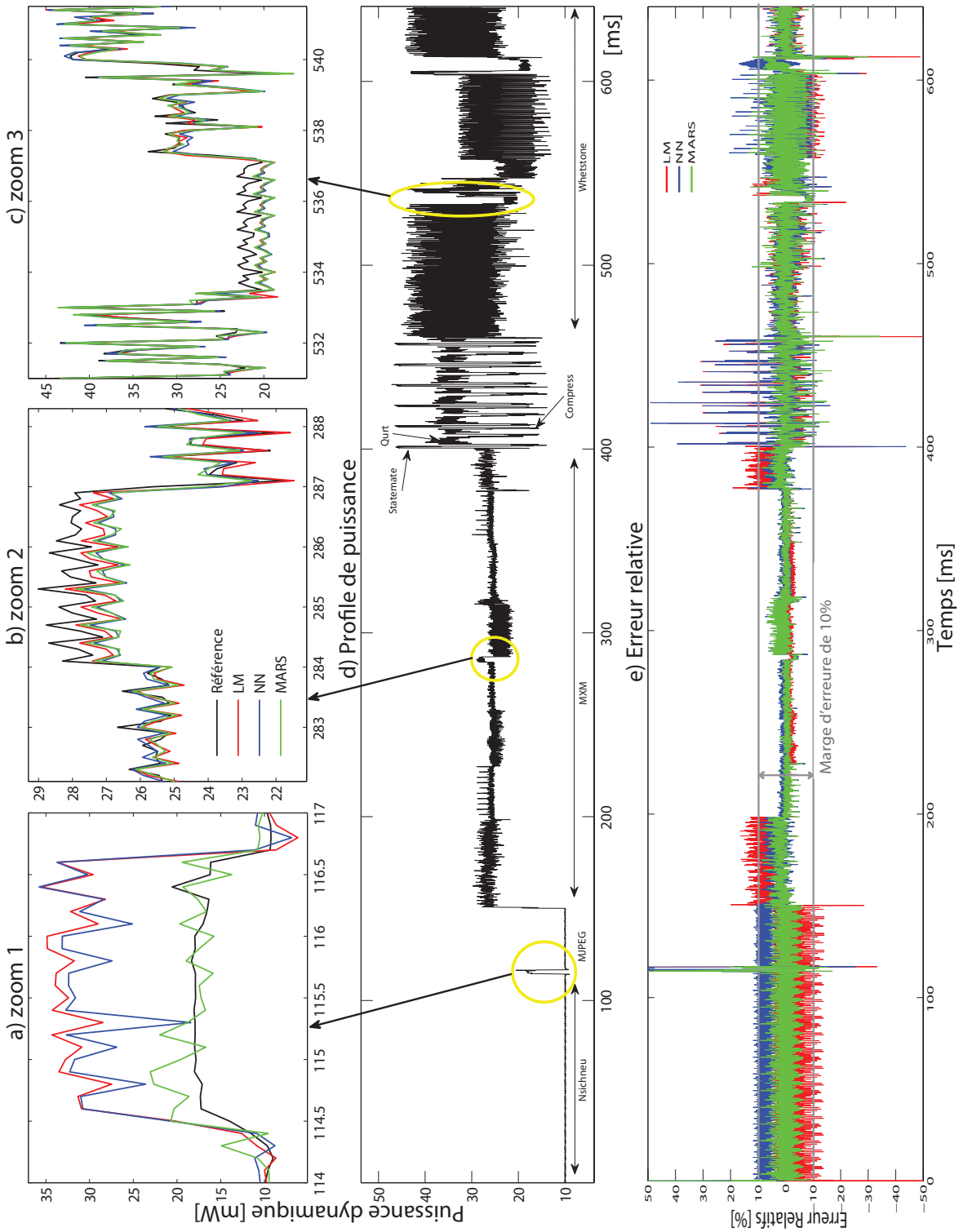


FIGURE 3.15 – Les estimations de la puissance par les trois modèles : LM, NN et MARS.

La précision des modèles obtenue précédemment pour une période égale à 100µs s'améliore pour une résolution temporelle plus basse, où la puissance est moyennée sur des durées plus longues. La Figure 3.16 montre la variation de la moyenne de l'erreur relative pour différentes périodes d'échantillonnage T. L'erreur du modèle linéaire est inférieure à 3% pour T égal à 1 ms ; l'erreur de NN et MARS est environ de 1%.

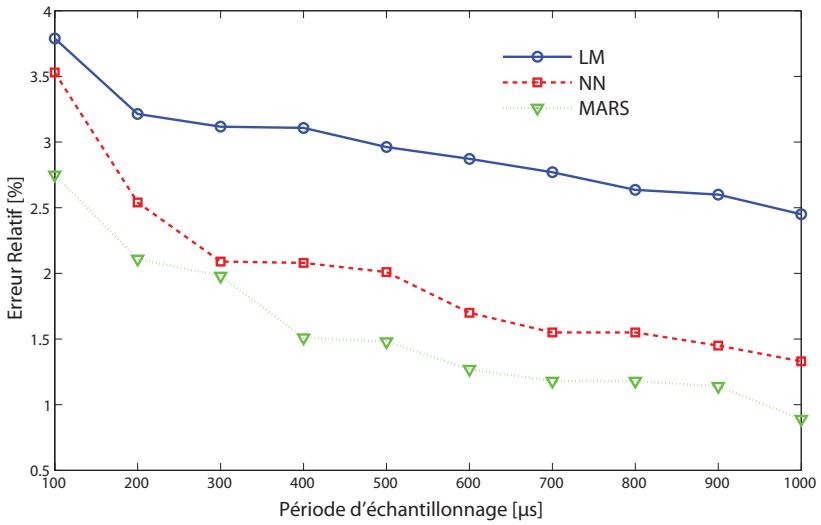


FIGURE 3.16 – La moyenne de l'erreur relative des modèles en fonction de la durée de la période d'échantillonnage.

3.6.6 Modèles réduits : Précision

Précédemment, les trois modèles ont été construits avec un sous-ensemble de signaux produisant le modèle le plus précis. Nous rappelons ainsi que la surveillance de l'activité de commutation en ligne sur chaque signal nécessite l'insertion d'un capteur d'activité matériel. Ceci rajoute un coût matériel, logiciel et énergétique. Donc, il est préférable de réduire au maximum le nombre de signaux dans le modèle afin de minimiser le coût du monitoring. Pour cela, nous allons étudier la précision des modèles dégradés en réduisant le nombre d'attributs.

Nous partons des trois bases de données construites précédemment à une fréquence d'échantillonnage égale à 100µs : B1 avec 8 attributs pour créer le modèle LM, B2 avec 6 pour NN et B3 avec 6 pour MARS. De manière itérative, nous supprimons l'attribut qui a le plus petit impact sur la précision. Ensuite, nous entrainons les modèles en utilisant les attributs restants. L'erreur est ainsi représentée par la moyenne de l'erreur relative et R^2 dans les deux Figures 3.17 et 3.18. Nous constatons que les trois modèles avec plus de 3 attributs sont capables de suivre les variations de puissance (R^2> 0.9) et ce avec une bonne précision (erreur <6%). En outre, le gain en précision à partir de 3 attributs est inférieur à 0.5% pour chaque attribut supplémentaire. Nous constatons aussi que MARS est plus précis que NN et LM sauf pour k égal à 1. Nous observons également que NN est plus précis que LM pour k inférieur à 4 et l'erreur est comparable

pour les autres cas. En conclusion, nous considérons pour la suite des expérimentations deux solutions : (*i*) avec les modèles de base permettant d'obtenir la meilleure précision et (*ii*) avec les modèles réduits en utilisant 3 attributs qui correspondent au nombre minimal d'attributs nécessaires pour produire un modèle avec une précision suffisante (R^2> 0.9 et erreur <6%).

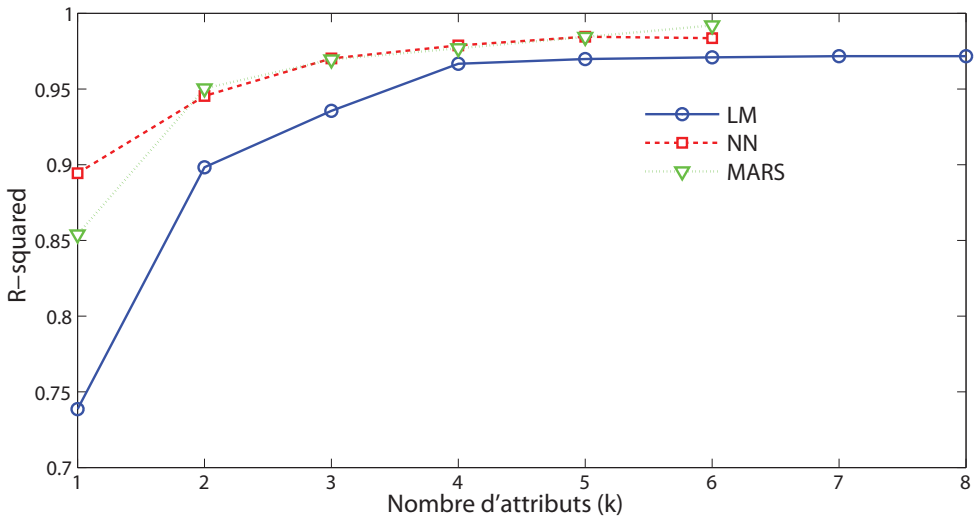


FIGURE 3.17 – Coefficient de détermination (R^2) pour LM, NN et MARS entrainés pour différentes tailles du sous-ensemble d'attributs.

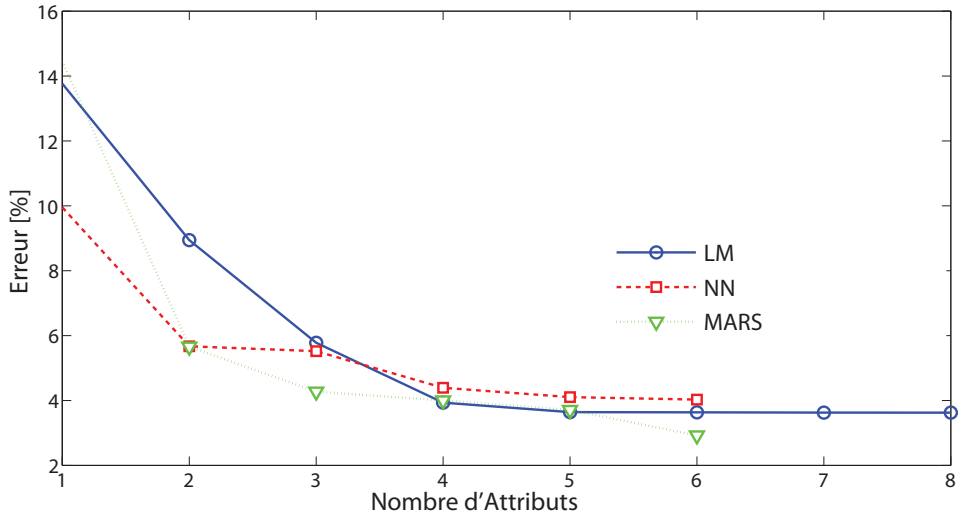


FIGURE 3.18 – La moyenne de l'erreur relative du LM, NN et MARS entrainés pour différentes tailles du sous-ensemble d'attributs.

Un autre aspect de notre étude correspond à l'impact de la résolution temporelle sur la précision des modèles réduits avec 3 attributs. Dans la Figure 3.19, nous avons tracé la moyenne de l'erreur relative pour les trois modèles en fonction de la période d'échantillonnage. D'après cette figure, nous constatons que les modèles avec 3 signaux sont capables d'atteindre une précision assez élevée pour une granularité temporelle assez fine. Pour une période supérieure à

400µs, les trois modèles ont une erreur moyenne inférieure à 5%. Globalement, MARS est plus précis que LM et NN.

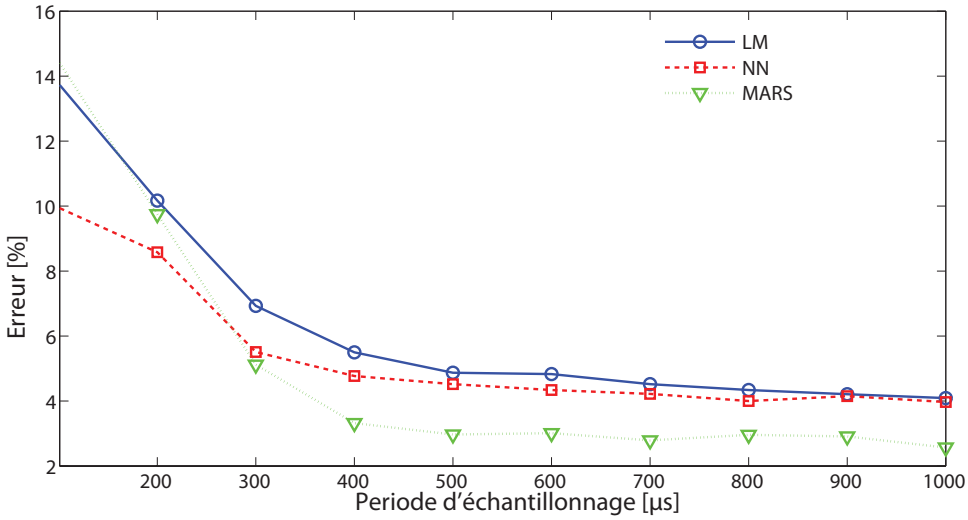


FIGURE 3.19 – La moyenne de l'erreur relative des modèles réduits de LM, NN et MARS pour différentes périodes d'échantillonnages.

3.6.7 Coût du monitoring

Dans la section précédente, nous avons exposé les résultats concernant la précision de notre méthode de monitoring de puissance dynamique, en nous basant uniquement sur l'information issue de commutation de quelques signaux au niveau RTL. Il convient maintenant d'étudier ce qu'implique cette approche, notamment en terme de coût matériel (surface), coût de performance et coût énergétique. L'évaluation est conduite sur les deux cas suivants : (*i*) les modèles réduits à 3 signaux et (*ii*) les modèles de base (8 attributs pour LM et 6 attributs pour NN et MARS).

Coût en Surface

Nous avons conçu le sous-système de monitoring TAPoM proposé comme cas d'étude dans cette expérimentation. Le compteur EC compte le nombre de fronts montants et descendants sur un signal en entrée, lorsque le processeur joue le rôle de l'unité de contrôle (PCC) qui gère tous les compteurs intégrés en les activant et en les réinitialisant périodiquement à travers une couche logicielle. La taille du compteur varie en fonction de la période d'échantillonnage qui nécessite 12 à 16 bits. Cette taille n'a pas d'impact sur le FPGA SC6LX45, car les ressources utilisées sont les mêmes ; chaque compteur occupe 8 "slices-register" et 8 *slices* LUTs. Le SecretBlaze occupe 1574 *slices* registres et 2466 slices LUTs. Le coût matériel du monitoring de la

puissance varie donc entre 1.53% (pour le modèle réduit avec 3 signaux) et 4.08% de *slices* registres supplémentaires (pour un modèle avec 8 signaux), et entre 0.96% et 2.56% de *slices* LUTs supplémentaires.

Coût en Performance

La charge supplémentaire de calcul correspond au nombre de cycles d'horloge nécessaires pour effectuer le calcul de la puissance après avoir collecté les valeurs des compteurs pendant la période d'échantillonnage. Ce calcul est illustré dans la Figure 3.20 : il dépend du modèle choisi et du nombre de signaux considérés. Cependant, les trois modèles ont des complexités différentes qui affectent le coût en performance du monitoring. LM nécessite seulement une multiplication et une addition pour chaque signal, alors que MARS est la somme de M fonctions basiques (voir equation (3.3)), chacune nécessitant : 1 multiplication, 1 addition et 1 test de condition. En outre, le temps pris par le modèle NN pour compléter le calcul dépend du nombre de neurones dans la couche cachée. Nous avons approximé la complexité de NN par l'équation (3.8), où k est le nombre de signaux, C_{Mult}, et C_{Add} le nombre de cycles pour exécuter 1 multiplication et addition respectivement. D'après cette équation, nous pouvons constater la forte dépendance entre le nombre de cycles pris par le réseau NN et le nombre de neurones dans les couches cachées (N_{HL}). Ceci justifie notre choix de minimiser ce paramètre ; dans notre cas il est égal à 3. De plus, le calcul de la fonction de transfert est très coûteux puisqu'elle contient la fonction exponentielle (voir equation (3.6)). Afin de réduire ce temps, nous avons choisi de remplacer dans cette expérimentation la fonction de transfert classique par une fonction plus rapide dans l'équation (3.9). Dans tous les cas, la fonction de transfert n'affecte que le temps de calcul et d'apprentissage du modèle et a un effet négligeable sur la précision du modèle [SJS13].

$$C_{NN} = N_{HL}.(k.C_{Mult} + K.C_{Add} + C_t + C_{Mult} + C_{Add}) + C_t \tag{3.8}$$

$$FastSigmoid(x) = \frac{0.5 \times x}{1 + \bar{x}} + 0.5 \tag{3.9}$$

Le coût en performance du monitoring est donc calculé comme étant le nombre de cycles divisé par la période d'échantillonnage. Nous représentons ce coût sur la Figure 3.21 pour différentes périodes d'échantillonnage. À noter que, le coût minimal d'un modèle pour une période d'échantillonnage définie correspond au temps de calcul de la version réduite à 3 signaux de ce modèle, lorsque le coût maximal est celui des modèles avec la précision maximale (LM : 8 signaux, NN : 6 signaux et MARS : 6 signaux). Il n'est pas surprenant que le modèle linéaire soit le moins coûteux, mais il est aussi intéressant de constater que tous les modèles ont un coût relativement faible (<10%) pour une période d'échantillonnage supérieure à 200µs. Nous

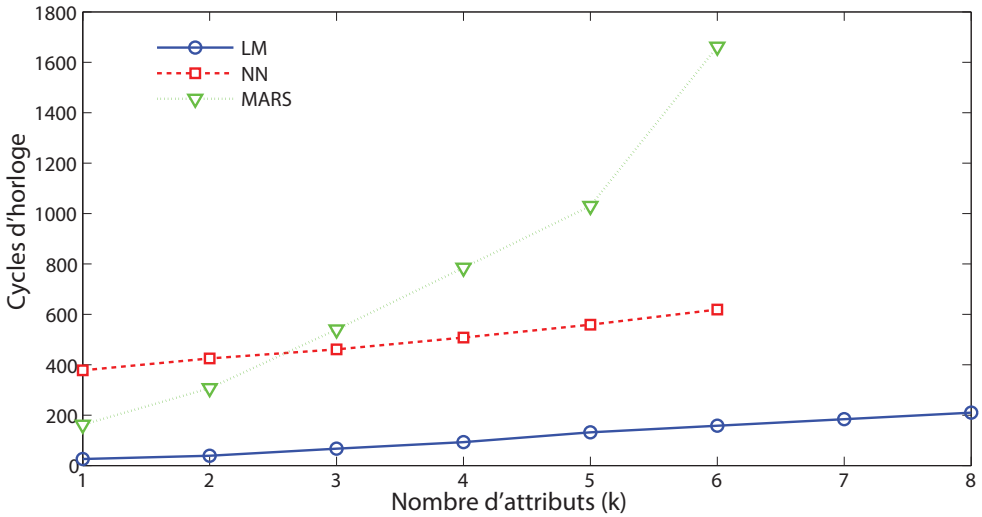


FIGURE 3.20 – La complexité des trois modèles LM, NN et MARS en nombre de cycles d'horloge en fonction du nombre d'attributs.

observons aussi que MARS est très coûteux : cela est dû à la méthode de sélection qui associe plusieurs fonctions basiques pour un signal.

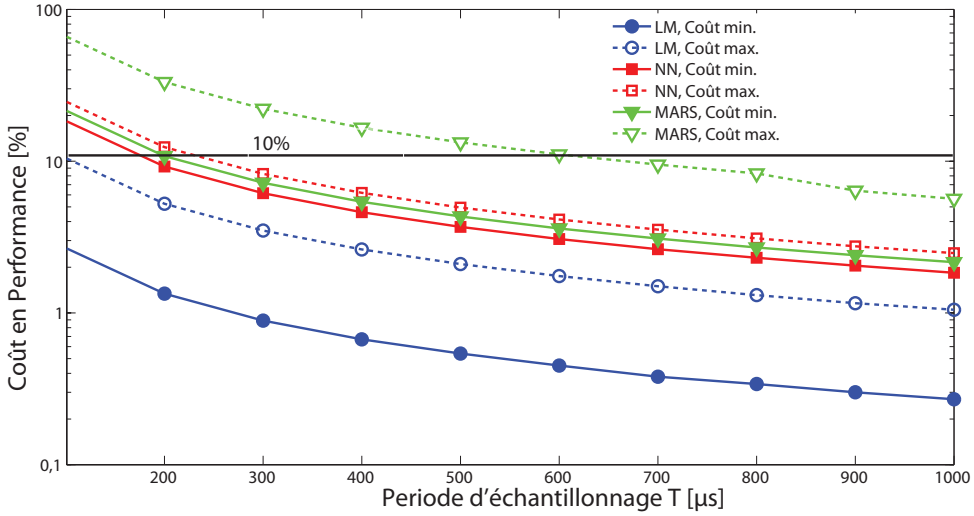


FIGURE 3.21 – Le coût en performance des modèles.

Coût Énergétique

Le coût énergétique du monitoring dépend de la puissance consommée par les compteurs d'évènements (EC), et par le processeur qui prend en charge le calcul. L'énergie supplémentaire consommée par tout le système de surveillance est exprimée en fonction du nombre de signaux k et de la période d'échantillonnage T dans l'équation (3.10). E_{EC} correspond à l'énergie consommée par les compteurs d'évènements, et est égale à $k \times P_{EC} \times T$. La puissance maximale

consommée par un compteur d'évènements P_{EC} est de 0.01 mW. E_{min} est l'énergie minimale consommée par le système de base. Il est égal à $P_{min} \times T$: P_{min} est estimé à 10 mW, ce qui correspond à une activité minimale du système. E_{model} est l'énergie consommée par le processeur lors du calcul de la valeur de la puissance par un modèle précis. Il est égal à $P_{model} \times C_{model}$, avec C_{model} le nombre de cycles nécessaires pour compléter ce calcul (voir Figure 3.20).

$$E_{ov} = \frac{E_{extra}}{E_{min}} = \frac{k \times E_{EC} + E_{model}}{E_{min}} = \frac{k \times P_{EC} \times T + P_{model} \times C_{model}}{P_{min} \times T} \tag{3.10}$$

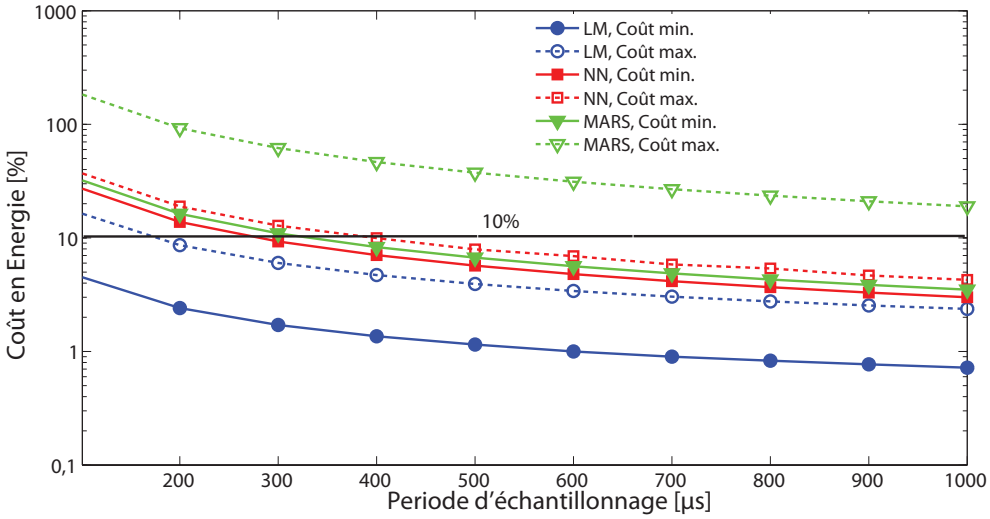


FIGURE 3.22 – Le coût énergétique des modèles.

Dans cette expérimentation, nous avons considéré le pire cas, où les compteurs sont toujours actifs. La Figure 3.22 montre le coût énergétique minimal des trois modèles réduits et le coût maximal pour une précision maximale (LM avec 8, NN avec 6 et MARS avec 6 signaux). Au regard de cette figure, nous pouvons constater que LM est le plus adapté, où le coût maximal en énergie (pour k égal à 8) pour une période supérieure à 200µs est inférieur à 10%. Cependant, le coût des deux modèles NN et MARS devient abordable (<10%) pour une période d'échantillonnage supérieur à 400µs.

3.6.8 Interprétation des résultats

Notre méthode a été évaluée en termes de précision et de coût. Une solution optimale correspond ainsi à celle qui a la plus grande précision et un coût minimal. Les études réalisées précédemment montrent que le nombre de signaux k et la période d'échantillonnage T ont un impact direct sur la précision et le coût global. Pour cela, plusieurs compromis peuvent être envisagés en fonction de ces deux paramètres. La première solution consiste à utiliser le modèle linéaire avec au moins 3 signaux ; le modèle est capable de suivre les variations de la puissance

pour une résolution temporelle très fine (≥ 200 µs). L'erreur moyenne est de l'ordre de 4% dans cette configuration, avec un coût en performance et en énergie inférieur à 10% pour le pire cas considéré. La deuxième solution consiste à utiliser le modèle MARS avec 3 ou 4 signaux, où l'erreur moyenne tourne autour de 2.5%. Le modèle est capable de suivre toutes les variations de la puissance, mais nécessite des périodes d'échantillonnage plus longues par rapport à LM, compte tenu des pénalités énergétiques qu'implique cette approche (période ≥ 400 µs). Compte tenu des résultats obtenus dans cette étude, le modèle de réseau de neurones peut être éliminé ; pour une précision équivalente, son coût en performance et en énergie est plus élevé.

On trouve dans la littérature récente des méthodes sur ce sujet. Le tableau 3.7 compare notre méthode à quelques méthodes existantes. La seule approche similaire est proposée dans [MBTC13]. Leur solution est validée sur de petits circuits tels que DSP et RAM : l'erreur moyenne obtenue est de 10% pour une résolution temporelle très fine (100µs). Notre solution est capable de suivre la puissance pour la même résolution temporelle avec une précision plus élevée et pour un circuit plus complexe. L'erreur obtenue est inférieure à 10% et atteint 3% pour les modèles construits à partir de 6 signaux.

TABLEAU 3.7 – Comparaison avec les méthodes existantes.

Publication	**Niveau d'abstraction**	**Erreur**	**Résolution temporelle**	**Coût**	**Plateforme**
[MBTC13]	RTL	10%	100µs	Non précisé	DSP et RAM
[PMV13]	Système	8-10%	500ms	Non précisé	ARM Big.LITTLE
[BWeWK03]	Système	7%	1ms	Non précisé	Intel Pentium4
[IM03]	Système	6-12%	440ms	Non précisé	Intel Pentium4
[BJ12b]	Système	<9%	1s	Non précisé	Intel AMD Dual-core
Notre méthode	RTL	10 → 3%	100µs → 1ms	<4% (surface) 10→0.5% (perf.) 10→2% (énergie)	SecretBlaze

La majorité des méthodes existantes repose sur des informations au niveau système pour estimer la puissance consommée par un circuit [PMV13, BWeWK03, IM03, BJ12b]. La résolution temporelle à ce niveau d'abstraction est relativement élevée (> 1ms) à cause du monitoring hybride qui se base aussi sur des moniteurs logiciels. L'erreur mesurée varie entre 6 à 10%. En comparant nos résultats, notre méthode est beaucoup plus précise où l'erreur est environ 3% à la résolution de 1 ms.

TABLEAU 3.8 – Comparaison du coût de notre sous-système de monitoring *TAPoM* avec un PMU.

Monitoring	**Plateforme**	***Slices* Registre**	***Slices* LUTs**
PMU [KP]	Virtex6	3.2%	6%
TAPoM	Spartan6	1.53%	0.96%

Le coût du monitoring est peu indiqué dans la littérature, où la plupart des techniques évaluent seulement la précision des estimations. Pour cela, nous avons choisi de comparer le coût matériel de notre méthode par rapport au coût d'un PMU qui est largement utilisé pour le monitoring de la puissance. Les auteurs dans [KP] ont proposé une architecture d'un PMU pour le processeur LEON3 avec 6 compteurs matériels implémentés sur le Xilinx ML605 Virtex-6 FPGA. D'après le tableau 3.8 qui résume le coût des deux méthodes, nous pouvons constater que notre sous-système de monitoring est deux fois moins coûteux.

3.7 Validation et Extension des résultats

3.7.1 Validation

Dans les expérimentations précédentes, les résultats ont été obtenus à partir des estimations de la puissance dynamique à l'aide de l'outil XPower. Dans cette partie, nous souhaitons valider ces résultats par une implémentation de la méthode sur une carte FPGA ATLYS qui dispose d'une sonde de courant. Le capteur de puissance est un convertisseur analogique-numérique de type sigma-delta LTC2481C [Tec14]. Il échantillonne à la fréquence de 50Hz (20 ms) la puissance consommée sur le rail d'alimentation de la puce FPGA. Les valeurs en sortie du capteur codé sur 16 bits sont accessibles à l'aide d'une connexion série JTAG.

De manière similaire aux simulations, nous implémentons le système SecretBlaze à une fréquence de 25 MHz, avec un nouvel ensemble d'applications pour produire une séquence d'activité différente : AES (*Advanced Encryption Standard*) et DES (*Data Encryption Standard*) deux applications de cryptographie et l'application NOP qui contient des instructions "nop". 3 compteurs ECs enregistrent les activités des signaux sélectionnés dans l'étude proposée précédemment (voir Sections 3.6.4 et 3.6.6). Une unité de contrôle gère les compteurs et effectue l'estimation de la puissance consommée par le circuit.

Le suivi de la puissance totale est effectué ici pour une résolution temporelle égale à 100ms et avec les trois modèles LM, NN et MARS (Figure 3.23). La courbe noire correspond à la puissance mesurée par le capteur de puissance externe, tandis que les autres courbes correspondent à l'estimation de LM, NN et MARS. Au regard de cette figure, nous pouvons constater que notre sous-système de monitoring suit les variations de la puissance avec une erreur très faible (égale à 0.9%). Il devient ainsi possible de surveiller sur la puce la puissance consommée par un circuit avec une résolution temporelle très fine et un coût très faible. Finalement, cette expérience permet une première validation des résultats à partir des méthodes de sélection présentées dans les sections précédentes.

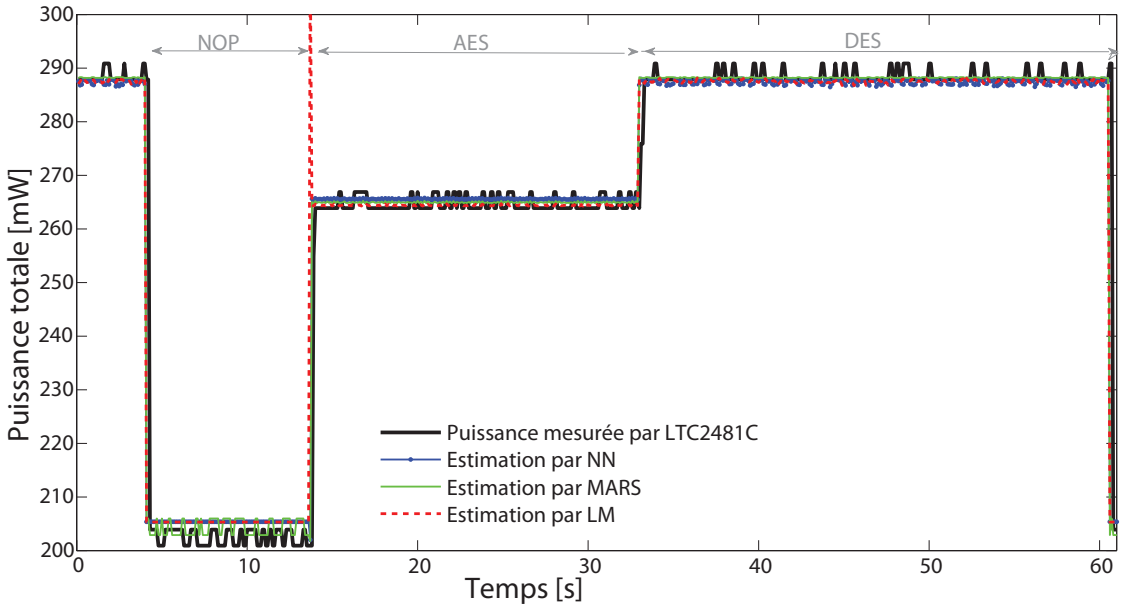


FIGURE 3.23 – Estimation en-ligne de la puissance totale par les 3 modèles et la sonde de courant LTC2481C.

3.7.2 Extension des résultats à d'autres technologies

Notre méthode consiste principalement à modéliser la consommation à partir d'un sous-ensemble de signaux, permettant une estimation en ligne de la puissance. Nous rappelons dans l'équation (3.11) que P_{totale} est une fonction de l'activité α, mais aussi d'autres paramètres tels que la fréquence de fonctionnement f , la tension d'alimentation V_{dd}, la température T et d'autres caractéristiques technologiques qui affectent directement le courant de court-circuit I_{cc} et le courant de fuite I_f. Dans toutes nos expérimentations, nous avons supposé tous ces paramètres constants en variant seulement l'activité interne du circuit. Dans cette partie, nous étudierons le portage de notre méthode sur une autre technologie avec des caractéristiques différentes.

$$P_{totale} = f(\alpha, C, V_{dd}, f, T, I_{cc}) \tag{3.11}$$

TABLEAU 3.9 – Caractéristiques des FPGA : Atlys et Virtex5.

FPGA	Technologie	type des LUTs	V_{dd}	Speed Grade
Atlys (xc6slx45)	45	LUT6	1.2V	-3
Virtex5 (xc5vlx110t)	65	LUT6	1V	-1

Nous avons choisi d'expérimenter notre méthode sur le FPGA Virtex5 XUPV5 ML509 dont les caractéristiques sont comparées à celle de l'Atlys dans le tableau 3.9. Dans Xc5vlx110t un

slice est principalement constitué de 4 LUTs et 4 *Flip-Flop*, alors que dans xc6slx45 il existe 3 types de slices : (*i*) SliceX avec des LUTs et des Flips-Flops, (*ii*) SliceL contient en plus du SliceX un multiplexeur et de la logique supplémentaire et (*iii*) SliceM comprend les mêmes ressources que SliceL mais avec un registre à décalage et une interface 64-bits vers les blocs RAM [Xilb].

Nous souhaitons dans cette partie étudier le portage de notre méthode sur une technologie 65nm, ainsi que l'impact de l'implémentation physique de ressources sur la précision des modèles. Le SecretBlaze est toujours considéré comme un cas d'étude, et fonctionne à une fréquence deux fois plus élevée (50MHz) pour exécuter les trois applications : AES, DES et NoP. Les données ont été extraites à partir de notre flot (présenté dans la Section 3.5) pour une période d'échantillonnage égale à 1µs.

Dans la Figure 3.24, nous avons tracé en noir la puissance dynamique de référence, en gris l'estimation du modèle linéaire entrainé précédemment sur l'Atlys en utilisant 3 signaux et en rouge les estimations du modèle linéaire ajusté en nous basant sur les mêmes signaux. Nous avons ré-entrainé le modèle sur la base de données extraites pour le Virtex5 afin de calibrer et ajuster les coefficients du modèle. Nous constatons que le modèle ajusté est capable de suivre les variations de la puissance sur Virtex5 en se basant sur les mêmes signaux sélectionnés dans l'expérimentation précédente sur l'Atlys. Le coefficient de détermination R^2 et la moyenne de l'erreur relative sont comparables aux résultats précédents ; R^2 est égal à 0.843 et l'erreur est égale à 8.61% pour une granularité temporelle très fine (égale à 1µs). De la même manière que le modèle linéaire, les coefficients du réseau de neurones et MARS peuvent aussi être ajustés.

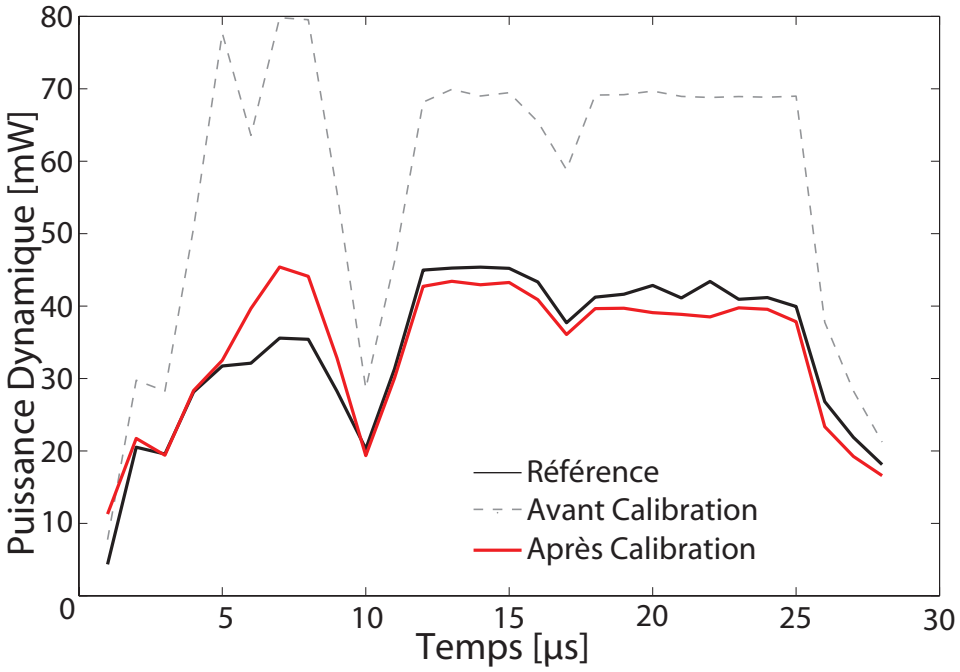


FIGURE 3.24 – Estimation de la puissance sur Virtex5.

D'après cette expérimentation, nous pouvons constater que les modèles construits restent valables pour une nouvelle implémentation, ainsi que l'analyse conduite sur la sélection de signaux stratégiques : une simple calibration du modèle obtenue à partir de notre méthode peut permettre d'estimer la puissance.

Bien que la validation ait été conduite sur deux FPGA ayant des architectures proches, notre méthode reste générale et indépendante de ces technologies. Le flot présenté précédemment dans la Section 3.5, peut-être porté sur d'autres types de circuits FPGA, mais aussi les ASICs. Toutefois, l'information au niveau RTL ne change pas pour un *benchmark* donné, il suffit de remplacer les outils d'estimation de la puissance par ceux des fournisseurs de circuits ASICs pour extraire des bases de données et appliquer notre méthode.

3.8 Conclusion

Dans ce chapitre, notre proposition d'instrumentation pour le monitoring de la puissance au niveau matériel a été développée. Une approche en amont a été proposée qui introduit un ensemble de techniques issues du domaine de la fouille de données pour l'analyse de données extraites des différents niveaux d'abstractions à partir du flot de conception, et ce afin de définir une solution optimale en terme de coût et de précision. Dans cette optique, nous avons réalisé une instrumentation complète du flot de conception dans le but d'extraire les données à différents niveaux d'abstraction. L'ensemble des méthodes supervisées sont déployées et comparées dans le but d'identifier l'information pertinente qui corrèle avec la puissance du circuit.

Les commutations enregistrées à niveau « transfert de registre » fournissent un indicateur fiable et peuvent être observées facilement en ligne par de simples compteurs d'évènements implémentés sur quelques signaux stratégiques. La problématique principale concernait l'intégration de ces dispositifs dans un circuit complexe avec des milliers de signaux internes. Nous avons étudié en premier lieu le problème d'emplacement de ces compteurs. Ensuite, nous avons présenté une méthode globale pour sélectionner les signaux stratégiques les plus importants à observer. Nous avons proposé dans un second temps plusieurs modèles pour l'estimation en ligne de la puissance en fonction de l'activité de commutation à surveiller sur ces signaux.

Notre solution a été expérimentée sur un cas d'étude constitué d'un système sur puce à base d'un processeur 32 bits implémenté sur FPGA. Grâce à notre méthode, nous avons pu sélectionner un petit nombre suffisant pour un monitoring précis de la puissance. Les résultats montrent aussi la capacité des modèles à suivre les variations de la puissance à une résolution fine avec un faible coût par rapport aux solutions existantes : l'erreur de nos modèles est inférieure à 5% pour une résolution temporelle supérieure à 400µs et ce avec un coût en surface, en

énergie et en performance inférieur à 5%.

À la fin, une discussion est présentée sur l'extension de la méthode sur une nouvelle technologie. Dans le chapitre suivant, nous proposerons une optimisation de l'utilisation des ressources existantes dans les plateformes multi-cœurs pour mettre en œuvre un monitoring hybride de la puissance à granularité fine.

CHAPITRE 4

MONITORING DE LA PUISSANCE AU NIVEAU SYSTÈME

« Notre plus grande faiblesse est d'abandonner. La façon la plus sûre de réussir est toujours d'essayer une fois de plus » `break`

Thomas Edison

Sommaire

4.1 Introduction

Obtenir un monitoring à grain fin précis consiste à identifier les informations pertinentes à observer, et à mettre en place une infrastructure de monitoring à faible coût. Dans ce but, nous avons proposé dans le chapitre précédent une approche en amont qui consiste à analyser des données issues du flot de conception pour retrouver l'information de commutation qui corrèle avec la puissance du circuit. Ensuite, nous avons proposé une structure à faible coût basée sur des compteurs d'évènements pour observer en ligne ces commutations et estimer la puissance. Cependant, la plupart des systèmes intégrés disposent de compteurs de performance qui peuvent être exploités efficacement dans le but de réaliser un monitoring de la puissance. Dans une nouvelle optique, nous proposons dans ce chapitre une optimisation de l'utilisation de ces ressources pour mettre en œuvre une solution à granularité fine et un coût négligeable.

Ce que nous proposons dans ce chapitre est une nouvelle heuristique inspirée des techniques de fouille de données, qui permet de sélectionner les évènements de performance pertinents pour la modélisation de la puissance. Nous introduisons aussi un modèle robuste aux variations de la température externe prenant en compte les mécanismes d'adaptation (DVFS). Dans ce qui suit, nous commencerons par présenter les compteurs de performance ainsi que l'ensemble des méthodes existantes qui utilisent ces compteurs pour l'estimation de la puissance. Ensuite, nous nous pencherons sur l'explication de la méthode de sélection et les modèles de puissance. Un cas d'étude est enfin présenté pour valider notre approche et évaluer la solution proposée.

4.2 De l'évènement de performance au monitoring de la puissance

4.2.1 Compteur et évènement de performance

La plupart des systèmes intégrés possèdent une unité pour surveiller en ligne des informations qui représentent la performance du système (appelés évènements de performance) pour le débogage des applications [SGG+08], tels que le "*Performance Counter Monitor*" (PCM) d'Intel intégré dans la famille des processeurs Xeon et le PMU ("*Performance Monitoring Unit*") de chez ARM intégré dans les séries de processeurs CortexA Armv6/7. Ce composant est capable de compter des évènements au niveau matériel. La Figure 4.1 illustre un cluster générique qui intègre un PMU. On peut distinguer 2 types d'évènements matériels : (*i*) évènement local lié à des activités locales aux cœurs et au cache L1 et (*ii*) évènement partagé au niveau des ressources partagées par tous les cœurs à l'intérieur d'un cluster.

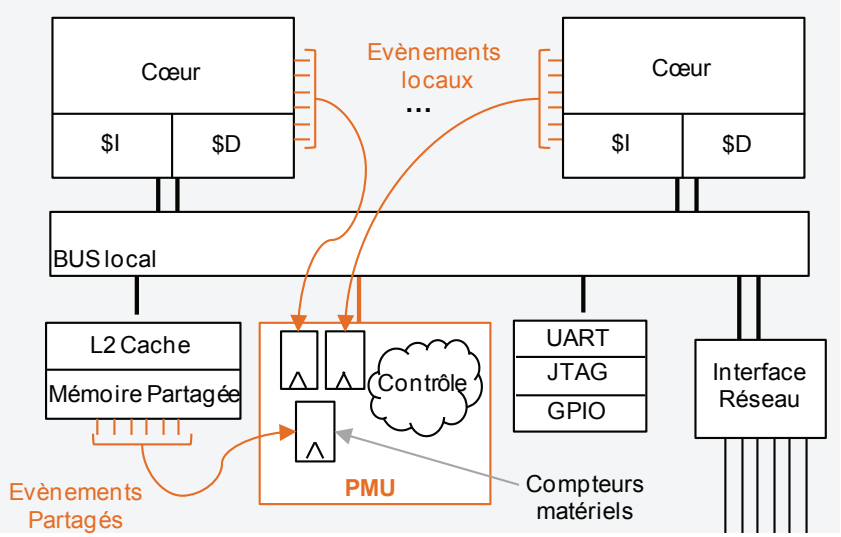


FIGURE 4.1 – Évènements observables au niveau local et partagé.

Un PMU intègre plusieurs registres configurables qui servent à compter le nombre d'occurrences de quelques évènements particuliers. Le nombre de compteurs par type d'évènement dépend de la version du matériel, par exemple, le PMUv2 dans les séries ARM CortexA9/15 intègre 6 compteurs par cœur pour surveiller des évènements locaux distincts, et 2 compteurs pour les évènements partagés. L'ensemble des registres est géré par une unité de contrôle qui permet la configuration de ces registres à partir du système d'exploitation ou bien à partir d'une interface de débogage externe. Elle est capable aussi d'interroger ces compteurs de manière périodique, permettant ainsi le monitoring en ligne des évènements configurés.

Il est également possible de relever des évènements logiciels de performance, tels que l'activité de l'ordonnanceur ou bien le nombre d'interruptions logicielles. La surveillance de ces évènements ne nécessite pas de circuit dédié ; les compteurs sont implémentés dans les couches logicielles et sont gérés par des outils dédiés tels que perf [dM10], Oprofile [Lev04] ou bien DS5 [ARM15a]. Un exemple d'évènement disponible dans la famille CortexA est illustré dans le tableau 4.1. Les évènements locaux observés au niveau du cœur sont généralement liés aux instructions, à l'horloge, ou bien à l'activité des instructions de branchement. La deuxième famille des évènements locaux sont ceux produits par les caches L1 de données et d'instructions. Dans la famille CortexA, le seul périphérique partagé à l'intérieur d'un cluster est le cache L2. Pour cela, tous les évènements partagés dépendent de l'activité du cache L2 tel que le nombre d'accès. Les évènements au niveau système correspondent à l'activité des routines dans les couches du système d'exploitation qui gèrent les cœurs, l'ordonnanceur ou bien l'interruption.

TABLEAU 4.1 – Évènements observables, au niveau local, partagé et système.

Ressource	Type	Nom	Description
Locale			
cœur	Branchement	BR_IMMED_RETIRED, BR_PRED, BR_MIS_PRED	Nombre d'instructions de branchement immédiat, prédit et mal-prédit.
	Horloge	CPU_CYCLES	Nombre de cycles d'horloge lorsque le CPU est actif
	Instruction	EXECUTED, FLOATING_-POINT, LOAD_STORE, MEMORY_READ_WRITE	Nombre d'instructions : exécuté, calcul virgule flottante, accès en lecture/écriture, accès en mémoire
Cache	Donnée	CACHE_ACCESS, DATA_-EVICTION, COHERENCY_-HIT, COHERENCY_MISS, PREFETCH_HIT	Nombre : d'accès à la cache de données, d'évictions de ligne, d'accès de cohérence en *hit* et *miss*
	Instruction	CACHE_ACCESS, CACHE_-REFILL, CACHE_TLB_RE-FILL	Nombre : d'accès à la cache d'instruction, recherche de ligne de cache, recharge du TLB
Partagée			
L2 Cache	Eviction	CastOUT	Eviction d'une ligne de cache L2
	Donnée	DRHIT, DRREQ, DWHIT, DWREQ, DWTREQ	Nombre d'accès au données, requêtes reçue, écritures *hit*, requêtes d'écriture avec ou sans éviction (*Write Through*)
	Prefetch	EPFALLOC, EPFHIT, EP-FRCVD	Nombre de pré-lectures : allouer, *hit*, reçue
	Instruction	IRHIT, IRREQ	Nombre d'accès en instruction *hit*, de requêtes
Système			
OS	Ordonnanceur	MIGRATION, SWITCH	Nombre de tâches qui ont été migrées de processeur, ou ordonnancées sur un même cœur
	CPU	CONTENTION_WAIT, CONTENTION_IO_WAIT	Nombre de cycles d'horloge en attente de dépendance ou bien de données sur les entrées/sorties
	Interruption	IRQ, SoftIRQ	Nombre d'interruptions matérielles et logicielles
	Mémoire	USED, FREE	Espace mémoire RAM : utilisé et libre

4.2.2 Problématiques et Enjeux

Les évènements de performance représentent une source d'information intéressante pour les mécanismes de gestion de la puissance et de la température. Celle-ci est facilement accessible par les couches du système d'exploitation, ce qui simplifie la mise en place des techniques d'adaptation. Nous nous intéressons ici à l'exploitation de ces évènements pour la modélisation de la puissance.

En fait, il existe un grand nombre d'évènements au niveau système, qui peut atteindre des centaines ou bien milliers d'unités [SGG+08]. Donc, si on veut relever ces évènements, on va avoir à faire face à un problème de gestion d'une grande quantité de données. Il est donc nécessaire de repérer parmi ces évènements ceux qui caractérisent au mieux la consommation du circuit afin de produire une méthode d'estimation en ligne précise et peu coûteuse.

Le problème de la sélection de variable pour la modélisation est bien connu dans la littérature : il existe plusieurs méthodes telles que les méthodes de sélection d'attributs (tel que nous l'avons abordé dans le chapitre précédent), les analyses en composantes principales (ACP ou PCA en anglais), les techniques de réduction de dimensionalité, *etc.* Cependant, à cause du nombre limité de compteurs matériels, il n'est pas possible de mesurer simultanément l'occurrence de tous les évènements disponibles. En d'autres termes, il n'est pas possible de construire une base de données avec une échelle temporelle unique contenant tous les évènements qui sont issus de plusieurs extractions différentes. À titre d'exemple, il existe 4 registres pour surveiller 160 évènements configurables distincts dans le processeur AMD Opteron [ZA12], 6 compteurs par cœur pour 62 évènements locaux distincts, et 2 compteurs pour 15 évènements du type partagés dans le PMUv2 dans la série d'ARM CortexA. Par conséquent, l'utilisation des méthodes standards est assez limité, ce qui nécessite la mise en place d'une méthodologie spécifique pour garantir une meilleure sélection des évènements.

La sélection des évènements de performance pour la modélisation de la puissance a été peu abordé dans la littérature. Les évènements ont été choisis d'une manière empirique dans [BWeWK03], en se basant sur des pré-connaissances du comportement du système. En outre, les auteurs de [BVLJ05] sélectionnent les évènements en utilisant la corrélation de Pearson. Dans [LWT+12], elle est basée sur le calcul du coefficient de corrélation du rang de Spearman. Ces critères statistiques ne tiennent pas compte de l'interaction entre les évènements sélectionnés. Deux évènements parfaitement corrélés avec la puissance peuvent être redondants, dans le sens où leur combinaison n'apporte aucune information supplémentaire. Une autre méthode de projection sur la base de l'analyse ACP a été proposée dans [ZA12] : celle-ci tient compte de la relation entre les évènements avant la sélection, pour le processeur Dell PowerEdge Opteron. C'est une approche intéressante mais qui reste assez spécifique, puisque les évènements sont sélectionnés pour quatre applications séparément, sans prise en compte de

l'exécution multi-tâche potentielle.

Afin de résoudre ce problème, nous proposons une méthode efficace pour la sélection des évènements de performance inspirée des algorithmes de fouille de données. L'originalité vient de la prise en considération des interactions entre les évènements des deux niveaux matériels et logiciels avant la sélection. De plus, plusieurs modèles sont introduits pour estimer la puissance, et ce afin de mettre en œuvre un monitoring hybride précis et à faible coût qui s'appuie sur les informations des deux niveaux d'abstractions.

4.3 Nouvelle méthode d'analyse de données du PMU

4.3.1 *PESel* : Sélection des évènements de performance

Dans cette partie, nous présentons une méthode de sous-espaces pour la sélection des évènements de performance dans les circuits intégrés. La méthode repose sur la définition des éléments suivants :

- N_e est le nombre total d'évènements et qui est égal à la somme des évènements locaux (N_{le}), partagés (N_{se}) et systèmes (N_{sse}).
- Q est l'ensemble de tous les évènements, avec $Card(Q) = N_e$.
- N_{cl} et N_{cs} représentent le nombre de compteurs matériels pour les évènements locaux et partagés respectivement.
- $N_{samples}$ est le nombre d'instances de mesure (nombre d'échantillons).
- E_i est l'ensemble des valeurs enregistrées pour un évènement particulier e_i, avec $i \in \{1, ..., N_e\}$, et $Card(E_i) = N_{samples}$.
- P est la valeur de la puissance totale enregistrée pendant les $N_{samples}$ échantillons.

La méthode proposée est itérative et utilise une nouvelle heuristique appelée "*Performance Events Selection*" (PESel) pour la sélection des évènements pertinents afin de modéliser la puissance. Une itération de la méthode est représentée dans la Figure 4.2, où k représente le numéro de l'itération (ou évènements sélectionnés). À la fin de l'itération k, un évènement (e_k) est sélectionné : il peut être un des trois types d'évènements. Soit S_k l'ensemble des évènements solution obtenue à la fin de l'itération k et R_k l'ensemble des évènements restants : R_k est équivalent à la différence des ensembles Q et S_k, $R_k = Q \setminus S_k$. Le but final de la recherche est de retrouver N_{cl} évènements locaux, N_{cs} partagés pour configurer les compteurs matériels et un nombre minimum N_{csys} d'évènements systèmes.

Nous rappelons qu'il n'est pas possible de construire une base de données avec une échelle temporelle unique contenant tous les évènements. C'est la raison pour laquelle nous avons

adapté la recherche heuristique avec l'évaluation CFS (approche *filter* présentée précédemment dans le Chapitre 3 Section 3.3) dans PESel pour opérer sur des sous-ensembles extraits à partir de plusieurs enregistrements.

Prenons un exemple : le processeur ARM CortexA9 contient 6 compteurs pour configurer des évènements locaux ($N_{cl} = 6$) et 2 autres compteurs pour configurer des évènements partagés ($N_{cs} = 2$). Le nombre total d'évènements disponible dans le Cortex A9 est égal à $N_e = N_{le} + N_{se} + N_{sse} = 62 + 15 + 13 = 90$ évènements. Donc, si l'on souhaite étudier l'impact de chaque évènement sur la puissance, il faut réaliser : $\frac{N_{le}}{N_{cl}} = \frac{62}{6} = 11$ enregistrements en configurant différemment les évènements locaux, dont $\frac{N_{se}}{N_{cs}} = \frac{15}{2} = 8$ pour les évènements partagés. Notre méthode recherche dans ces enregistrements un sous-ensemble d'évènements solution suffisants pour modéliser la puissance.

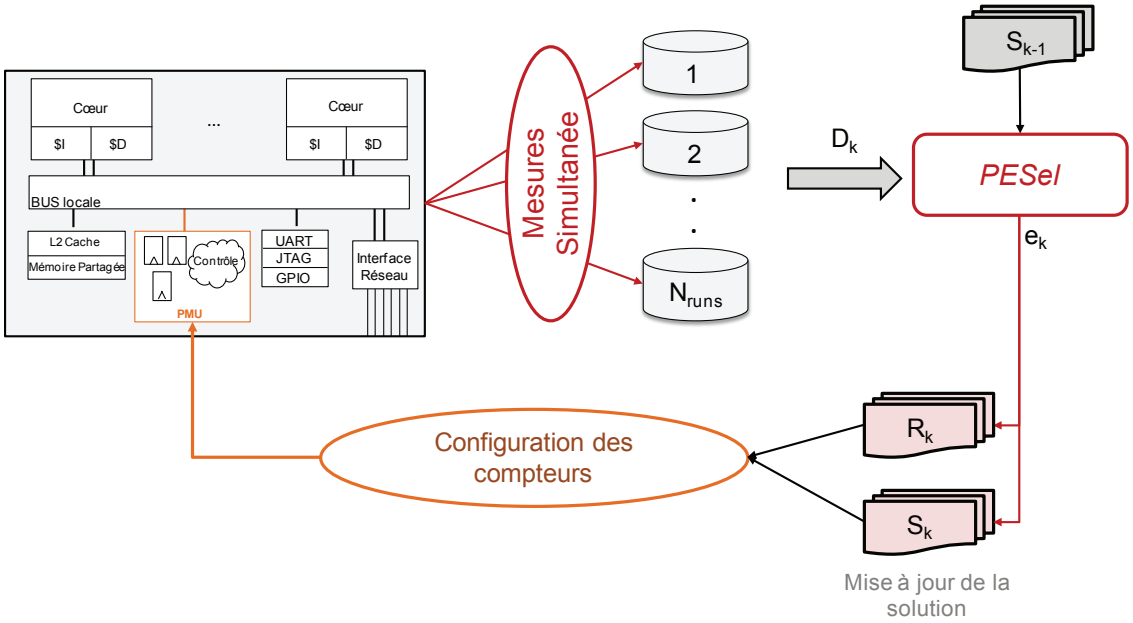


FIGURE 4.2 – Représentation de la méthode de sélection des évènements de performance pour la modélisation de la puissance.

Les deux étapes suivantes sont exécutées à chaque itération de la méthode :

Mesure Simultanée

À l'itération k, nous enregistrons simultanément le nombre d'occurrences de chaque membre de l'ensemble R_{k-1} avec tous les membres de la solution S_{k-1} de l'itération $k-1$. En principe, les évènements sélectionnés dans S_{k-1} occupent quelques compteurs matériels parmi les N_{cl} et N_{cs} disponibles. Par conséquent, les compteurs restants sont utilisés pour configurer différemment les évènements de R_{k-1} en utilisant le même jeu d'instructions. À cet effet, plusieurs opérations d'enregistrement sont nécessaires pour finir cette étape à l'itération k. Après chaque

opération, un ensemble C contenant les évènements avec la puissance consommée peut être construit. Soit N_{runs} le nombre total d'opérations, D_k est donc un ensemble de sous-ensembles construit à chaque enregistrement ($D_k = \cup C_i$, $i = \{1, ..., N_{runs}\}$).

Application de l'algorithme *PESel*

La deuxième étape de la méthode consiste à appliquer l'algorithme proposé PESel pour repérer l'évènement le plus pertinent parmi les évènements restants dans l'ensemble R_{k-1}. L'algorithme (2) illustre le pseudo-code de PESel. Il évalue tous les candidats construits à partir de la combinaison de l'union de chaque évènement $\{e\}$ de R_{k-1} et la solution S_{k-1}.

input : S_{k-1} solution à l'itération $k-1$ et D_k ensemble des opération d'enregistrement simultanées
output : e_k meilleur évènement trouvé à l'itération k

```
1  M_best = 0 ; e_k = {};                                   // Initialisation des variables
2  /* Parcourir tous les ensembles extraits des enregistrements                    */
3  for C in D_k do
4      C_cand ← C \ S_{k-1};                                // Retrouver les évènements candidats
5      /* Évaluer chaque évènement séparément                                      */
6      for e in C_cand do
7          L_e ← S_{k-1} ∪ {e};          // tester l'évènement candidat avec la solution précédente
8          M = Merit(L_e);                  // mesurer le mérite du sous-ensemble construit
9          /* Rechercher le meilleur évènement                                     */
10         if M > M_best then
11             M_best = M ;
12             e_k ← e ;
13         end
14     end
15 end
16 return e_k;                                              // Fin de la recherche
```

Algorithm 2: Algorithme PESel.

$$M = \frac{p \times \bar{R}_{E_i,P}}{\sqrt{p + p \times (p-1) \times \bar{R}_{E_i,E_j}}} \tag{4.1}$$

Le critère d'évaluation correspond au mérite de l'évaluation *filter* CFS représenté dans l'équation (4.1), où p est le nombre d'évènements dans le sous-ensemble candidat, E_i et E_j correspondent à deux évènements dans le sous-ensemble candidat, $\bar{R}_{E_i,P}$ est la moyenne de la corrélation entre tous les évènements E_i et la puissance P, et $\bar{R}_{E_i,E_j}$ est la moyenne de la corrélation entre les évènements dans un sous-ensemble candidat. L'évènement qui appartient au candidat qui aura le plus grand mérite sera considéré comme solution à l'itération k. Il sera donc ajouté à l'ensemble solution et enlevé de l'ensemble de recherche R_{k-1} : $S_k \leftarrow S_{k-1} \cup \{e_k\}$, $R_k \leftarrow R_{k-1} \setminus \{e_k\}$, et $k = k+1$.

Exemple

Afin de mieux comprendre le fonctionnement de la méthode, nous illustrons dans la Figure 4.3 un exemple d'application. On suppose un PMU disposant des caractéristiques suivantes : 4 compteurs matériels pour configurer 20 évènements locaux distincts et 2 autres pour 8 partagés. Il dispose aussi de 4 évènements au niveau système qui ne nécessitent pas de compteurs matériels.

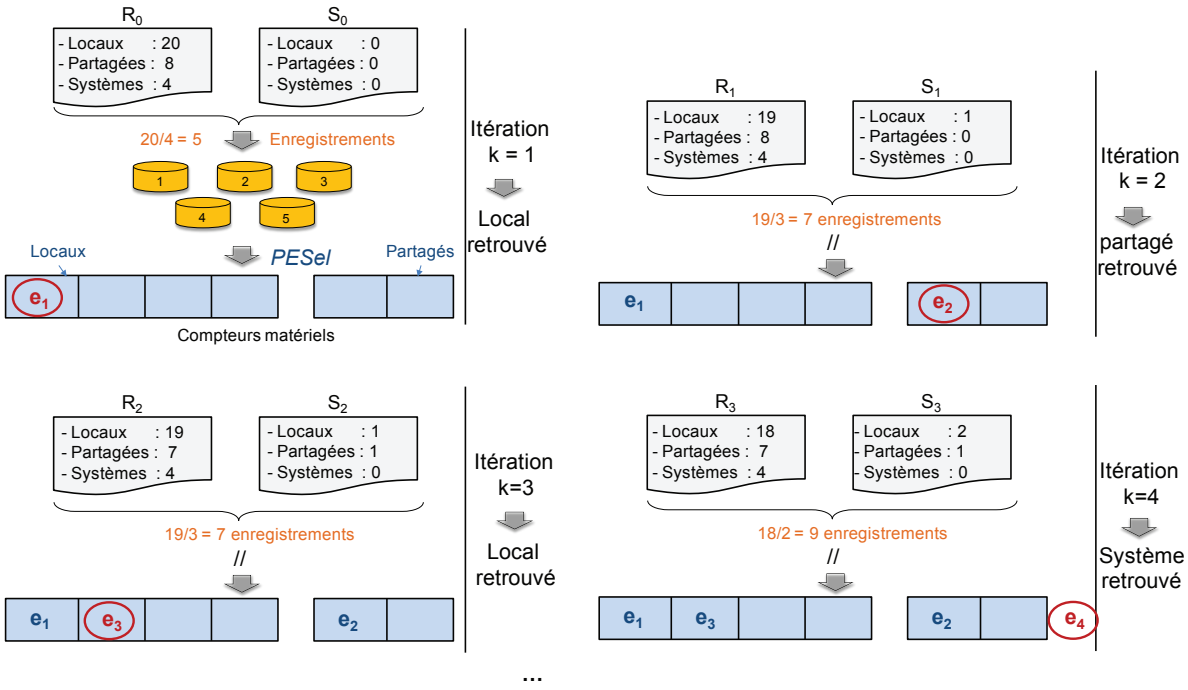


FIGURE 4.3 – Représentation de la méthode de sélection des évènements de performance.

À la première itération ($k = 1$), on aurait besoin de $\frac{20}{4} = 5$ enregistrements, dont $\frac{8}{2} = 4$ qui configurent différemment les évènements partagés. Dans cette itération, l'application de l'algorithme PESel sur les 5 enregistrements permet de retrouver l'évènement ayant la plus forte corrélation avec la puissance ; l'équation du mérite devient égale à un simple calcul du coefficient de corrélation pour $p = 1$. À la fin de cette itération, l'évènement retrouvé e_1 va être déplacé de l'ensemble d'évènements restant R_0 vers la solution S_0 pour constituer deux nouveaux ensembles pour l'itération suivante (R_1 et S_1).

Dans la deuxième itération ($k = 2$), le nombre d'enregistrements nécessaires est égal au nombre d'évènements restants divisé par le nombre de compteurs disponibles ($\frac{19}{3} = 7$). PESel évalue chaque évènement de R_1 ayant 19 locaux, 8 partagés et 4 systèmes, avec la solution retrouvée à l'itération précédente S_1. La solution va correspondre donc à ce qui aura le mérite le plus élevé, dans cet exemple supposé que c'est un évènement associé aux ressources partagées. De la même manière, la méthode opère à l'itération $k = 3$ pour retrouver, par exemple, l'évènement local e_3. Les itérations se terminent lorsque l'algorithme retrouve 4 évènements

locaux et 2 partagés et un nombre minimum d'évènements systèmes, qui permettent à l'aide d'un modèle d'estimer en ligne la puissance.

L'exploitation des ressources existantes permet de mettre en place une méthode de monitoring à coût négligeable. L'avantage de notre méthode réside dans le fait qu'elle ne nécessite pas une synchronisation de tous les enregistrements sur une même échelle temporelle, et qu'elle aboutit à une solution identique à celle de l'approche itérative CFS dans le cas où la synchronisation est possible. Cette méthode sera complétée par un modèle afin d'estimer en ligne la puissance en fonction des évènements sélectionnés.

4.3.2 Modélisation de la puissance totale consommée

Dans la section précédente, nous avons proposé un algorithme qui permet de sélectionner : N_{cl} évènements locaux, N_{cs} partagés et N_{csys} d'évènements systèmes pour la modélisation de la puissance. Dans cette section, nous présentons deux modèles pour estimer la puissance totale consommée par les systèmes multi-cœurs prenant en considération la variation de la fréquence de fonctionnement.

Modèle Linéaire

Nous rappelons que la puissance totale consommée peut être représenté par :

$$P_{totale} = \alpha.C.V^2_{dd}.f + V_{dd}.I_{leakage} + V_{dd}.I_{cc} \tag{4.2}$$

où α est un facteur lié à l'activité interne, C est la capacité de commutation, V_{dd} et f sont la tension d'alimentation et la fréquence respectivement, et $I_{leakage}$ et I_{sc} sont le courant de fuite et de court-circuit respectivement. Dans un premier temps, nous supposons constants tous les paramètres technologiques qui affectent $I_{leakage}$. En outre, I_{cc} est utile pour la détection de la puissance pic et est généralement négligé dans la modélisation de la puissance moyenne. Elle est due à un phénomène rapide de court-circuit au niveau des transistors et sa participation à la puissance moyenne dans une période de temps est négligeable.

D'autre part, la fréquence f et la tension d'alimentation V_{dd} sont modifiées simultanément ensemble par une technique de DVFS pour changer le mode de fonctionnement du système. Selon [Gha12], le terme $V^2_{dd}.f$ varie selon f^β avec β entre 2 et 3. Par conséquent, nous pouvons exprimer la puissance totale par :

$$P_{totale} = a_1.\alpha.C.f^\beta + a_2 \tag{4.3}$$

où, a_1 et a_2 sont deux constantes.

D'après [MBM+09] et [WMW09], la relation entre la puissance et la fréquence peut être considérée comme linéaire. En négligeant la dépendance entre l'activité α et la fréquence f et en supposant qu'elles affectent indépendamment la puissance, l'équation (4.3) peut être approximée par :

$$\mathrm{P}_{totale} = k_0 + k_\alpha.\alpha. + k_f.f^\beta \tag{4.4}$$

où k_0, k_α et k_f sont des constantes. Cette hypothèse est aussi posée dans plusieurs travaux existants pour la modélisation de la puissance. Par exemple, les auteurs de [Gha12] ont négligé l'interaction entre la fréquence et la température pour représenter linéairement la puissance en fonction de la fréquence et de la température de la puce.

Cependant, l'activité globale α est représentée ici par un ensemble d'évènements de performance sélectionnés par PESel. Nous rappelons que les N_{cl} évènements locaux sont observés par cœur à l'aide du PMU, lorsque les N_{cs} partagés et les N_{csys} systèmes sont au niveau de tout le système. Ainsi, le modèle linéaire de la puissance en fonction des évènements de performance est approximé par :

$$\mathrm{P}_{totale} = k_0 + \sum_{i=0}^{\mathrm{N}_{core}} \sum_{j=0}^{\mathrm{N}_{cl}} k^l_{i,j}.e^l_{i,j} + \sum_{j=0}^{\mathrm{N}_{cs}} k^s_j.e^s_j + \sum_{j=0}^{\mathrm{N}_{csys}} k^{sys}_j.e^{sys}_j + k_f.f^\beta \tag{4.5}$$

où, k_0 est une constante qui correspond à la puissance "idle", $k^l_{i,j}$ est le coefficient de la $j^{ième}$ évènement locale $e^l_{i,j}$ surveillé sur le cœur i qui représente le poids de cet évènement sur la variation de la puissance. k^s_j et k^{sys}_j sont les coefficients du $j^{ième}$ évènement partagé e^s_j et système e^{sys}_j respectivement.

Si les cœurs par cluster sont homogènes, la contribution d'un évènement local sur la puissance consommée par cœur est identique à la contribution du même évènement sur les autres cœurs. Autrement dit, chaque évènement local e^l_j est associé à un coefficient identique k^l_j quel que soit le cœur i. Nous pouvons ainsi factoriser l'équation (4.5) en agrégeant l'occurrence des évènements locaux. Par exemple, une occurrence d'un "miss" du cache L1 sur le cœur 0 ou le cœur 1 de la plateforme ARM Cortex-A9 va avoir le même impact sur l'augmentation de la puissance. D'où, l'équation linéaire de la puissance qui devient égale à :

$$\mathrm{P}_{totale} = k_0 + \sum_{j=0}^{\mathrm{N}_{cl}} k^l_j.(\sum_{i=0}^{\mathrm{N}_{core}} e^l_{i,j}) + \sum_{j=0}^{\mathrm{N}_{cs}} k^s_j.e^s_j + \sum_{j=0}^{\mathrm{N}_{csys}} k^{sys}_j.e^{sys}_j + k_f.f^\beta \tag{4.6}$$

Réseau de neurones

Nous avons vu que la puissance peut être considérée comme variant linéairement en fonction de l'activité logique α. Toutefois, l'activité est représentée sous forme d'un ensemble d'évènements de performance, la relation entre les évènements de performance et la puissance peut ne pas être linéaire. Donc, une approximation linéaire du modèle peut ne pas toujours être précise, surtout si l'on souhaite utiliser dans la modélisation très peu d'évènements. À cet effet, nous proposons dans cette partie une architecture de réseau de neurones pour la modélisation de la puissance.

Dans ce modèle, nous supposons la prise en compte de tous les évènements de performance sélectionnés par PESel, la fréquence de fonctionnement et la température externe de la puce. La Figure 4.4 montre le réseau de neurone perceptron mono-couche proposé pour la modélisation de la puissance. Chaque entrée est associée à un neurone qui conduit leurs valeurs vers la couche intermédiaire. Chaque neurone j (avec $j \in \{0..M\}$) de la couche intermédiaire, un poids w_{ij} à chaque entrée i avant de réaliser la somme pondérée des résultats. La dernière étape consiste à filtrer le résultat de la somme par une fonction de transfert (*e.g.* fonction sigmoid). Le neurone de la couche de sortie réalise la même opération sur les sorties de la couche intermédiaire pour fournir une estimation de la puissance.

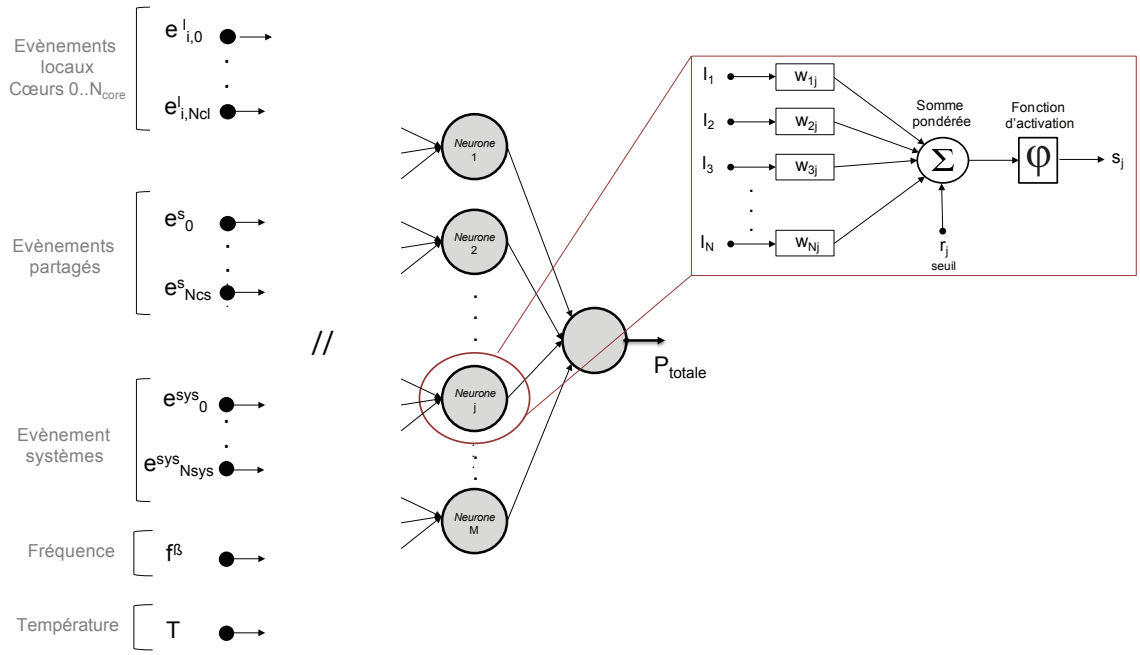


FIGURE 4.4 – Architecture du réseau de neurones pour la modélisation de la puissance.

4.4 Évaluation

Dans cette partie, nous allons présenter les résultats de l'implémentation de notre méthode de monitoring de la puissance consommée par une plateforme multi-cœur. Nous allons d'abord présenter le protocole expérimental puis nous nous pencherons sur les résultats de la sélection des évènements de performance à l'aide de PESel pour la modélisation de la puissance en présentant aussi une comparaison avec les méthodes existantes.

4.4.1 Description du cas d'étude

Nous avons choisi d'expérimenter notre méthode sur le MPSoC Snowball qui a été intégré dans plusieurs téléphones portables comme le Galaxy SII et dans diverses tablettes tactiles comme l'Asus "Eee Pad Transformer". Pour ce faire, nous avons mis en place un montage expérimental pour l'extraction de données issues du PMU et pour la validation des estimations des modèles de la puissance. Tous ces points seront abordés en détail dans les sections suivantes.

Snowball

La plateforme de développement SKY-S9500-ULP-CXX snowball PDK [Pet11] est utilisée dans nos expériences. Les caractéristiques de la carte sont illustrées dans le tableau 4.2. Snowball contient principalement le processeur NovaTMA9500 de ST-Ericsson qui intègre le double cœur Cortex-A9 (Figure 4.5). C'est un microprocesseur multi-cœur conçu par la société ARM et qui possède un jeu d'instructions ARMv7-A avec un pipeline superscalaire permettant l'exécution *out-of-order* des instructions. Le processeur Cortex-A9 intègre aussi la dernière version de l'unité de surveillance de la performance PMUv2 conçue par ARM. Cette unité est utilisée pour le débogage et la surveillance en temps réel de l'activité des applications exécutées par les deux cœurs. PMUv2 contient 7 compteurs matériels par cœur dont : 6 configurables dédiés à la surveillance des évènements du type local et 1 non-configurable qui compte le nombre de cycles du processeur. La surveillance des évènements partagés est réalisée à travers deux compteurs matériels configurables intégrés également dans le PMUv2.

En plus du processeur, Snowball intègre une mémoire interne *eMMC*, une interface pour mémoire SD, interface Ethernet et plusieurs sorties telles qu'une sortie Vidéo, interface de debug JTAG, port USB série, *etc.* Le *governor*, quant à lui, est en charge de la gestion dynamique de la fréquence et de la tension en fonction de la charge des processeurs afin d'optimiser la puissance consommée par le Cortex-A9.

TABLEAU 4.2 – Spécification de la plateforme SKY-S9500-ULP-CXX snowball PDK.

Caractéristique	Description
A9500	Processeur double-cœur ARM Cortex
eMMC	8GBytes, Carte multimédia embarquée
LPDDR2	8GBytes, mémoire SDRAM PoP
mémoire SD	4GBytes, carte MicroSD
Autres	Sortie Vidéo, 10/100 Ethernet, LEDs, interface USB série, JTAG, ...

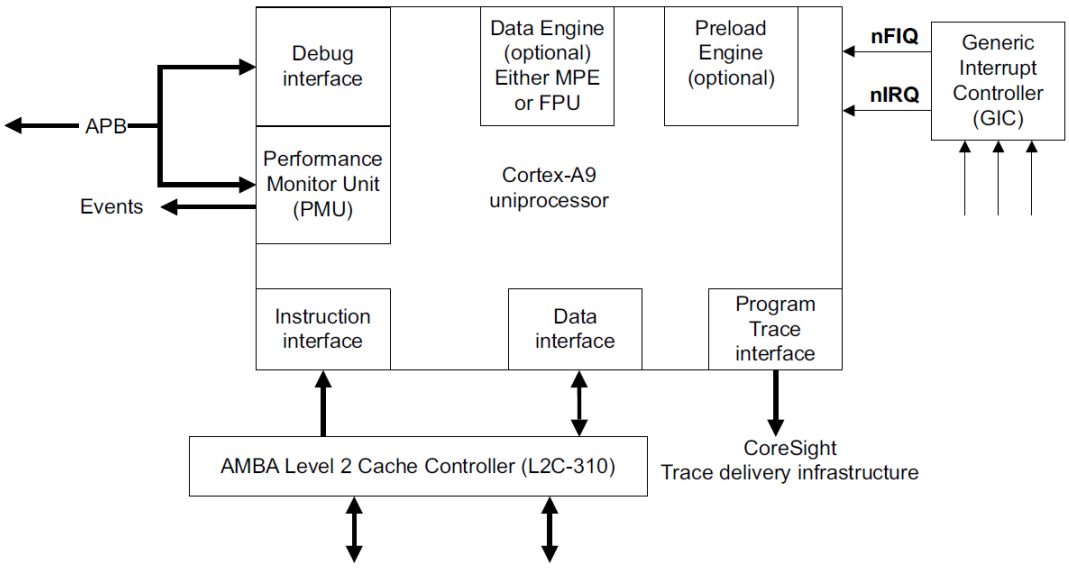


FIGURE 4.5 – Schéma de l'architecture du processeur Cortex-A9 [ARM10a].

Benchmarks

En plus de la plateforme matérielle, nous avons choisi un ensemble de *benchmarks* : Mibench [GRE+01], Linpack [DBMS79] et Whetstone [CW76] exécutés par le processeur afin d'évaluer des activités diverses et variées. Ces applications sont illustrées dans le tableau 4.3 et appartiennent à plusieurs catégories telles que les applications pour le test de la performance, sécurité, réseau et télécommunication, *etc.* Elles sont regroupées en deux en-sembles : (*i*) le premier est utilisé pour l'entrainement des modèles et (*ii*) le deuxième pour la validation et la mesure de l'erreur.

TABLEAU 4.3 – Ensemble des Benchmarks.

Applications		**Description**
entrainement	***test***	
Whetstone, Linpack	mxm	Test de la performance et gestion de donnée
Basicmath, bitcnt, Qsort	Susan	Contrôle industriel et automobile
JPEG		Consommateur
Stringsearch		Programme office
Sha	Blowfish	Sécurité
Dijkstra, Adpcm, CRC32	Patricia, FFT	Réseau et télécommunication

Extraction des données

Afin d'extraire les données pour le traitement, nous avons mis en place un montage expérimental illustré dans la Figure 4.6. La distribution Linux Linaro 12.02 est d'abord installée sur la carte permettant ainsi de lancer les *benchmarks* à travers des terminaux. Dans ce montage, nous avons utilisé l'outil ARM DS-5 v5.21 [ARM15a] pour la collection de données. En principe, l'accès en ligne au PMU nécessite l'insertion d'une couche au système d'exploitation tournant sur la carte. Pour ce faire, nous avons compilé et installé le module kernel "*Gator*" de DS-5 sous Linaro. Une fois ce module installé, l'application "*Gator Daemon*" utilise les routines définies par le module "*Gator*" pour accéder à la contenue des compteurs. L'outil ARM "*Streamline Performance Analyser*" installé sur un PC externe assure la communication avec "*Gator Daemon*" à travers la connexion Ethernet et collecte périodiquement les données des compteurs permettant ainsi une observation en ligne de l'exécution de la plateforme.

D'autre part, l'extraction des valeurs de la puissance consommée par toute la carte est effectuée à l'aide d'une sonde externe de courant (*Energy probe*) [ARM15b]. Cette sonde permet de

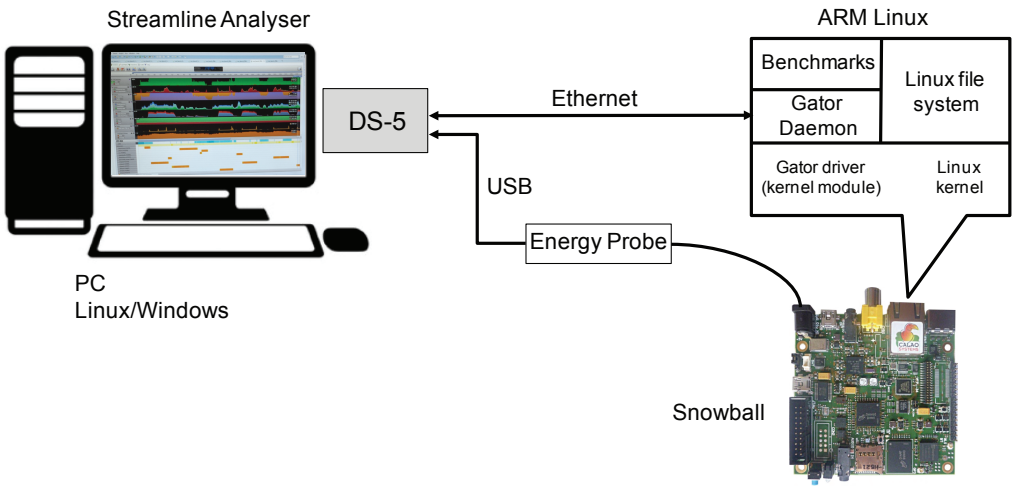


FIGURE 4.6 – Montage expérimentale.

mesurer le courant et la tension à travers une résistance dédiée sur 3 voies simultanément avec une erreur inférieure à 5%. À l'aide d'un convertisseur analogique-numérique, les valeurs de la puissance consommée sont ensuite transmises à DS-5 à travers la connexion USB 2.0 avec une résolution temporelle égale à 100 µs. Enfin, la synchronisation des valeurs de la puissance avec les évènements de performance collectés du PMU est gérée par l'outil "*Streamline Analyser*" pour une résolution temporelle maximale de 1 ms.

Grâce à ce montage expérimental, il devient possible de configurer les compteurs avec un ensemble d'évènements, de lancer un ensemble d'applications et enfin d'effectuer un enregistrement pour produire des données prêtes pour le traitement.

4.4.2 Sélection des évènements de performance

Dans l'architecture de Cortex-A9, le PMUv2 contient 7 compteurs matériels dont : 6 configurables ($N_{cl} = 6$) pour les évènements locaux et 1 non-configurable dédié au cycle d'horloge du processeur. Il en intègre aussi 2 autres ($N_{cs} = 2$) pour les évènements partagés. D'autre part, il existe : 62 évènements locaux (N_{le}), 15 partagés (N_{se}) et 13 systèmes (N_{sse}). L'enjeu principal est d'extraire un sous-ensemble d'évènements pour un monitoring précis de la puissance consommée par le système.

Dans cette section, nous souhaitons évaluer la méthode de sélection PESel proposée précédemment dans la section 4.3.1. Nous rappelons que PESel est un algorithme itératif qui opère sur un ensemble d'enregistrements pour retrouver les évènements pertinents pour la modélisation de la puissance. Dans cette expérience, nous fixons la fréquence des cœurs à la fréquence maximale disponible (1 GHz) afin d'augmenter l'impact de l'activité sur la variation de la puissance. Chaque enregistrement contient environ 4000 instances pour une période d'échantillon-

nage égale à 100 ms. En outre, le temps pris pour effectuer un enregistrement est d'environ 10 minutes.

Chaque itération de PESel permet de retrouver l'évènement le plus pertinent (local, partagé ou système). Nous avons réalisé au total 11 itérations de l'algorithme pour trouver les 6 évènements locaux et les 2 partagés pour configurer les compteurs matériels. À la première itération ($k = 1$), il fallait réaliser $\frac{N_{le}}{N_{cl}} = \frac{62}{6} = 11$ opérations d'enregistrement pour configurer différemment tous les évènements disponibles. PESel a sélectionné le premier évènement e_1 qui correspond au nombre de cycles de processeur. À la deuxième itération (k = 2), nous allons fixer l'évènement e_1 dans tous les enregistrements réalisés pour cette itération. Puisque cet évènement ne nécessite pas un compteur configurable, le nombre d'enregistrements à l'itération $k = 2$ est toujours égal à 11 ; les 6 compteurs sont encore disponibles. Ce nombre varie lorsqu'un évènement associé à un compteur configurable (local ou partagé) est retrouvé par PESel. Au final, nous avons réalisé successivement : 11, 11, 11, 11, 13, 15, 20, 20, 29, 29 et 57 opérations d'enregistrements qui correspondent aux 11 itérations. L'ensemble des évènements sélectionnés par PESel est illustré dans le tableau 4.4.

TABLEAU 4.4 – Évènements sélectionnés par PESel.

Ev.	Nom	Type	Description
e_1	Clock :Cycles	Locale	Nombre de cycles d'horloge des processeurs
e_2	Memory :Used	Système	Quantité de mémoire réservée
e_3	Memory :Free	Système	Quantité de mémoire disponible
e_4	Cache :Inst dependent stall	Locale	Nombre de cycles d'horloge du cache d'instruction
e_5	Stalls :Data main TLB miss	Locale	Nombre de miss de données du TLB
e_6	Instruction :Load/Store	Locale	Nombre d'instructions de lecture/écriture
e_7	L2 Cache :Data Read Hit	Partagé	Nombre de requêtes vers la cache L2
e_8	Instruction :Memory Read	Locale	Nombre d'instructions de lecture en mémoire principale
e_9	L2 Cache :Data Read Request	Partagé	Nombre de requêtes de lecture de donnée vers la cache L2
e_{10}	L1 Cache :Coherency miss	Locale	Nombre de miss dues à la cohérence de cache L1
e_{11}	Instruction :Floating Point	Locale	Nombre d'instructions à virgule flottante

Afin de comparer notre méthode avec les méthodes existantes, nous avons implémenté trois méthodes de sélection que l'on trouve dans la littérature : (*i*) selon le critère de corrélation de Pearson [BVLJ05], (*ii*) selon le rang de Spearman [LWT+12], et (*iii*) selon l'analyse ACP [ZA12]. Ainsi, la meilleure solution est celle qui produit le modèle de puissance le plus précis. À cet effet, nous avons utilisé un modèle linéaire (LM) pour modéliser la puissance comme cela était proposé par les méthodes existantes [LWT+12, ZA12]. En plus du modèle linéaire, nous avons aussi expérimenté notre méthode de sélection avec le réseau de neurones (NN) proposé précédemment dans la section 4.3.2. Dans cette expérimentation, nous allons varier le nombre d'attributs sélectionnés par toutes les méthodes. Ensuite, nous entrainerons les modèles sur les données d'entrainement.

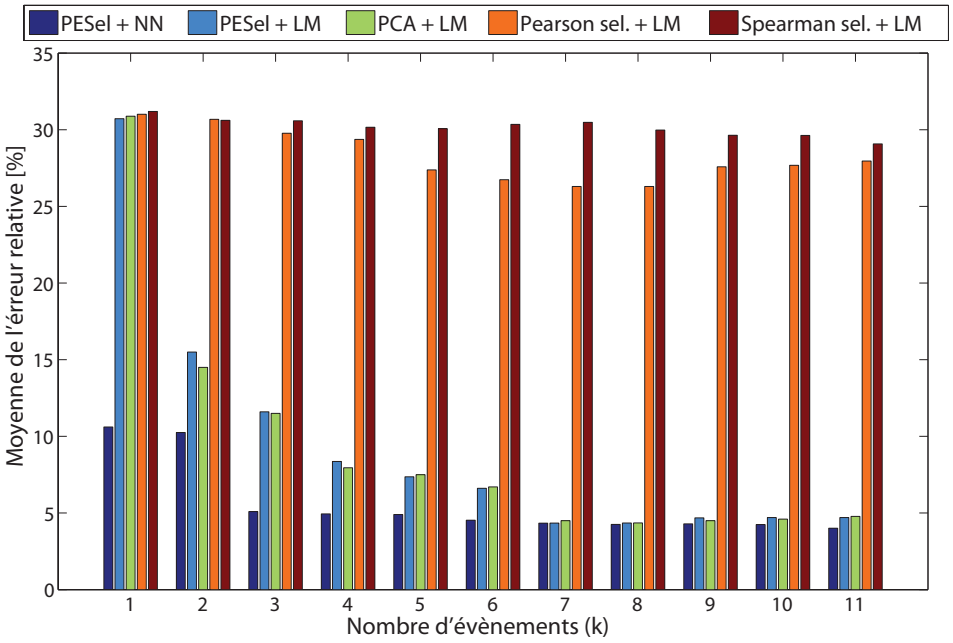


FIGURE 4.7 – La moyenne de l'erreur relative mesurée pour les modèles LM et NN construits à partir des évènements sélectionnés par les méthodes : *PESel*, ACP, Pearson et Spearman.

La Figure 4.7 compare la moyenne de l'erreur relative mesurée sur l'ensemble des applications de test des modèles construits à partir des évènements sélectionnés par PESel et les trois méthodes existantes. Au regard de cette figure, nous pouvons constater que la sélection basée sur les méthodes de statistique classique à base des corrélations de Pearson et Spearman n'aboutit pas à une solution suffisante pour la modélisation de la puissance. Ce résultat rejoint les conclusions formulées par les auteurs de [GE03] qui comparent différentes méthodes pour la sélection des attributs, y compris les méthodes de fouille de données. En outre, PESel produit une solution proche de celle trouvée par la méthode basée sur l'analyse ACP, mais deux fois plus rapide. Par exemple, à l'itération $k = 11$, PESel produit une solution en 129.8 s par rapport à 226.22 s le temps pris par la méthode à base de ACP. Nous pouvons aussi conclure que le modèle NN est plus précis que LM pour $k < 6$. Pour $k > 6$, les deux modèles produisent des estimations avec une erreur comparable, environ 5% pour la résolution temporelle de 100 ms, mesurée sur l'ensemble des applications de test. Le coefficient de détermination (R^2) est aussi d'environ 0.8801.

4.4.3 Impact de la période d'échantillonnage sur la précision des modèles

Dans la section précédente, nous avons comparé la précision des deux modèles LM et NN construits à partir des évènements sélectionnés par PESel pour une résolution temporelle de 100 ms. Dans cette partie, nous souhaitons étudier l'impact de la période d'échantillonnage des données sur la précision des estimations. À cet effet, nous avons entrainé les deux mo-

dèles en utilisant les 11 évènements sélectionnés précédemment sur des données extraites à partir de l'ensemble des applications d'entrainement et pour des périodes d'échantillonnages différentes. La moyenne de l'erreur relative mesurée sur l'ensemble des applications de test est représentée sur la Figure 4.8. L'erreur du modèle LM varie entre 6% à 1 ms et 4.3% à 1 s. L'erreur de NN est inférieure, variant entre 5.5% à 1 ms et 3.1% à 1 s. Le coefficient de détermination R^2 est compris entre : 0.8102 à 1 ms et 0.8912 à 1 s pour LM, et 0.8221 à 1 ms et 0.9084 à 1 s pour NN.

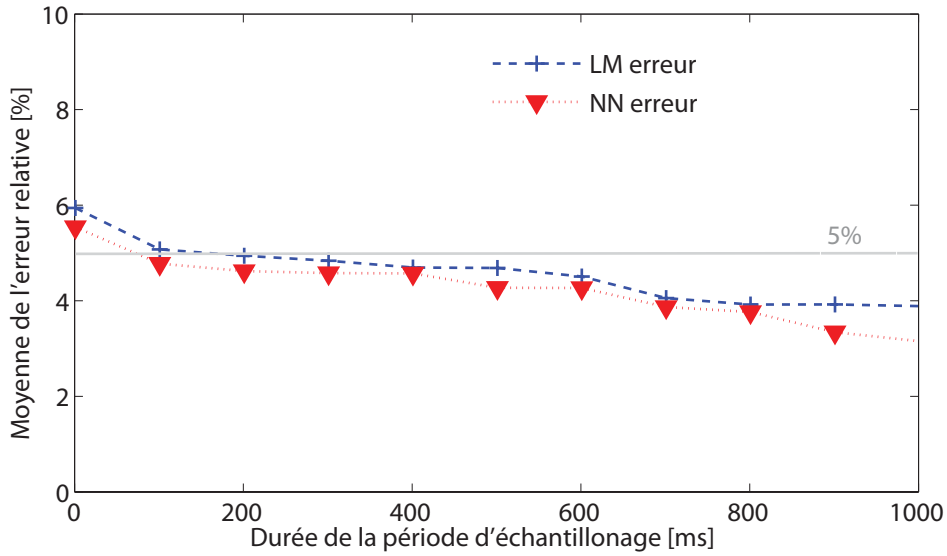


FIGURE 4.8 – Moyenne de l'erreur relative de LM et NN à partir des 11 évènements sélectionnés pour différentes périodes d'échantillonnage

4.4.4 Suivi de la Puissance et mise en œuvre du DVFS

Nous souhaitons étudier le suivi de la puissance en présence d'une technique de gestion dynamique de la fréquence et de la tension d'alimentation (DVFS). Pour cette raison, nous avons enregistré les données issues du PMU pour les 11 évènements sélectionnés en gérant manuellement la fréquence des cœurs exécutant les deux ensembles d'applications (entrainement et test). Nous rappelons que l'information liée à la fréquence était introduite dans les deux modèles LM et NN présentés précédemment dans la section 4.3.2. Nous entrainons ces deux modèles sur la base extraite pour suivre les variations de la puissance quelle que soit la fréquence de fonctionnement.

Le profil de la puissance est illustré sur la Figure 4.9 pour une résolution temporelle égale à 100 ms. Le suivi de la puissance par LM en bleu et NN en rouge pour deux cas différents est aussi représenté sur cette figure. Nous constatons que les deux modèles sont capables de suivre les variations de la puissance quelle que soit la fréquence de fonctionnement des cœurs. L'erreur

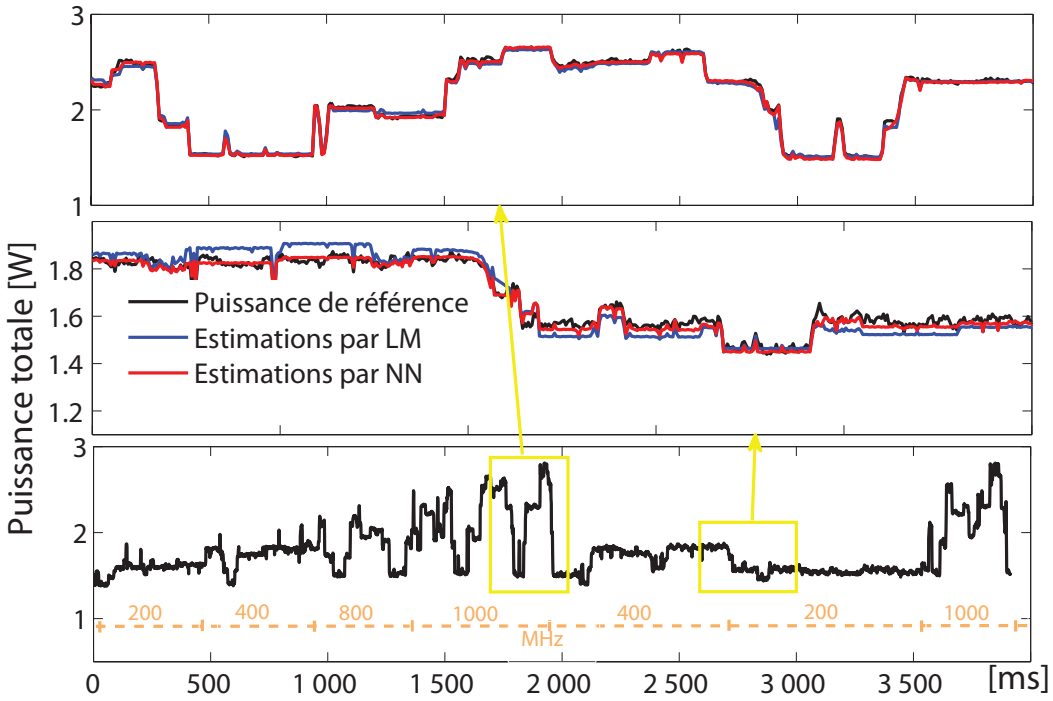


FIGURE 4.9 – Le suivi des variations de la puissances à 100 ms par les modèles LM et NN.

ainsi mesurée pour LM et NN est égale à 1.99% et 1.2% respectivement avec un coefficient de détermination R^2 égal à 0.9772 et 0.9902 respectivement.

4.4.5 Évaluation de la robustesse du modèle contre les variations de la température

Nous proposons dans ce qui suit d'étudier la robustesse du monitoring contre les variations de la température externe. Dans cette section, nous allons expérimenter cette robustesse sur le modèle NN qui produit des estimations plus précises que LM d'après l'analyse conduite dans la section 4.4.2. Afin de varier la température externe, nous avons placé la carte dans une enceinte thermique, en variant la température de -20 °C à +80 °C. La puissance de référence consommée par la carte pour différentes températures est représentée sur la Figure 4.10-a. Puisque le temps requis pour varier la température dans l'enceinte thermique est assez long (> 30 minutes), les phases transitoires ne sont pas représentées sur cette figure.

La Figure 4.10-b montre que pour le même ensemble d'applications exécutées à deux températures différentes (ici à +80°C et +10°C), une différence significative dans la consommation peut être constatée. Ceci reflète le besoin d'un modèle de puissance robuste à la variation de la température. Pour ce faire, nous avons entrainé le modèle NN sur l'ensemble des applications en prenant en compte l'information liée à la variation de la température. Ce modèle désigné par NN_R est comparé à un deuxième, construit à +20°C (NN_{20}). NN_{20} ne prend en compte que la

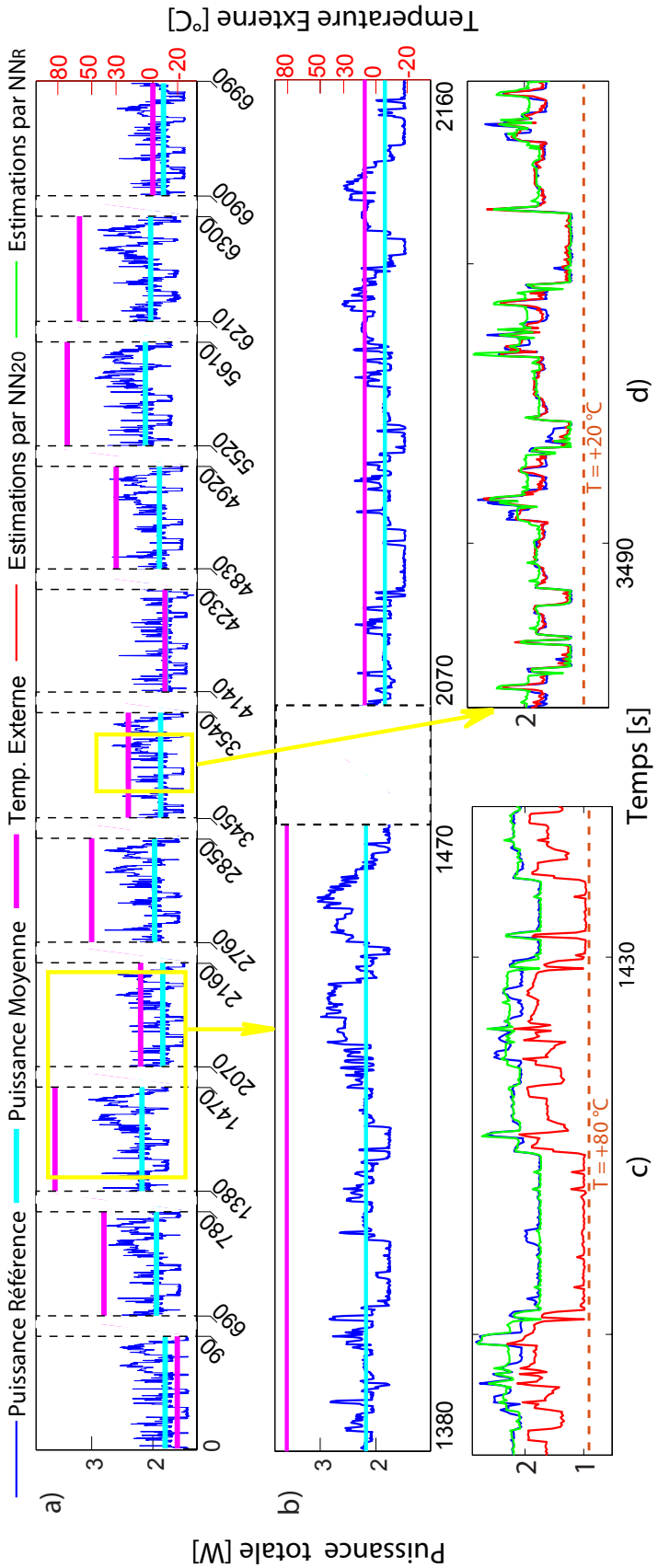


FIGURE 4.10 – La puissance consommée par Snowball à différentes températures externes (a) et (b) ; Le suivi de la puissance par les deux modèles NN_{20} et NN_R (c) and (d).

variation de l'activité représentée par les 11 évènements de performance pour estimer la puissance. Les tracés (c) et (d) de la Figure 4.10 montrent le suivi de la puissance par les deux modèles à +80°C et +20 °C respectivement. Nous pouvons constater qu'à +20°C les deux modèles sont capables de suivre les variations de la puissance, alors qu'à +80°C seul le modèle NN_R est capable de suivre ces variations. Ce modèle suppose l'utilisation d'un capteur de température externe.

4.4.6 Évaluation du Coût

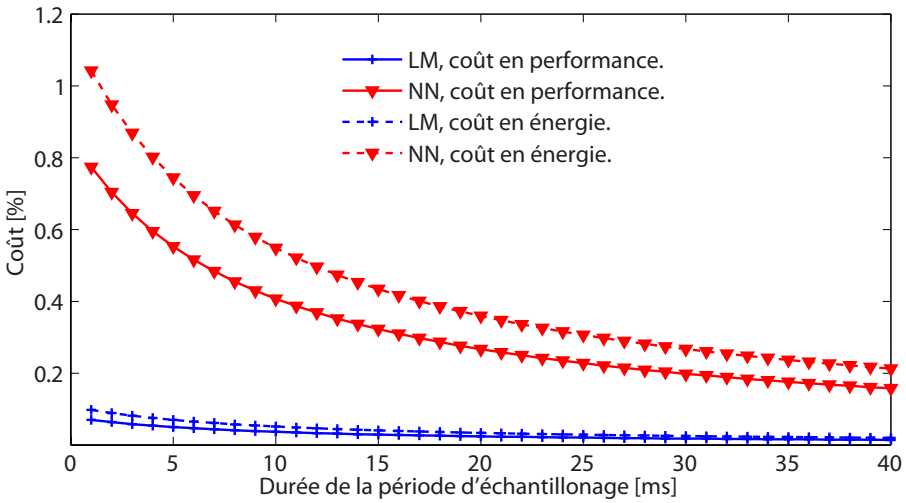


FIGURE 4.11 – Le coût du monitoring par LM et NN au niveau système.

Le monitoring de l'activité ne nécessite pas l'insertion de matériel dédié puisqu'elle repose sur l'utilisation du PMU déjà disponible dans certains MPSoCs : l'estimation en ligne de la puissance entraine un coût supplémentaire seulement en termes d'énergie et de performance. Afin d'évaluer ce coût, les deux modèles LM et NN sont exécutés par le processeur à 1 GHz, où la pénalité maximale est atteinte par l'observation des 11 évènements. Le temps requis pour réaliser le calcul est mesuré à 0.7µs et 7.7µs par LM et NN respectivement. La puissance consommée pendant cette opération est égale à 2.02W. L'énergie est ainsi égale au produit de la puissance et du temps de calcul pour les deux modèles LM et NN. Ensuite, le coût en énergie est calculé comme le rapport entre cette énergie sur l'énergie requise par le processeur en mode idle (lorsque le processeur n'exécute pas d'applications). D'autre part, le coût en performance est calculé comme étant le rapport entre le temps pris pour réaliser l'estimation et la période d'échantillonnage des données. La Figure 4.11 trace le coût en performance et en énergie des deux modèles. Nous pouvons constater que le coût est relativement faible et devient négligeable pour une résolution temporelle supérieure à 15 ms (coût < 0.5%).

4.4.7 Interprétation des résultats

Il existe dans la littérature de nombreux travaux reposant sur la modélisation hybride de la puissance en utilisant les évènements de performance au niveau matériel et logiciel. Le tableau 4.5 compare notre méthode aux travaux existants. Le monitoring de la puissance n'est pas abordé comme une approche en ligne, ce qui justifie le manque de l'évaluation du coût du monitoring dans ces publications. La résolution temporelle la plus élevée est atteinte dans [BWeWK03], où la puissance est estimée seulement en fonction de l'activité. D'autres méthodes sont moins précises à une résolution moins élevée. Seule la solution proposée dans [SKUY15] estime la puissance, en tenant compte de la fréquence et de la température. Cependant, l'activité est considérée comme une constante caractérisée par application. Notre méthode reste plus robuste et plus précise à une granularité plus fine. Elle permet de suivre les variations de puissance à une résolution maximale égale à 1 ms avec une précision inférieure à 6% en prenant en compte la fréquence de fonctionnement du système, et elle est robuste aux variations de température.

TABLEAU 4.5 – Comparaison avec les méthodes existantes.

Publication	Paramètres	Robuste	Erreur	Res. temporelle	Plateforme
[PMV13]	Activité	Non	8-10%	500ms	ARM Big.LITTLE
[BWeWK03]	Activité	Non	7%	1ms	Intel Pentium4
[BJ12b]	Activité	Non	<9%	1s	Intel AMD Dual-core
[SKUY15]	Freq, T	Non	<4%	1s	ARM Big.LITTLE
Notre méthode	Activité, Freq	Yes	6 → 3%	1ms → 1s	Snowball PDK

4.5 Conclusion

L'exploitation des ressources existantes pour un monitoring à grain fin de la puissance à faible coût a été abordée dans ce chapitre. La méthode repose sur l'optimisation de l'utilisation des compteurs de performance qui sont chargés d'observer des évènements au niveau matériel et logiciel, pour mettre en œuvre une méthode de monitoring hybride efficace.

En effet, il existe un petit nombre de compteurs qui peuvent être configurés par un grand nombre d'évènements distincts. L'enjeu est de retrouver le sous-ensemble d'évènements qui permet une estimation précise de la puissance. Une heuristique inspirée des algorithmes de fouille de données a été développée dans ce but, et elle permet de sélectionner automatiquement les évènements ayant le plus d'impact.

D'autre part, le monitoring de la puissance à une granularité fine ne dépend pas seulement de l'activité, mais aussi d'un ensemble de paramètres, y compris de la fréquence de fonction-

nement du processeur et de la température externe de la puce. Dans ce chapitre, nous avons mis en place un modèle de la puissance qui permet de suivre les variations à différents niveaux de la hiérarchie du système. Il prend en compte les mécanismes d'adaptation (DVFS), et est robuste aux variations de température. La mise en œuvre de cette méthode sur un système MPSoC montre un suivi dynamique de la consommation d'une précision autour de 98% avec une résolution temporelle de 100 ms.

L'impact de la température sur la variation de la puissance est essentiel pour mettre en place des mécanismes d'adaptations efficaces. Toutefois, la température dissipée de la puce affecte aussi la fiabilité de la puce, par exemple, par la présence des points chauds. C'est donc naturellement que, dans le prochain chapitre, nous nous pencherons sur l'exploration des nouvelles méthodes d'analyses de données pour adresser le problème du monitoring de la température.

CHAPITRE 5

MONITORING DE LA TEMPÉRATURE DISSIPÉE DES PUCES

« Ceux qui sont assez fous pour penser qu'ils peuvent changer le monde sont ceux qui le font »

Steve Jobs

Sommaire

5.1 Introduction

De manière générale, la température varie avec la puissance. Cette relation est exprimée selon la loi d'Ohm thermique dans l'équation (5.1) [AAE+13a], θ (en °C/W) est un paramètre qui dépend des caractéristiques thermiques du matériel. Au-delà de la température ambiante T_a (en °C), une augmentation de la température dans une région est déclenchée par la présence d'une activité dans le système. Cette activité fait croître la partie dynamique de la consommation en puissance ($P_{dynamique}$ en Watt), qui à son tour augmente la température selon l'équation (5.1). Cette dernière aura pour impact une hausse de la partie statique de la puissance qui par conséquent, augmente aussi la température. En régime permanent (lorsque l'activité devient stable), ce phénomène converge vers une température moyenne constante. Dans certains cas, celle-ci peut dépasser un seuil critique en certains points du circuit.

$$T = T_a + \theta \times (P_{statique} + P_{dynamique}) \tag{5.1}$$

Les effets de la température ne se limitent pas à l'augmentation de la consommation (statique), mais aussi dégrade la fiabilité des circuits par la présence des pics de température (*hotspot* ou points chauds), et la performance des transistors, notamment en affectant la tension de seuil. Un point chaud se caractérise par une température relativement élevée en un point du circuit, et ce pour une durée variable. Il peut être dû à la présence d'une activité logique concentrée en endroit fortement localisé. La position de ce hotspot peut varier en fonction de l'application et des données. De plus, des nouveaux pics de température peuvent se produit à cause du vieillissement [MM06]. Afin de mettre en œuvre des mécanismes d'adaptation efficaces pour limiter tous ces effets, une méthode de monitoring à grain fin est indispensable. À titre d'exemple, la méthode de distribution de la charge de travail par migration et placement de tâches présentée dans [SZS10], la gestion pro-active des ressources basées sur la prédiction des points chauds introduite dans [SKUY15], et les techniques de DVFS à grain fin au niveau des composants du circuit dans [ZXD+08], requièrent tous une observation de la température au niveau de chaque cœur.

Un exemple de cartographie thermique est représenté sur la Figure 5.1 à un instant donné obtenu à partir d'une image par caméra thermique de la puce Zynq [Inc] qui contient une partie système avec un double cœur ARM, ainsi qu'une partie programmable. Il est clair que la température dissipée est non-uniforme à un instant donné et dépend de la densité de puissance consommée en chaque point du circuit. Ainsi, l'intégration d'un capteur permet de mesurer la température dans une zone bien particulière du circuit. La haute résolution temporelle de certains capteurs permet la détection des variations fines de température. Cependant, une bonne observabilité à une résolution spatiale fine nécessite l'intégration de plusieurs capteurs sur la même puce. Même s'il y a une tendance nette à l'augmentation du nombre de capteurs dans

les processeurs commerciaux, il faut se poser la question d'une approche générique et optimale de l'intégration de ces capteurs ; le challenge étant d'estimer la température du circuit en tout point avec un minimum de capteurs.

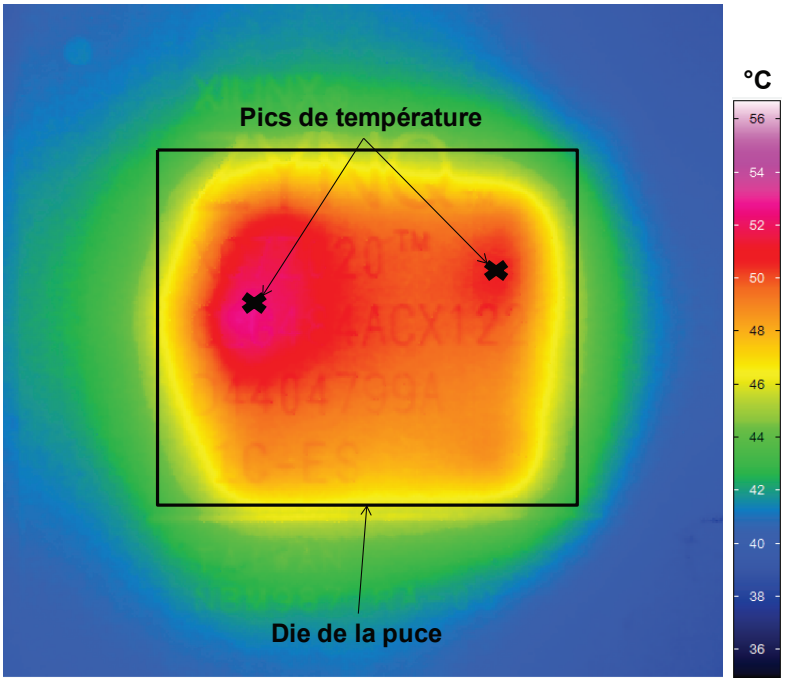


FIGURE 5.1 – Exemple de dissipation thermique à un instant donné de la puce FPGA zynq.

Dans ce chapitre, nous proposons une méthode d'analyse de données issues du flot de conception pour identifier l'information pertinente pour un monitoring efficace de la température. Pour cela, nous avons mis en place un ensemble d'outils nous permettant de réaliser des simulations thermiques du circuit. Grâce à l'instrumentation des logiciels utilisés, nous pouvons mettre en œuvre des méthodes de segmentation des données afin de définir les régions thermiquement homogènes. À partir de cette information, nous proposons une solution de placement des capteurs de température pour mettre en œuvre une solution de monitoring précise et peu coûteuse à une résolution spatio-temporelle fine.

Dans un premier temps, nous nous penchons sur la description de l'outil qui permet l'extraction de données à partir du flot de conception. Ensuite, nous présentons l'ensemble des méthodes de segmentation pour le regroupement des nœuds thermiquement homogènes, ainsi que la méthode de placement de capteurs de température. Enfin, nous validons notre méthode sur un cas d'utilisation.

5.2 Simulation thermique des circuits FPGA

En premier lieu, il convient d'analyser le comportement thermique du circuit. Afin de retrouver l'information utile pour un monitoring efficace de la température, nous avons mis en place un flot qui permet d'extraire les données à partir des simulations des circuits sur FPGA. Le prototypage sur FPGA nous offre de nombreux avantages, y compris la flexibilité du reconfigurable, et la validation de la méthode de monitoring par des mesures réelles après implémentation sur carte.

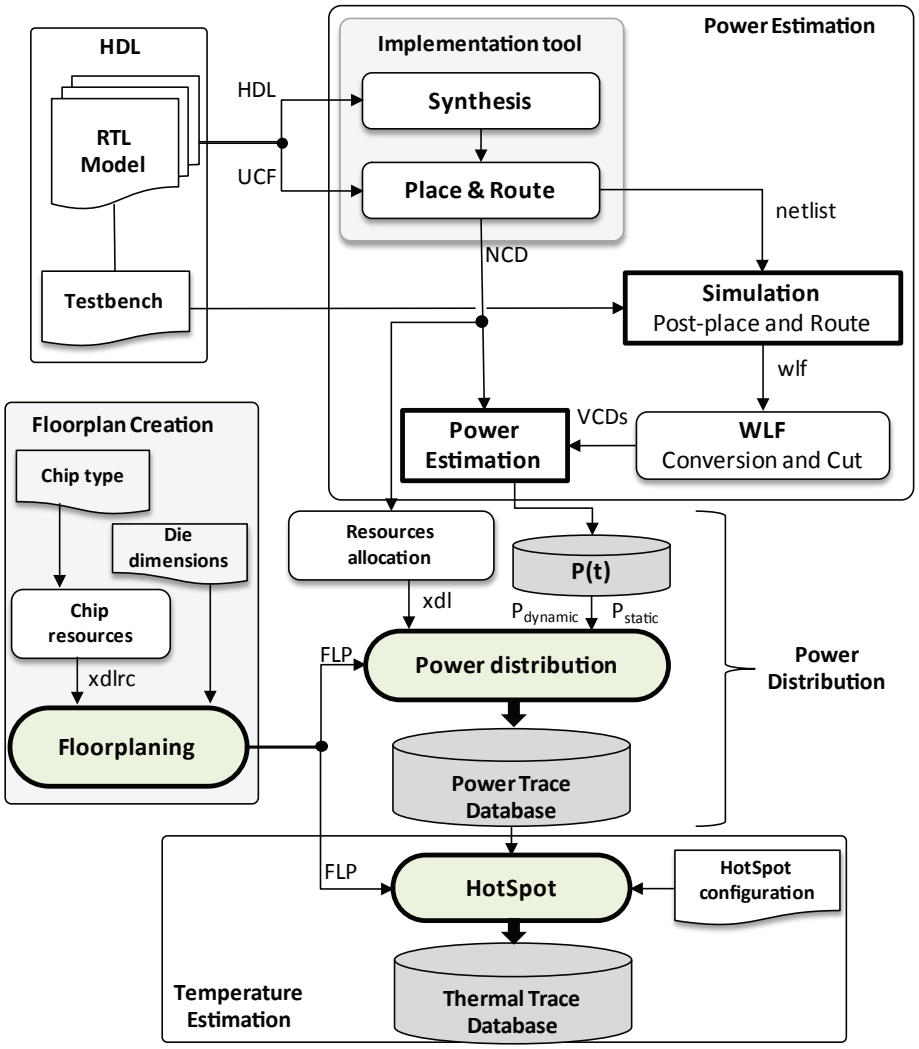


FIGURE 5.2 – PoETE : Le flot CAO complet pour la génération des bases de données thermiques.

Le flot complet *PoETE* (*Power Estimation and Temperature Estimation*) illustré dans la Figure 5.2 permet de simuler le comportement thermique du circuit à partir des outils de conception FPGA. Les différentes étapes nécessaires, ainsi que les outils déployés dans cette optique, sont présentés dans la suite de cette partie.

5.2.1 Estimation de la puissance

Les cartographies thermiques sont des estimations/mesures de la température sur toute la surface du cœur et à des instants différents. La première étape de *PoETE* consiste à estimer la puissance instantanée consommée, qui va permettre ensuite d'estimer la température à chaque instant. Le processus est identique à ce qui a été présenté dans la Section 3.5. Il découpe la simulation de la *netlist* placée et routée en P sous-simulations de période t_i. Pour chacune, l'outil d'estimation de la puissance analyse l'activité (VCD) et produit un rapport de puissance. Ce rapport contient en plus de l'estimation de la partie statique de la consommation, des estimations de la puissance dynamique consommée par tous les composants du système.

5.2.2 Création du Floorplan

La deuxième phase consiste à diviser la surface du cœur en nœuds identiques afin de procéder à l'estimation de la température pour chacun d'entre eux. Ce découpage utilise les ressources disponibles du FPGA (fichier *xdlrc*) et les dimensions du *die* pour fractionner le rectangle en matrice de C × L blocs correspondant aux nœuds. Chacun se rapporte donc à une matrice de $n \times m$ ressources, qui peut contenir de la mémoire (*BRAM*), de la logique ou bien des LUTs (*Look-Up Tables*). Les dimensions du *die* obtenues à partir des spécifications ou bien des mesures par radiographies, sont utilisées pour calculer les dimensions de chaque bloc dans le *Floorplan*.

5.2.3 Distribution de la puissance

Une fois que les rapports de puissance sont établis et que le *Floorplan* est créé, la troisième étape consiste à répartir la puissance consommée sur les différents nœuds du *Floorplan*. Initialement, la puissance statique est distribuée de manière uniforme à tous les nœuds de la puce pour les P échantillons. D'après la relation en boucle entre la puissance statique et la température, la consommation statique devra être redistribuée en tenant compte de la dissipation thermique non uniforme sur toute la puce. Ainsi, cette boucle est prise en compte dans la modélisation de la température.

La distribution de la puissance dynamique consommée par les différents composants du circuit, nécessite l'identification des différentes ressources allouées par ces composants. Pour cela, le fichier *xdl* est analysé pour identifier les nœuds qui contiennent des ressources allouées à des composants. Ensuite, il faut analyser les rapports de puissance pour extraire la puissance consommée par les différents composants, ce qui permet par la suite de diviser cette puissance et de la rajouter à la partie statique aux nœuds alloués.

5.2.4 Estimation de la température spatio-temporelle

La dernière étape consiste à produire des estimations de la température en tout point du circuit et à chaque instant. Pour cela, nous utilisons l'outil *Hotspot* [HGV+06] qui crée pour chaque nœud un modèle thermique RC équivalent (voir Figure 5.3) en prenant en compte tous les paramètres de la puce, notamment les caractéristiques thermiques du semiconducteur, du *package* et du *die*. *Hotspot* produit donc une estimation de la température de chaque nœud en fonction de la distribution de puissance et du *Floorplan* réalisés ci-dessus. Il produit en premier lieu un état stationnaire en calculant la température de chaque nœud à chaque instant en fonction de la densité de puissance consommée. Ensuite, ces estimations sont ajustées en tenant compte de l'effet de la température sur la puissance statique et l'impact de la température des nœuds voisins. Ce flot génère les cartographies thermiques qui permettent de créer des bases de données qui représentent la température en 2 dimensions une spatiale et une temporelle.

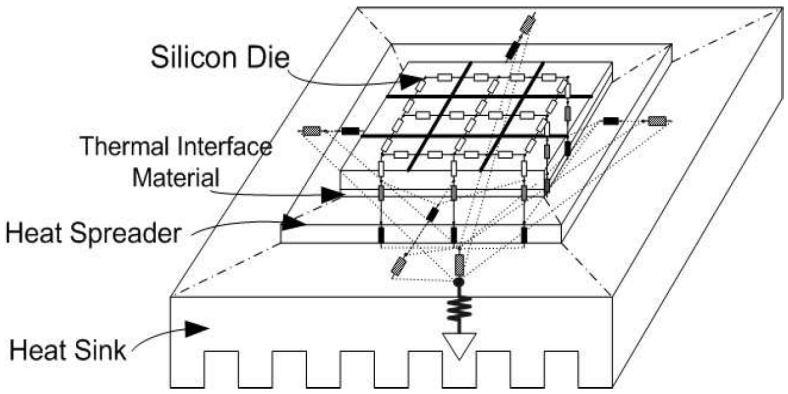


FIGURE 5.3 – Modélisation de la température dans *Hotspot* en découpant le cœur de la puce en matrice de nœuds.

5.3 Méthode d'analyse de données pour le monitoring de la température

En général, le monitoring est réalisé par l'insertion d'un ensemble de capteurs qui mesurent la température dans des régions précises du circuit. Ainsi, une solution efficace est celle qui produit une information précise à une résolution spatiale et temporelle fine, et à faible coût (un nombre minimal de capteurs). La résolution temporelle est une caractéristique du capteur, alors que la spatiale dépend du nombre de capteurs intégrés et de leurs positions. Les principales approches existantes pour le placement des capteurs sur puce peuvent être regroupées en 2 catégories (voir Chapitre 2 Section 2.5.4) : (*i*) uniforme [NCR10, RCN11], qui propose une structure uniforme de capteurs basé sur une décomposition géométrique de la surface du cœur

et (*ii*) non-uniforme [SR10, MMNL08, RCN11], où l'insertion de capteurs est basée sur des informations extraites du comportement du circuit. En général, l'approche uniforme nécessite l'insertion d'un grand nombre de capteurs pour obtenir une information précise à une résolution spatiale fine, alors que l'approche non-uniforme est moins coûteuse et nécessite moins de capteurs.

La solution recherchée consiste à minimiser le coût en réduisant le nombre de capteurs et maximiser la précision du monitoring. Nous proposons une méthode selon l'approche non-uniforme qui nécessite une phase d'apprentissage au moment de la conception du circuit pour définir l'emplacement optimal des capteurs. Cet apprentissage est conduit sur les cartographies thermiques extraites à partir du flot de conception à l'aide de l'outil *PoETE* présenté précédemment.

5.3.1 Bases de données

Une cartographie thermique est l'ensemble de températures des différents nœuds à un instant donné. L'ensemble des nœuds représentent donc la surface totale du cœur de la puce. La Figure 5.4 illustre la base de données : chaque colonne correspond à un nœud de la puce, et les lignes correspondent aux cartographies thermiques de la puce à des instants successifs.

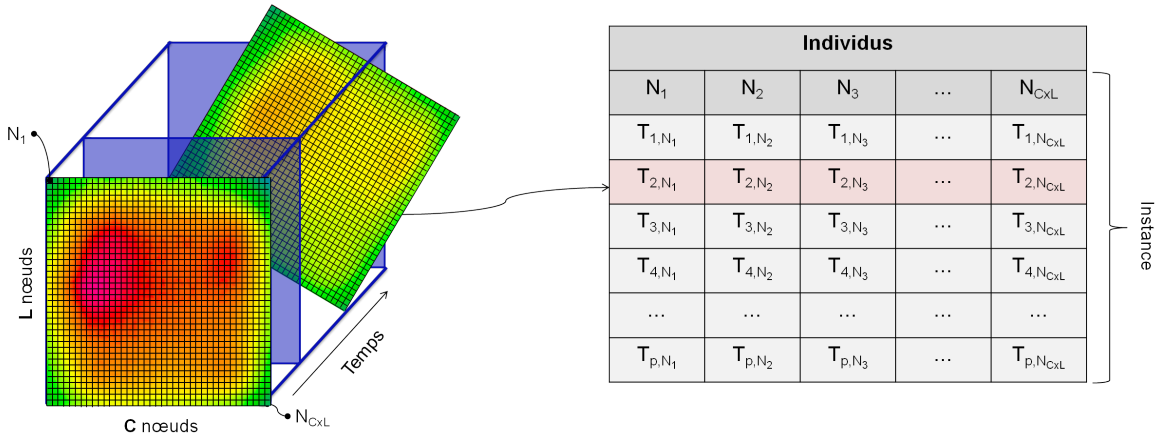


FIGURE 5.4 – Organisation de la base de données.

5.3.2 Analyse de données par segmentation ou *Clustering*

L'ensemble des nœuds de la base de données ont évidemment des comportements thermiques variables. Mais l'analyse de ces données par des techniques de segmentation peut mettre en lumière des singularités. Pour cela dans un premier temps, nous proposons des techniques d'identification de groupes de nœuds qui ont un comportement thermique similaire. La segmentation ou *clustering*, est une technique non-supervisée qui consiste à apprendre sans su-

perviseur (c-à-dire sans variable à prédire). C'est un thème de recherche majeur en apprentissage automatique, en analyse et en fouille de données ainsi qu'en reconnaissance de formes. L'objectif est, à partir de données constituées d'un ensemble d'individus (dans notre cas des nœuds thermiques) avec leurs caractéristiques, de construire des groupes homogènes dans le sens où :

- deux individus ayant des caractéristiques proches doivent appartenir à un même groupe ;
- deux individus ayant des caractéristiques éloignées doivent appartenir à des groupes différents.

Un exemple est représenté dans la Figure 5.5 qui illustre les différentes régions (ou groupe de nœuds) thermiquement homogènes (régions 1 à 5) sur une cartographie thermique produite à un instant donné. Dans cette étude, nous explorons l'utilisation des deux algorithmes de segmentation les plus connus : la méthode k-moyenne et la segmentation hiérarchique. Dans ce qui suit, nous présentons le critère d'évaluation des comportements thermiques des nœuds. Ensuite, nous nous penchons sur la description de ces deux algorithmes.

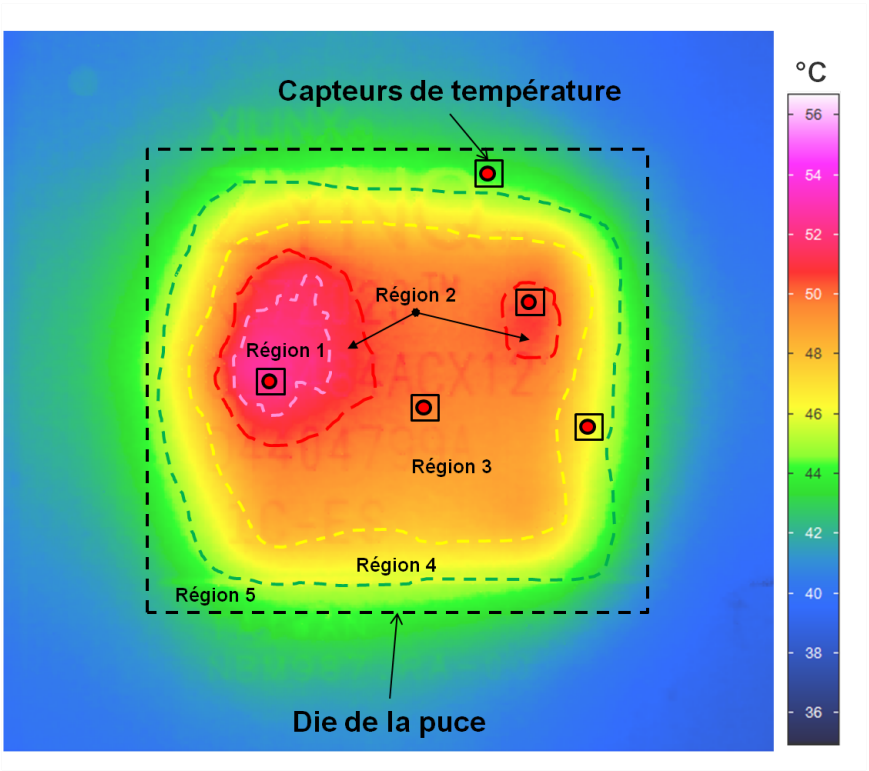


FIGURE 5.5 – Surveillance de la température.

Mesure de distance

L'objectif de la segmentation étant de déterminer des groupes homogènes et distincts, il est nécessaire de se baser sur des critères de similarité pour mesurer la ressemblance entre deux

nœuds. La distance euclidienne est une des plus utilisées dans les algorithmes de *clustering* et mesure la dissemblance entre deux nœuds. Par conséquent, lorsque la distance entre les deux nœuds N_a et N_b ayant des températures extraites pour p instants (voir équation 5.2) est proche de zéro alors les caractéristiques thermiques de N_a et N_b se rassemblent ($T_{j,N_a} = T_{j,N_b}$ avec $j = \{1..p\}$). Autrement dit, deux nœuds ont des comportements thermiques similaires s'ils ont des valeurs de températures proches à tout instant.

$$d_{N_a,N_b} = \|N_a - N_b\| = \sqrt{(T_{1,N_a} - T_{1,N_b})^2 + ... + (T_{p,N_a} - T_{p,N_b})^2} \tag{5.2}$$

k-moyenne *clustering*

L'algorithme de segmentation k-moyenne suppose qu'il existe K sous-ensembles distincts. D'abord, il désigne k centres $s_1..s_k$ parmi les individus associés à k groupes $S_1..S_k$. Ces centres sont généralement désignés aléatoirement avant que l'algorithme réalise les itérations suivantes :

- Pour chaque individu qui n'est pas un centre du groupe, il calcule la distance entre cet individu et tous les centres des groupes $S_1..S_k$. Ensuite, l'individu est ajouté au groupe pour lequel la distance minimale calculée.
- Dans chaque nouveau groupe ($S_1..S_k$), on définit le nouveau centre ($s_1..s_k$) comme étant le barycentre des individus du groupe.

L'algorithme s'arrête suivant un critère d'arrêt fixé par l'utilisateur qui peut être choisi parmi les suivants : un nombre d'itérations maximal atteint, ou la convergence de l'algorithme, c'est-à-dire que l'inertie intra-groupe ne s'améliore quasiment plus entre deux itérations.

clustering ascendant hiérarchique

La segmentation ascendante hiérarchique a pour objectif de construire une suite de partitions emboitées de données en n groupes (avec $n = C \times L noeuds$), $n-1$ groupes, ..., 1 groupe. Les itérations de l'algorithme peuvent être décrites de la manière suivante :

- À l'étape initiale, les n individus constituent des groupes à eux seuls ; il va initialement associer un groupe à chaque nœud.
- On calcule les distances deux à deux entre les individus, et les deux individus les plus proches sont réunis en un groupe.
- La distance entre ce nouveau groupe et les $n-2$ individus restants est ensuite calculée, et à nouveau les deux éléments (groupes ou individus) les plus proches sont réunis.

Ce processus est réitéré jusqu'à ce qu'il ne reste plus qu'une unique classe constituée de tous les individus. Les regroupements successifs sont ensuite représentés à l'aide d'une arborescence, aussi appelé dendogramme (voir l'illustration sur la Figure 5.6). En bas de l'arbre se

trouvent les individus. La fusion de deux éléments est représentée par une branche reliant ces deux éléments, dont la hauteur correspond à un critère de sélection proportionnel à la distance entre les deux éléments fusionnés. Le critère le plus connu pour la segmentation hiérarchique est la distance de Ward [Mur83] qui mesure la perte de variance intergroupes. Dans le cas de la Figure 5.6 correspondant à un exemple d'illustration, la hauteur des branches est proportionnelle au pourcentage de perte de variance.

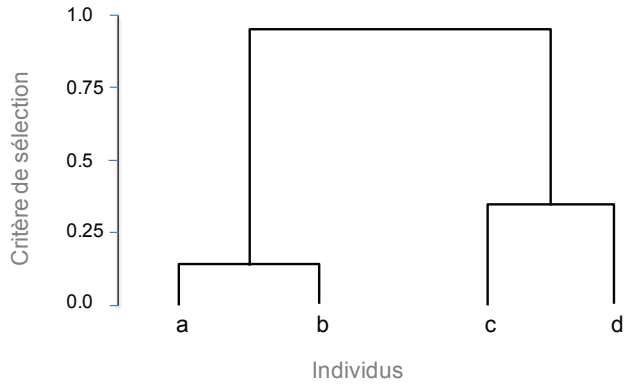


FIGURE 5.6 – Exemple de dendogramme représenté pour 4 individus {a,b,c et d}.

5.3.3 Placement de capteurs de température

Les groupes $C_1..C_k$ obtenus à l'issue de la segmentation de la base de données correspondent à des régions thermiquement homogènes. Plus le nombre de régions est élevé, plus la granularité spatiale est fine. Dans l'idéal, l'objectif est de pouvoir estimer la température du circuit avec une résolution spatiale et une précision maximale (en supposant que la résolution temporelle est atteinte par le choix du capteur). Mais compte-tenu du coût des capteurs en surface silicium, il est nécessaire de fixer un nombre limite, ce qui revient à jouer sur le compromis entre le nombre de capteurs et la précision.

Une fois que le nombre de régions k est fixé, il est nécessaire de déterminer la position des capteurs dans les régions. L'état de l'art montre différentes approches dans ce but. Une solution proposée dans [LMMM08, MMNL08] consiste à placer un capteur au milieu de chaque région. Une autre approche proposée dans [LRLZ13] est basée sur une double segmentation ; la première pour regrouper les nœuds thermiquement homogènes et la deuxième pour fusionner les groupes selon le diagramme de *Voronoi* qui analyse la distance entre les barycentres des régions détectés. Toutefois, ces approches s'appliquent à un ensemble qui contient des nœuds voisins créant une surface continue. Cette hypothèse n'est pas toujours vraie ; car la simple analyse des cartographies thermiques montre des situations dans lesquelles des nœuds espacés ont des comportements thermiques proches : c'est le cas de la région 2 de la Figure 5.5. Le placement d'un capteur dans ce cas n'est pas trivial car elle requiert une analyse de chaque

région afin de déterminer le positionnement optimal.

La Figure 5.7 illustre notre méthode de placement de capteurs de température. Chaque groupe C_i avec $i = \{1..k\}$ contient un nombre de nœuds qui constituent des régions thermiquement homogènes. Nous souhaitons ainsi trouver le nœud qui correspond à la position optimale du capteur dans chaque groupe. En supposant que k est suffisamment grand, la plupart des nœuds dans chaque groupe doit avoir des comportements thermiques proches, c'est-à-dire une distance euclidienne petite. Par conséquent, la température de tous les nœuds d'un groupe i est proche de la température moyenne T_{c_i} du groupe à chaque instant. C'est la raison pour laquelle nous avons choisi de sélectionner le nœud qui permet la meilleure estimation de la température moyenne du groupe. Pour ce faire, nous allons utiliser les méthodes de sélection d'attributs selon l'approche *wrapper* (voir Chapitre 3 Section 3.4) en considérant la température moyenne comme classe à prédire selon un modèle linéaire simple. La méthode évalue tous les nœuds d'un groupe d'une manière itérative en se basant sur le critère d'erreur RMSE (*Root Mean Squared Error*). Ainsi, le nœud qui produit le modèle linéaire le plus précis (RMSE le plus petit) sera sélectionné pour le placement du capteur.

Contrairement à la politique d'emplacement proposé dans [LMMM08, MMNL08] et celle présentée dans [LRLZ13], notre méthode reste indépendante de la forme géométrique des régions. Grâce à cette solution, il devient possible de définir l'emplacement optimal du capteur et ce de manière systématique.

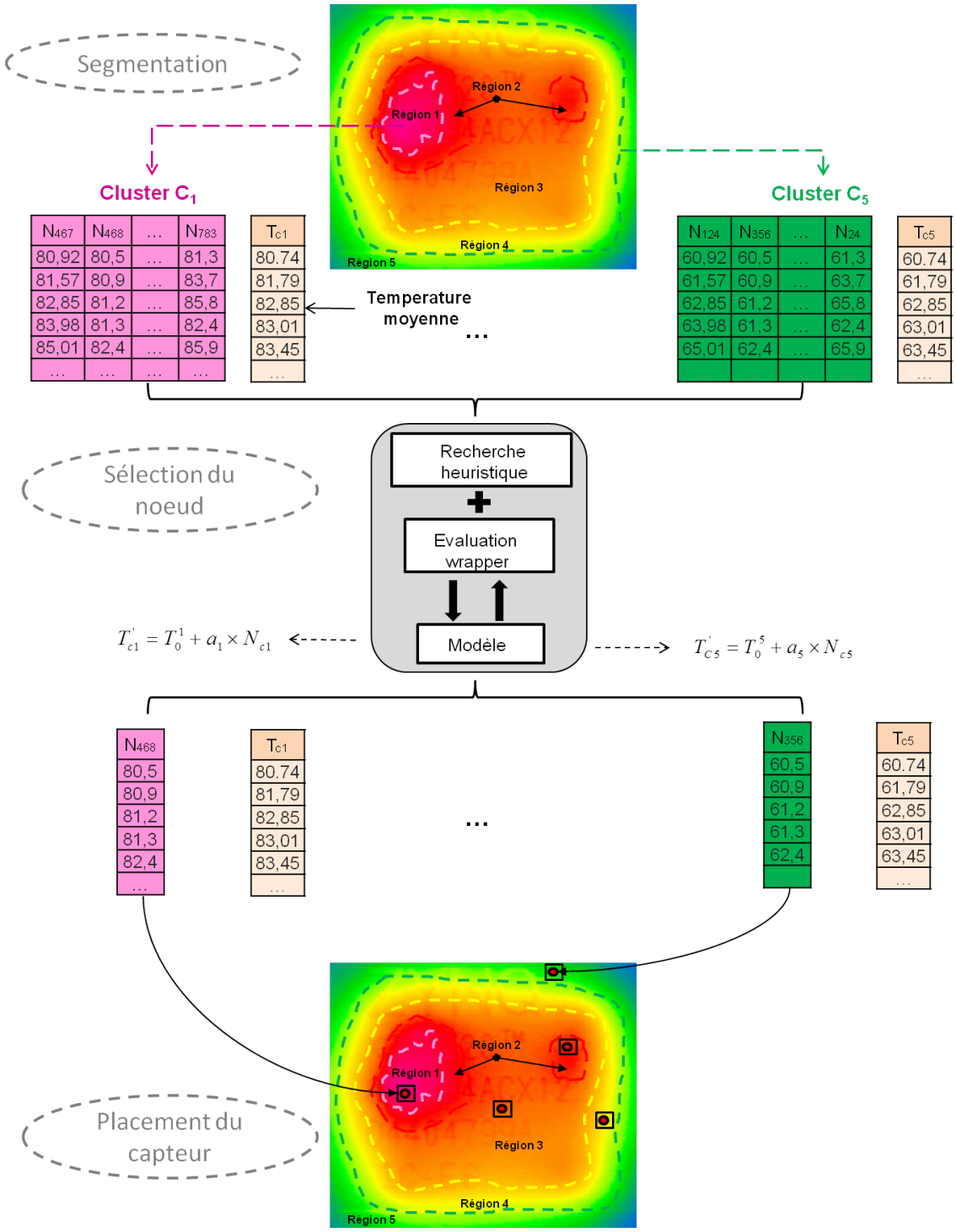


FIGURE 5.7 – Illustration de la méthode de placement de capteurs.

5.4 Évaluation

Nous avons proposé une méthode d'analyse de données qui permet d'extraire l'information utile pour un monitoring efficace de la température. Dans cette section, nous allons évaluer notre méthode sur un cas d'étude pour mettre en valeur cette solution. Nous présentons d'abord le système ainsi que l'ensemble de *benchmarks* utilisé dans ces expérimentations. Ensuite, nous nous penchons sur l'évaluation des estimations des cartographies. Puis, nous étudions la précision de notre solution pour le regroupement des nœuds thermiquement homogènes ainsi que le placement des capteurs dans les régions détectés. Enfin, nous présentons une discussion sur la mise en œuvre de cette solution, notamment par rapport au coût introduit.

5.4.1 Description du cas d'étude

La plateforme matérielle correspond au SoC SecretBlaze présenté précédemment dans le Chapitre 3 Section 3.6. C'est un système sur puce open-source à base de processeur RISC 32 bits qui utilise le jeu d'instructions du processeur MicroBlaze [Xila]. En plus du processeur, SecretBlaze comporte : un Timer, une UART, une unité d'interruption, des registres GPIO et une mémoire RAM. Le processeur est implémenté sur la carte Digilent ATLYS, basée sur un FPGA Spartan-6 LX45. Cette carte dispose d'un capteur de puissance LTC2481C [Tec14]. Ce capteur nous permet de mesurer la puissance consommée par le processeur pendant l'exécution des applications avant de lancer l'outil PoETE pour la collection des données thermiques. Il mesure la puissance sur le rail d'alimentation 1.2v qui alimente le cœur FPGA à une fréquence de 50Hz et une erreur inférieure à 1%.

Afin de couvrir plusieurs cas d'utilisation, nous avons choisi des *benchmarks* exécutés par le processeur à 25MHz ; Whetstone [CW76] (initialisation de matrice, sauts conditionnels, appel système, interruptions et fonction arithmétique, *etc.*), MXM [MxM10] (multiplication matricielle) et trois applications exécutées successivement : *statemat*, *qurt* et *compress* [GBEL10].

5.4.2 Génération des cartographies thermiques

PoETE permet de réaliser des simulations thermiques du circuit implémenté sur FPGA à une résolution spatiale et temporelle fine, dans le but de mettre en œuvre une structure de monitoring précise à faible coût. Nous rappelons qu'il existe plusieurs mécanismes d'adaptation qui requièrent un monitoring de la température à une résolution fine, telle que la technique de DVFS explorée dans [ZXD+08] qui consiste à observer les variations de la température chaque 1/100 du temps d'ordonnancement des tâches (environ 80µs) pour gérer dynamiquement la fréquence afin d'éviter les hotspots.

Avant de nous pencher sur les résultats des simulations thermiques, il est intéressant d'explorer l'utilisation des caméras thermiques infrarouges pour l'extraction des cartographies thermiques des circuits implémentés sur FPGA. Le tableau 5.1 compare les caractéristiques de quelques caméras du marché qui peuvent être impliquées dans le secteur de la micro-électronique. Ce choix ne représente pas forcément toute la gamme mais donne une idée sur la possibilité de l'utilisation de ces dispositifs pour remplacer nos simulations thermiques pour les implémentations sur FPGA. La résolution spatiale des caméras varie selon le modèle et dépend non seulement des caractéristiques du capteur infrarouge mais aussi du choix de l'optique. Dans ce tableau, nous avons considéré l'optique standard pour chaque caméra. Il est clair que plusieurs sont capables de mesurer la température à une résolution spatiale suffisante pour l'analyse de données en supposant qu'une ressource CLB dans une carte FPGA fait environ une dizaine de micromètres. D'autre part, ces caméras permettent d'enregistrer les valeurs de la température pour une fréquence relativement basse (< 200Hz). Cependant, PoETE permet d'avoir des estimations des cartographies thermiques à une résolution temporelle fine qui dépend du choix de la période d'échantillonnage de la puissance et à une résolution spatiale qui peut atteindre la taille d'une cellule FPGA (par exemple : 1 CLB dans XC6SLX45 est approximé à 88 µm x 57 µm).

TABLEAU 5.1 – Comparaison des caractéristiques des caméras thermiques infrarouges.

Caméras	Taille de matrice	Résolution spatiale	Résolution temporelle	Impliqué dans
InfraTec head hr [Inf]	640x480	25/50µm	60Hz = 16ms	-
	1024x768	17µm	30Hz = 33ms	
FLIR A6xxsc [FLI]	640x480	17µm	50Hz = 20ms	[Shu10]
	640x512	3-5µm	60Hz = 16ms	
FLIR SC5xxx [FLI09]	640x512	15-30µm	100Hz = 10ms	[NCR10, RCN11, NR11]
Optris PIxxx [Opt]	160x120	200µm	120Hz = 8.3ms	-
DIAS Infrared [Sys]	384x288	50µm	50Hz = 20ms	[AAE+13b, ERHH11, AES+13]

Une alternative à la caméra infrarouge correspond à l'utilisation des outils de modélisation de la température tels que ANSYS APACHE [Apa], DOCEA ACE THERMAL [POW], *etc.* D'une manière générale, ces outils se réfèrent à des librairies à bas niveau (niveau logique ou technologique) pour caractériser la puissance à une résolution spatiale et temporelle fine avec une précision élevée. Ensuite, les modèles thermiques estiment la température en fonction de la densité de puissance tenant compte des caractéristiques thermiques et des dimensions de chaque couche en partant du *die* jusqu'au dissipateur. Cependant, les paramètres de chaque couche qui impactent la quantité de chaleur dissipée sont spécifiés par les fabricants des puces électroniques.

À partir des informations de la XC6SLX45, nous avons calibré les paramètres de l'outil open-source Hotspot dans PoETE en utilisant les mesures de la caméra thermique InfraTech head hr [Inf] afin d'obtenir des estimations précises de la température à une résolution spatio-temporelle

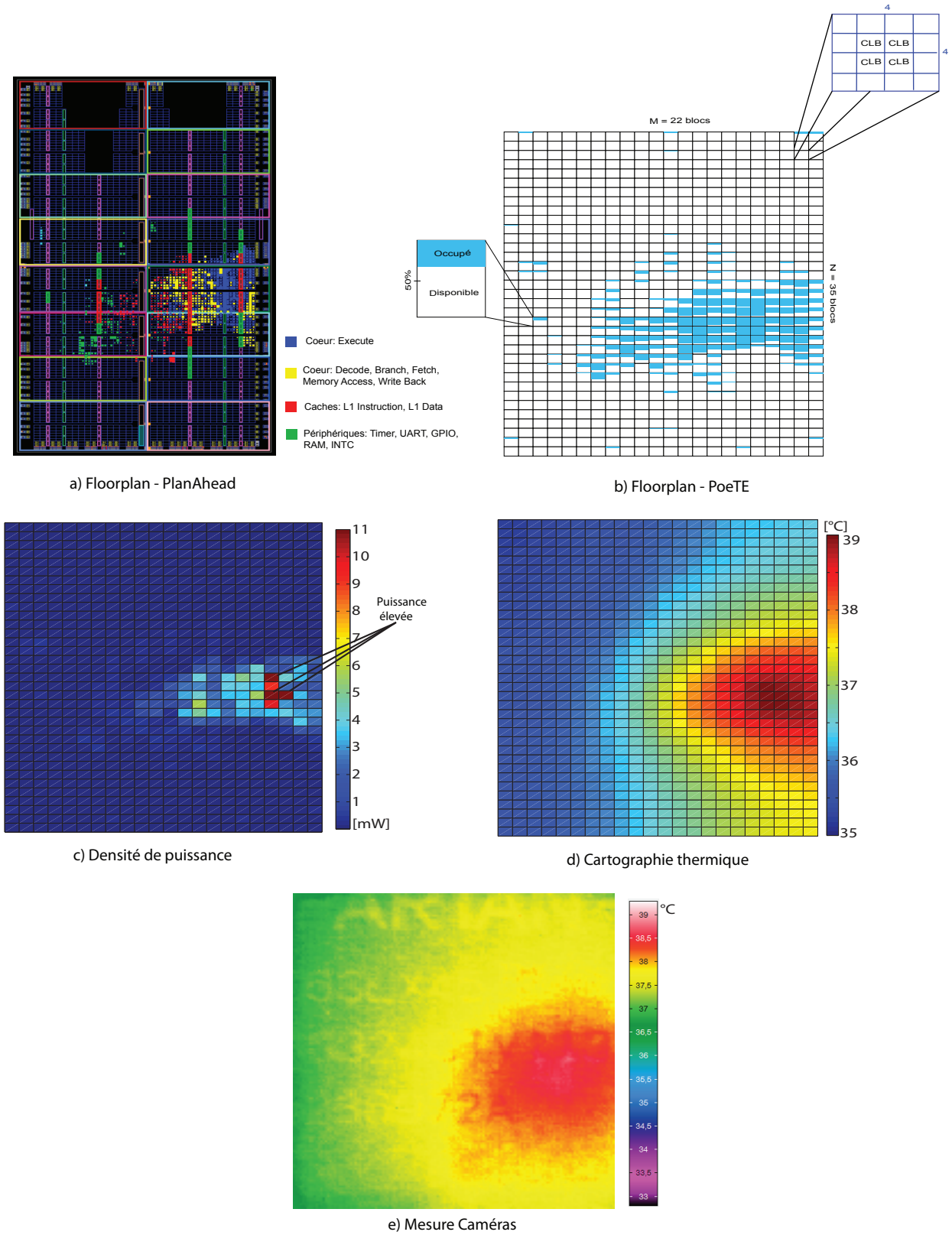


FIGURE 5.8 – Floorplan extrait de PlanAhead et PoETE a) et b) ; Densité de puissance et la cartographie thermique à un instant aléatoire c) et d) ; Image de la caméra thermique correspondante e).

fine. La XC6SLX45 est initialement représentée comme une matrice de 90x140 cellules CLB (*Configurable Logic Block*). Chaque cellule contient 2 ressources (*Slices*) qui peuvent contenir des registres, de la logique configurable et des blocs mémoires (SliceX, SliceL ou SliceM). Dans PoETE, nous avons décomposé cette représentation en matrice de bloc de dimension configurable. En effet, la taille du bloc est directement liée à la densité de puissance à partir de laquelle les estimations de la température sont produites ; c'est la puissance consommée par unité de surface. En augmentant la taille du bloc, on va agréger la puissance sur l'échelle spatiale, ce qui affecte la précision des estimations thermiques. Cependant, cette augmentation réduit aussi le nombre de nœuds thermiques à simuler, et donc le temps de simulation. Dans cette expérimentation, nous avons considéré la taille des blocs de 4x4 CLB qui correspond au compromis adéquat entre la précision des estimations et le temps de simulation.

Le floorplan de SecretBlaze créé par PoETE est comparé à celui de PlanAhead dans la Figure 5.8-a et b). Dans la Figure 5.8-b), le pourcentage d'occupation est le rapport entre le nombre de ressources allouées et le nombre total de ressources dans un bloc. Cette représentation nous a permis de distribuer la puissance sur tous les blocs qui constituent la surface du cœur FPGA (selon la méthode décrite dans la Section 5.2). La Figure 5.8-c) illustre un instant aléatoire de distribution de la puissance consommée pendant l'exécution de l'application Whetstone. Ainsi, la cartographie thermique issue de PoETE est comparée à l'image de la caméra infrarouge dans la Figure 5.8-d et e). Dans cette figure, le point chaud correspond aux nœuds associés aux blocs ayant des densités de puissance élevées (blocs rouge foncé dans la Figure 5.8-c)) où l'unité d'exécution de l'ALU (*Arithmetic and Logic Unit*) du processeur SecretBlaze était placé. L'ensemble des applications utilisées dans ce cas d'étude réalisent des opérations telles que la multiplication, l'addition, les sauts conditionnels, *etc*, mais toutes produisent de l'activité seulement au niveau du processeur. Dans ce cas, la position du point chaud dépend de l'emplacement de l'unité *EXECUTE* de l'ALU. Ainsi, la température maximale est fonction de la puissance, qui à son tour dépend de l'activité produite par les différentes applications ; les caractéristiques thermiques des différentes applications sont illustrées dans le tableau 5.2. Dans autre architecture, par exemple, le point chaud correspondait à l'ensemble de registres du processeur (*register file*) dans Alpha21364 [SSH+03] et à l'ALU dans Intel Pentium 4 [MM06]. Dans tous les cas, un point chaud est causé par une densité de puissance élevée due à la présence de l'activité dans une zone bien précise. Ce point chaud a tendance à produire d'autres pics de températures et donc accélère potentiellement son vieillissement. Ce phénomène apparaît dans la Figure 5.8-c et d) où trois nœuds provoquent l'échauffement d'une dizaine d'autres.

5.4.3 Évaluation de la précision

Notre méthode consiste à analyser les données extraites du flot de conception pour mettre en œuvre une solution de monitoring de la température précise à une résolution spatiale et

TABLEAU 5.2 – Comparaison des estimations de la température par PœTE avec la caméra thermique InfraTec Head hr.

Application	PœTE			Caméra		
	Min	Max	Moy. déviation	Min	Max	Moy. déviation
Whetstone	35.21	39.3	4.09	35.94	39.8	3.86
MXM	34.35	36.1	1.75	34.41	35.57	1.16
Statemat, qurt, compress	34.13	36.46	2.33	33.76	35.27	1.57

temporelle fine. Une fois les données extraites à l'aide de PoETE, nous utilisons les méthodes de segmentation non-supervisées pour détecter les zones thermiquement homogènes. À partir de cette analyse, un placement des capteurs de température peut être défini afin d'observer en ligne la dissipation thermique de la puce.

SecretBlaze exécute le *benchmark* Whetstone qui produit la plus grande variation de température grâce à la diversité des activités produites (voir Chapitre 3 Section 3.6). Nous avons aussi considéré le même placement et routage de SecretBlaze présenté dans la section précédente ; la puce XC6SLX45 est donc décomposée en une matrice de 35x22 blocs de 4x4 cellules. Ainsi, la base de données extraite de PoETE correspond aux cartographies thermiques de SecretBlaze à une résolution de 100µs. Donc, la taille de cette base de données est égale à 1750 instances avec $35 \times 22 = 770$ nœuds.

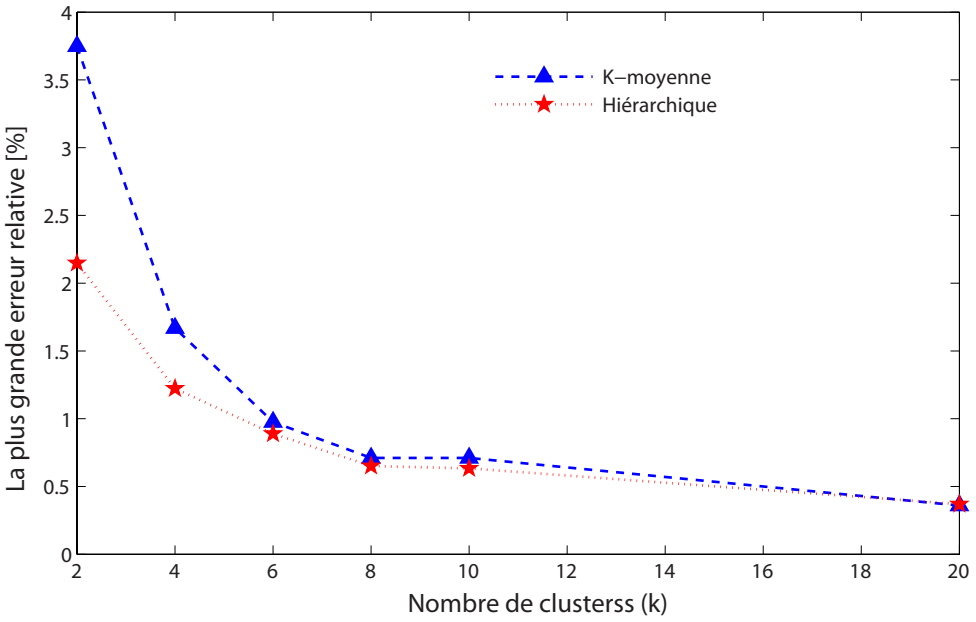


FIGURE 5.9 – Erreur relative nominale pour chaque méthode de segmentation.

L'outil open-source MeV (*MultiExperiment Viewer*) est utilisé pour l'application des algorithmes de segmentation sur la base de données extraite précédemment. C'est un outil d'analyse de données microarray polyvalentes qui intègre des algorithmes pour la segmentation de

données, la visualisation et l'analyse statistique. L'utilisation de MeV dans cette expérimentation permet de (*i*) gérer l'ensemble de nœuds à travers une interface graphique, (*ii*) appliquer les méthodes de segmentation telles que k-moyenne et la segmentation hiérarchique et (*iii*) visualiser et extraire les groupes de nœuds en format compatible avec d'autres outils pour l'interprétation des résultats.

Les deux méthodes de segmentation présentées dans la Section 5.3, permettent de créer des groupes de nœuds (ou *clusters*) avec une politique de regroupement différente. En partant d'une décomposition aléatoire, l'algorithme k-moyenne recherche d'une manière itérative k groupes en évaluant les nœuds à chaque itération par rapport aux barycentres. Tandis que, la segmentation hiérarchique associe initialement un groupe à chaque nœud. La fusion des nœuds est ensuite basée sur la mesure de la distance euclidienne la plus faible. Au final, une bonne méthode de segmentation est celle qui permet de créer peu de groupes avec une précision élevée. Un groupe thermiquement homogène contient des nœuds avec des températures proches à chaque instant. Donc, l'erreur dans un groupe peut être mesurée par le maximum de l'erreur relative entre chaque nœud et la moyenne du groupe. Ensuite, l'erreur globale de chaque méthode correspond au maximum de l'erreur calculée pour tous les groupes. La Figure 5.9 compare ainsi cette erreur pour les deux algorithmes avec différents nombres de groupes k. Nous pouvons constater que la segmentation par l'algorithme hiérarchique est plus précise que la méthode k-moyenne avec un petit nombre de groupes (<6). En augmentant la valeur de k, les deux algorithmes décomposent les nœuds en groupes avec une erreur relativement faible (<1% pour un nombre des groupes >10).

Nous avons illustré la méthode sur un cas d'étude, où l'activité est bien localisée au niveau du processeur créant ainsi un seul point chaud. La Figure 5.10 montre une illustration graphique des six groupes de nœuds crées par les deux méthodes de segmentation à partir des données extraites des deux applications Whetstone et MXM. D'une manière générale, les méthodes de segmentation décomposent la surface du cœur selon le niveau thermique des nœuds tenant compte des variations en fonction du temps (caractéristiques thermiques) ; elles regroupent les nœuds qui constituent l'endroit du point chaud (groupe C6). En outre, l'activité produite par le benchmark MXM a peu d'impact sur la variation de la température ; la différence de températures entre les nœuds n'est pas assez élevée. C'est la raison pour laquelle le groupe C6 dans MXM contient plus de nœuds que dans Whetstone (voir tableau 5.2).

5.4.4 Placement des capteurs et monitoring de la température

L'identification des zones thermiquement homogènes à l'aide des algorithmes de segmentation va permettre de placer un nombre limité de capteurs égal au nombre de groupes. Nous souhaitons ainsi évaluer notre méthode de placement de capteurs en considérant comme cas

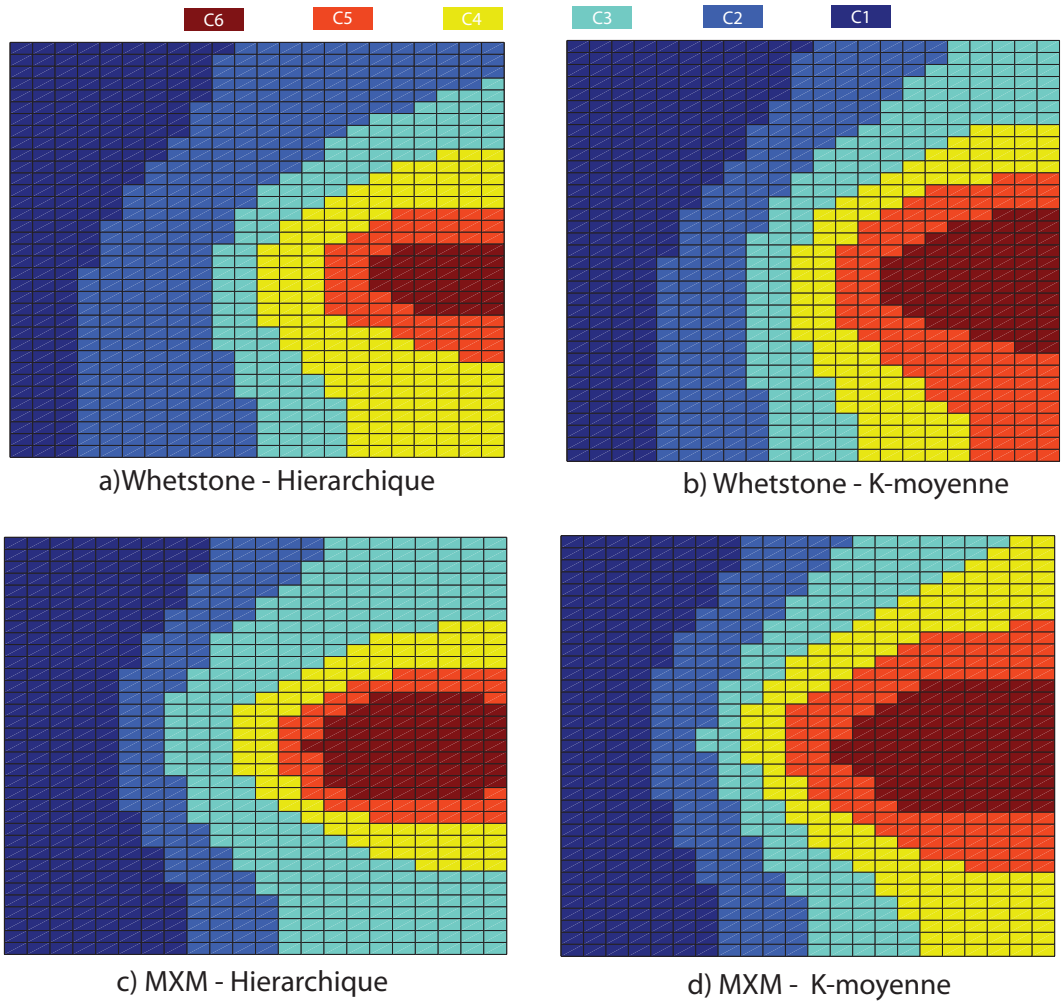
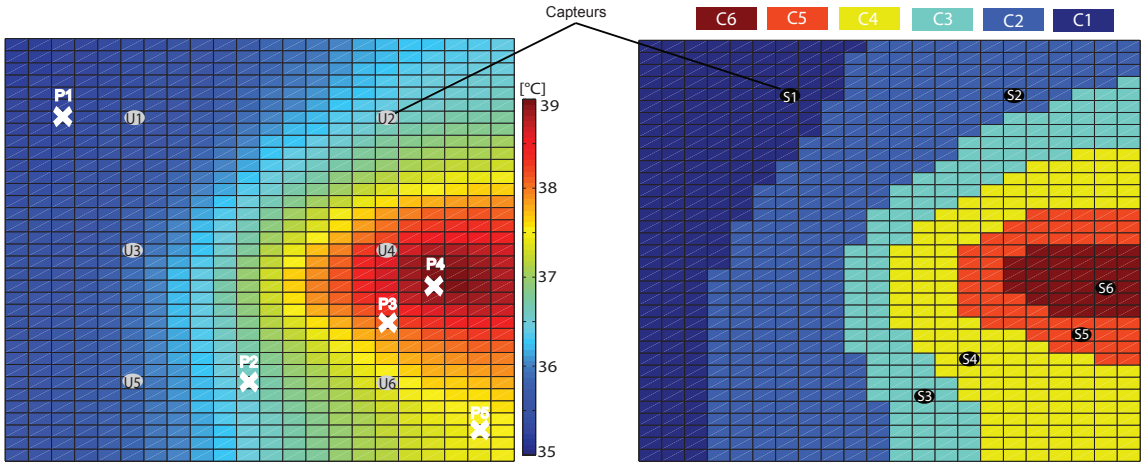


FIGURE 5.10 – Illustration graphique des 6 groupes crées par les méthodes de segmentation.

d'étude les six groupes construits par la segmentation hiérarchique appliqué sur les données extraites précédemment (illustré dans la Figure 5.10-a).



a) Cartographie thermiqe; Emplacement uniforme

b) Les 6 groupes avec l'emplacement des capteurs

FIGURE 5.11 – Placement des capteurs de température sur puce.

Notre méthode permet de définir automatiquement l'emplacement adéquat des capteurs $S_1..S_6$ dans les six groupes $C_1..C_6$ qui constituent la surface du cœur (Figure 5.11-b). Pour chaque groupe C_i, le capteur S_i est associé au nœud qui produit une meilleure estimation de la température moyenne de tout le groupe. Ce nœud est identifié à l'aide des méthodes de sélection d'attributs selon l'approche *wrapper* en utilisant un modèle linéaire simple (voir Section 5.3.3). Ainsi, le monitoring de la température dans n'importe quel point du cœur va dépendre de la position par rapport aux groupes identifiés. Si un point appartient à un groupe C_i, l'estimation de la température dans ce point sera basé sur les mesures du capteur S_i.

Notre méthode est comparée à l'approche uniforme qui propose l'intégration d'un ensemble de capteurs uniformément espacés dans le cœur [NCR10, RCN11]. La Figure 5.11-a) illustre sur une cartographie thermique le placement de six capteurs ($U_1..U_6$) selon cette approche. Le monitoring de la température dans un point est effectué par une approximation à partir des mesures du capteur le plus proche.

Pour comparer les deux méthodes, nous avons choisi l'ensemble de points $P_1..P_5$ représentés dans la Figure 5.11-a). Le point P4 est situé dans une région active où l'activité est maximale produisant un pic de température. P3 et P5 sont moyennement affectés par la température du point chaud. Ainsi, P1 et P2 correspondent à deux points où l'activité est nulle (pas de composant placé dans ces régions) et sont éloignés du point chaud. Le monitoring de la température dans le point P4 par l'approche uniforme est réalisé par le capteur U4. Cependant, notre méthode indique que ce point se situe dans le groupe C_6. Le monitoring de la température dans ce groupe est réalisé par le capteur S_6. La Figure 5.11-d compare les estimations de températures dans P_4 par ces deux capteurs ; il est clair que les estimations de S_6 sont plus précises que U4.

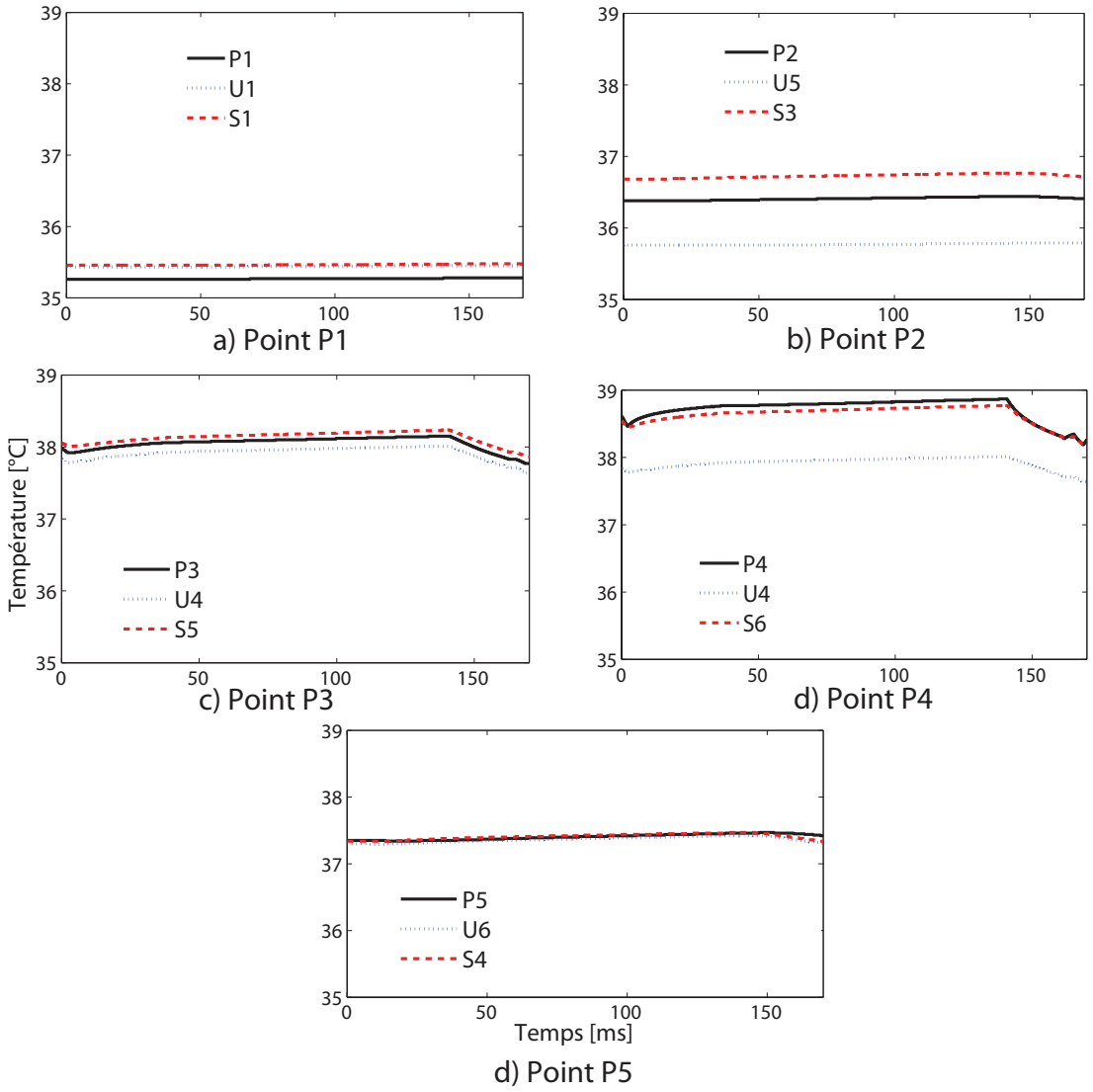


FIGURE 5.12 – Monitoring de la température par les deux méthodes dans les 5 points de comparaison.

C'est aussi le cas pour le monitoring de la température dans P3 par S5 par rapport à U4 (Figure 5.11-c). De plus, les estimations de la température dans le point P5 par U6 et S4 sont très proches. Cela est dû au fait que le capteur U6 se situe dans le même groupe que P5 (C4 dans la Figure 5.11-b). En outre, l'estimation de la température dans P2 par S3 est plus précise que U5 (Figure 5.11-b), lorsque dans le point P1 la température est presque constante où les deux méthodes sont identiques.

Nous avons pu voir que le monitoring de la température selon l'approche uniforme avec peu de capteurs n'est pas suffisant pour avoir une information précise sur la température à n'importe quel point du cœur. Il faut intégrer un grand nombre de capteurs pour essayer de couvrir toute la surface, ce qui est généralement très coûteux. Cependant, nous avons illustré dans ce cas d'étude l'efficacité de l'extraction de l'information pertinente au moment de la conception du circuit pour mettre en œuvre une solution de monitoring de la température précise avec peu de capteurs.

5.4.5 Capteur de température : résolution et coût

Nous avons pu voir dans la section précédente que la résolution spatiale d'une méthode de monitoring de la température dépend essentiellement du nombre de capteurs et de la politique de placement. Cependant, la granularité temporelle dépend des caractéristiques du capteur. Nous rappelons que l'objectif est toujours d'avoir une structure de monitoring à grain fin précise et à faible coût. Afin d'évaluer le coût de la solution de monitoring, nous allons considérer un oscillateur en anneau comme capteur de température. Grâce à sa structure numérique, il est le plus utilisé pour le monitoring de la température dans les circuits reconfigurables FPGA [VLSS].

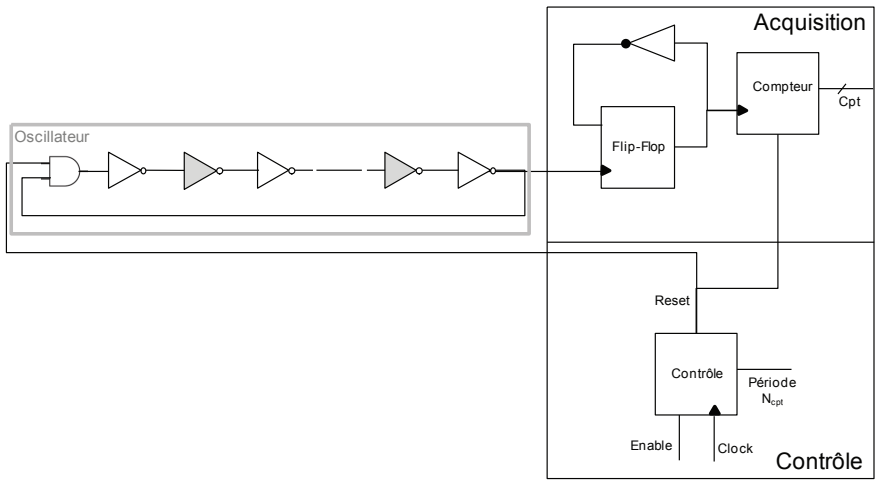


FIGURE 5.13 – Représentation d'un capteur de température à base d'un oscillateur en anneau.

La structure la plus souvent utilisée d'un oscillateur en anneau est basée sur l'utilisation d'un nombre impair d'inverseurs bouclés (la sortie du dernier de la chaine est relié au premier). Cette structure se met ainsi à osciller naturellement lorsqu'elle est alimentée. La fréquence d'oscillation de cette structure est directement fonction des paramètres technologiques du circuit mais aussi de grandeurs physiques telles que la tension d'alimentation du composant ou encore sa température. La mise en place d'un oscillateur en anneau dans un circuit nécessite en plus de la structure du capteur, une unité d'acquisition et de gestion telle que représentées dans la Figure 5.13. Cette unité prend en charge non seulement la gestion du capteur, mais aussi l'accumulation du nombre d'oscillations qui est une fonction de la température. En principe, la fréquence d'oscillation F_{osci} de ce capteur peut s'exprimer selon [Bru12] :

$$F_{osci} = \frac{cpt \times F_{clk}}{N_{cpt}} \tag{5.3}$$

où F_{clk} est la fréquence d'horloge, cpt est la valeur du compteur dans l'unité d'acquisition. N_{cpt} est la période d'échantillonnage des mesures en nombre de cycles d'horloges à F_{clk}. Elle permet de réinitialiser périodiquement le capteur ainsi que le compteur d'acquisition. Ainsi, la résolution temporelle est directement liée à la fréquence d'horloge utilisée et à la période d'échantillonnage : $Rés = \frac{N_{cpt}}{F_{clk}}$ [Bru12]. Ayant choisi une fréquence de fonctionnement qui respecte les contraintes liées au temps de propagation, la résolution temporelle du capteur est généralement suffisante pour détecter les variations fines de température. Par exemple à 25MHz, la résolution maximale est égale à 40ns ce qui est largement suffisant pour détecter les variations fines de température.

Le coût introduit par l'insertion d'un capteur de température est représenté par la surface silicium (ressources alloués dans un FPGA) et par l'énergie consommée qui peut introduise de l'auto échauffement du circuit. La surface totale requise est égale à la surface nécessaire à l'implémentation de l'oscillateur et à la surface de l'unité d'acquisition et de contrôle. La taille d'un oscillateur dépend essentiellement du nombre d'inverseurs, mais occupe généralement peu de ressources sur FPGA. Dans une implémentation sur l'Atlys XC6SLX45, la surface totale d'un capteur avec 3 inverseurs est d'environ 14 slices LUTs et un bloc RAM (RAMB16). Le coût par rapport à la surface occupée par le système sous surveillance SecretBlaze est donc égal à 0.6% de slices LUTs et 2% de RAMB16 par capteur. Cependant, la collection de données des compteurs ainsi que la mise en œuvre de toute la boucle de contrôle n'est pas considérée dans ce calcule. Ceci va dépendre du mécanisme d'adaptation et de l'architecture du système. Par exemple, la mise en place d'une technique d'adaptation au niveau système telle que la migration de tâche, nécessite l'implémentation d'une interface matérielle et logicielle pour accéder aux compteurs à partir de l'interconnexion en spécifiant le capteur comme étant un périphérique du système. En outre, la puissance consommée par un oscillateur en anneau est relativement faible. Elle est d'environ 1 mW et nécessite une exploration par implémentation sur carte pour mieux étudier son impact sur la température.

5.5 Conclusion

Dans ce chapitre, le monitoring de la température des circuits a été développé. Une approche en amont a été proposée : elle introduit un ensemble de techniques pour la segmentation de données extraites du flot de conception, et ce afin de définir une solution optimale en terme de coût et de précision. Dans cette optique, nous avons proposé un flot complet pour l'extraction de données thermique de nos circuits. Grâce à l'instrumentation des logiciels utilisés, nous avons mis en œuvre des méthodes de segmentation pour définir les régions thermiquement homogènes. A partir de cette information, nous avons proposé une solution de placement des capteurs de température sur puce.

L'approche est validée par des expérimentations calibrées sur les résultats de cartographies issues d'une caméra infrarouge. Notre solution permet de reconstituer sur la base des données capteurs la cartographie thermique à une résolution temporelle fine et une précision spatiale ajustable ; sur un exemple constitué d'un SoC implémenté dans une technologie FPGA, nous obtenons une erreur de 1% à une résolution de la centaine de µss/µm. Une comparaison conduite avec les approches uniformes de placement de capteurs montre l'efficacité de notre méthode en terme de nombre de capteurs à utiliser pour le monitoring.

CHAPITRE 6

CONCLUSION

« Les machines un jour pourront résoudre tous les problèmes, mais jamais aucune d'entre elles ne pourra en poser un ! »

Albert Einstein

Sommaire

6.1 Bilan

Les travaux de cette thèse s'insèrent dans le contexte du monitoring des Systèmes-sur-Puce (SoCs) auto-adaptatifs ; des systèmes qui reposent sur un processus bouclé en trois phases : (*i*) le monitoring qui consiste à observer l'état du système, (*ii*) le diagnostic qui analyse les informations relevées pour optimiser le fonctionnement du système, et (*iii*) l'action qui règle les paramètres en conséquence. L'efficacité d'une méthode d'adaptation dépend non seulement de l'algorithme d'optimisation mais aussi de la précision de l'information observée en ligne. Nous avons pu constater que peu de travaux dans la littérature se focalisent sur l'aspect monitoring qui constitue pourtant une phase indispensable de la boucle de contrôle, et que les méthodes industrielles actuelles sont très coûteuses et nécessitent l'insertion d'un grand nombre d'unités pour avoir une information précise sur le comportement du système à une résolution spatiale et temporelle fine.

Dans un contexte où la consommation et la fiabilité apparaissent comme de fortes contraintes, cette thèse s'intéresse plus particulièrement au monitoring de la puissance et de la température sur puce. Nous avons proposé une approche innovante qui intervient en amont pour fournir une solution de monitoring optimale ; un ensemble de techniques issues du domaine de la fouille de données est mis en œuvre pour l'analyse de données extraites des différents niveaux d'abstractions à partir du flot de conception, ce afin de définir une solution optimale en terme de coût et de précision. Notre méthode permet de dégager de manière systématique l'information pertinente requise pour la mise en œuvre d'un monitoring efficace.

La première contribution présentée dans le Chapitre 3, se concentre au niveau matériel et permet de générer des estimations à grain fin de la puissance. Dans cette optique, nous avons réalisé une instrumentation complète du flot de conception dans le but d'extraire les données à différents niveaux d'abstraction. L'ensemble des méthodes supervisées de la fouille de données sont déployées et comparées dans le but d'identifier l'information pertinente qui corrèle avec la puissance du circuit. Les commutations enregistrées à niveau « transfert de registre » fournissent un indicateur fiable et peuvent être observées facilement en ligne par de simples compteurs d'évènements implémentés sur quelques signaux stratégiques. La problématique principale concernait l'intégration de ces dispositifs dans un circuit complexe avec des milliers de signaux internes. Nous avons étudié en premier lieu le problème d'emplacement de ces compteurs. Ensuite, nous avons présenté une méthode globale pour sélectionner les signaux stratégiques les plus importants à observer. Nous avons proposé dans un second temps plusieurs modèles pour l'estimation en ligne de la puissance en fonction de l'activité de commutation à surveiller sur ces signaux. Notre solution est expérimentée sur un cas d'étude constitué d'un système sur puce à base d'un processeur 32 bits implémenté sur FPGA. Grâce à notre méthode, nous avons pu sélectionner un petit nombre suffisant pour un monitoring précis de la puissance.

La second partie concernait l'optimisation de l'utilisation des ressources existantes dans les processeurs pour mettre en œuvre un monitoring de la puissance à coût négligeable. L'analyse des données s'effectue au niveau matériel et logiciel, à l'aide des sondes intégrées (telles que les PMU : "*Performance Monitoring Unit*") pour un monitoring hybride à granularité fine. En effet, il existe un petit nombre de compteurs d'évènement qui peuvent être configurés par un grand nombre d'évènements distincts. L'enjeu est de retrouver le sous-ensemble d'évènements qui permet une estimation précise de la puissance. Une heuristique inspirée des algorithmes de fouille de données a été développée dans ce but, et elle permet de sélectionner automatiquement les évènements ayant le plus d'impact. D'autre part, le monitoring de la puissance à une granularité fine ne dépend pas seulement de l'activité, mais aussi d'un ensemble de paramètres, y compris la fréquence de fonctionnement du processeur et la température externe de la puce. Pour cela, nous avons mis en place un modèle de la puissance qui permet de suivre les variations à différents niveaux de la hiérarchie du système. Il prend en compte les mécanismes d'adaptation (DVFS), et il est robuste aux variations de température.

L'information liée à la température est un élément incontournable dans les systèmes intégrés actuels ; elle augmente la consommation statique, dégrade la fiabilité et accélère le phénomène de vieillissement de la puce. C'est pour cela, nous avons abordé le monitoring de la température dans une troisième contribution (Chapitre 5). Un ensemble d'outils a été mis en place pour réaliser des simulations thermiques de nos circuits. Grâce à l'instrumentation des logiciels utilisés, nous avons mis en œuvre des méthodes de segmentation des données pour localiser les régions thermiquement homogènes. À partir de cette information, nous avons proposé une solution de placement des capteurs de température. L'approche est validée par des expérimentations calibrées sur les résultats de cartographies issues d'une caméra infrarouge. Notre solution permet de reconstituer sur la base des données capteurs la cartographie thermique à une résolution temporelle fine et une précision spatiale ajustable.

6.2 Perspectives

Les travaux engagés au cours de cette thèse représentent un véritable engouement vers une architecture auto-adaptative consciente de son état (*self-awareness*). Nous avons proposé une méthode d'analyse de données qui permet d'aboutir à un monitoring de l'état du système relativement précis et peu coûteux. Les résultats sont prometteurs, et notre solution s'avère très intéressante. Néanmoins, elle nécessite d'être encore plus étoffée afin de couvrir l'étendue du sujet posé au départ. Il paraît donc souhaitable de poursuivre ce travail, et d'en ébaucher quelques axes d'améliorations.

Nous allons donc maintenant aborder ces points d'amélioration tout en mettant en perspective plusieurs idées afin d'approfondir les recherches sur ce thème. Tout d'abord, nous avons

évalué nos méthodes matérielles de monitoring de la puissance (Contribution 1) et de la température (Contribution 3) sur un processeur 32-bits implémenté sur une technologie FPGA. Il serait intéressant d'étendre cette analyse considérant des architectures multi-cœurs plus complexes implémentées sur des technologies différentes. En outre, l'exploration de l'utilisation des ressources dans les processeurs pour le monitoring de la puissance (Contribution 2) a été évaluée sur une plateforme à double cœur où nos modèles estiment la puissance consommée au niveau de toute la plateforme. Une autre perspective serait de mener une étude sur une plateforme plus récente afin de mettre en œuvre des solutions à une granularité plus fine ; par exemple au niveau des cœurs, de la mémoire et des différents éléments du circuit.

En ce qui concerne l'implémentation des méthodes de monitoring, il reste plusieurs pistes à étudier. En effet, nous avons proposé un ensemble de modèles validés sur des données extraites du flot de conception. Une autre perspective serait de mener une étude pour déployer des mécanismes d'apprentissage en ligne afin de garantir la robustesse des modèles utilisés face aux variations technologiques et environnementales. De plus, il serait judicieux de se pencher sur des méthodes d'analyse de données pour mettre en place des modèles prédictifs, permettant au processus de prise de décision d'anticiper des actions afin d'éviter des états indésirables (par exemple une surchauffe qui va endommager le circuit). Dans la même optique, il serait intéressant de s'arrêter sur des méthodes pour combiner les données issues des différents types de capteurs pour mettre en œuvre une solution précise, robuste et peu coûteuse qui estime l'état global du circuit. Enfin, la mise en place d'actionneurs et des prises de décisions associées à différents objectifs d'optimisation pourra être abordées afin de venir boucler la boucle de compensation.

Toutes ces pistes tracent des axes de recherche à part entières. Ce travail de thèse n'était qu'une première lancée modeste vers des systèmes sur puce autonomes et auto-adaptatifs. Beaucoup d'améliorations restent envisageables.

BIBLIOGRAPHIE

[AAE+13a] A Amouri, H Amrouch, T Ebi, J Henkel, and M Tahoori. Accurate Thermal-Profile Estimation and Validation for FPGA-Mapped Circuits. In *Field-Programmable Custom Computing Machines (FCCM), 2013 IEEE 21st Annual International Symposium on*, pages 57–60, 2013.

[AAE+13b] A Amouri, H Amrouch, T Ebi, J Henkel, and M Tahoori. Accurate Thermal-Profile Estimation and Validation for FPGA-Mapped Circuits. In *Field-Programmable Custom Computing Machines (FCCM), 2013 IEEE 21st Annual International Symposium on*, pages 57–60, 2013.

[AES+13] Hussam Amrouch, Thomas Ebi, Josef Schneider, Sridevan Parameswaran, and Jorg Henkel. Analyzing the thermal hotspots in FPGA-based embedded systems. In *Field Programmable Logic and Applications (FPL), 2013 23rd International Conference on*, pages 1–4, 2013.

[Apa] ANSYS Apache. Sentinel-ti thermal simulation.

[ARM10a] ARM. Cortex-A9 Technical Reference Manual, 2010.

[ARM10b] ARM. Intelligent Energy Management (IEM), 2010.

[ARM15a] ARM. ARM DS-5 Development Studio, 2015.

[ARM15b] ARM. ARM Energy Probe, 2015.

[BCBT11] L Barthe, L V Cargnini, P Benoit, and L Torres. The SecretBlaze : A Configurable and Cost-Effective Open-Source Soft-Core Processor. In *Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on*, pages 310–313, 2011.

[Bes07] Roland E Best. *Phase locked loops*. McGraw-Hill Professional, 2007.

[BH99] A Bakker and J H Huijsing. A low-cost high-accuracy CMOS smart temperature sensor. In *Solid-State Circuits Conference, 1999. ESSCIRC '99. Proceedings of the 25th European*, pages 302–305, September 1999.

[BJ12a] Srikar Bhagavatula and Byunghoo Jung. A Low Power Real-time On-Chip Power Sensor in 45-nm SOI. *Circuits and Systems I : Regular Papers, IEEE Transactions on*, 59(7) :1577–1587, 2012.

[BJ12b] W L Bircher and L K John. Complete System Power Estimation Using Processor Performance Events. *Computers, IEEE Transactions on*, 61(4) :563–577, 2012.

[BK01] P Bratek and A Kos. Temperature sensors placement strategy for fault diagnosis in integrated circuits. In *Semiconductor Thermal Measurement and Management, 2001. Seventeenth Annual IEEE Symposium*, pages 245–251, 2001.

[BL97] Avrim L Blum and Pat Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97(1–2) :245–271, 1997.

[Bru12] Florent Bruguier. *Characterization and monitoing methods of technological and environmental changes for adaptive reconfigurable systems*. Theses, Universit{é} Montpellier II - Sciences et Techniques du Languedoc, 2012.

[BTM00] David Brooks, Vivek Tiwari, and Margaret Martonosi. *Wattch : a framework for architectural-level power analysis and optimizations*, volume 28. ACM, 2000.

[BVLJ05] W L Bircher, M Valluri, J Law, and L K John. Runtime identification of microprocessor energy saving opportunities. In *Low Power Electronics and Design, 2005. ISLPED '05. Proceedings of the 2005 International Symposium on*, pages 275–280, 2005.

[BWeWK03] Frank Bellosa, Andreas Weiß el, Martin Waitz, and Simon Kellner. Event-Driven Energy Accounting for Dynamic Thermal Management. In *Proceedings of the Workshop on Compilers and Operating Systems for Low Power (COLP'03)*, 2003.

[CAC11] CACTI. An integrated cache and memory access time, cycle time, area, leakage, and dynamic power model, 2011.

[CCF03] Deming Chen, J Cong, and Yiping Fan. Low-power high-level synthesis for FPGA architectures. In *Low Power Electronics and Design, 2003. ISLPED '03. Proceedings of the 2003 International Symposium on*, pages 134–139, 2003.

[CF94] Rich Caruana and Dayne Freitag. Greedy Attribute Selection. In *In Proceedings of the Eleventh International Conference on Machine Learning*, pages 28–36. Morgan Kaufmann, 1994.

[CH10] J M Chabloz and A Hemani. Distributed DVFS using rationally-related frequencies and discrete voltage levels. In *Low-Power Electronics and Design (ISLPED), 2010 ACM/IEEE International Symposium on*, pages 247–252, 2010.

[CK07] Jian-Jia Chen and Chin-Fu Kuo. Energy-efficient scheduling for real-time systems on dynamic voltage scaling (DVS) platforms. In *null*, pages 28–38. IEEE, 2007.

[CKS+09] Wonil Choi, Hyunhee Kim, Wook Song, Jiseok Song, and Jihong Kim. ePRO-MP : A Tool for Profiling and Optimizing Energy and Performance of Mobile Multiprocessor Applications. *Sci. Program.*, 17(4) :285–294, 2009.

[CW76] Harold J Curnow and Brian A Wichmann. A synthetic benchmark. *The Computer Journal*, 19(1) :43–49, 1976.

[DB07] B Datta and W P Burleson. Low power on-chip thermal sensors based on wires. In *Very Large Scale Integration, 2007. VLSI - SoC 2007. IFIP International Conference on*, pages 258–263, October 2007.

[DBMS79] Jack J Dongarra, James R Bunch, Cleve B Moler, and Gilbert W Stewart. *LINPACK users' guide*, volume 8. Siam, 1979.

[Dev82] A J Devaney. A filtered backpropagation algorithm for diffraction tomography. *Ultrasonic imaging*, 4(4) :336–350, 1982.

[DL97] M Dash and H Liu. Feature Selection for Classification. *Intelligent Data Analysis*, 1 :131–156, 1997.

[dM10] Arnaldo Carvalho de Melo. The new linux perftools. In *Slides from Linux Kongress*, 2010.

[DT14] S Divekar and A Tiwari. Interconnect matrix for multichannel AMBA AHB with multiple arbitration technique. In *Green Computing Communication and Electrical Engineering (ICGCCEE), 2014 International Conference on*, pages 1–5, 2014.

[EDL+04] Dan Ernst, Shidhartha Das, Seokwoo Lee, David Blaauw, Todd Austin, Trevor Mudge, Nam Sung Kim, and Others. Razor : circuit-level correction of timing errors for low-power operation. *IEEE Micro*, (6) :10–20, 2004.

[EFH09] T. Ebi, M. Faruque, and J. Henkel. TAPE : Thermal-aware agent-based power econom multi/many-core architectures. *2009 IEEE/ACM International Conference on Computer-Aided Design - Digest of Technical Papers*, pages 302–309, 2009.

[ERHH11] Thomas Ebi, Holm Rauchfuss, Andreas Herkersdorf, and Jörg Henkel. Agent-based thermal management using real-time I/O communication relocation for 3D many-cores. In *Integrated Circuit and System Design. Power and Timing Modeling, Optimization, and Simulation*, pages 112–121. Springer, 2011.

[FBCP10] J J L Franco, E Boemo, E Castillo, and L Parrilla. Ring oscillators as thermal sensors in FPGAs : Experiments in low voltage. In *Programmable Logic Conference (SPL), 2010 VI Southern*, pages 133–137, 2010.

[FLI] FLIR. FLIR SC6000 Series MWIR Science-Grade Cameras.

[FLI09] FLIR. FLIR Silver SC5000 MWIR, 2009.

[Fri91] Jerome H Friedman. Multivariate adaptive regression splines. *The annals of statistics*, pages 1–67, 1991.

[GBEL10] Jan Gustafsson, Adam Betts, Andreas Ermedahl, and Björn Lisper. The Mälardalen WCET Benchmarks – Past, Present and Future. pages 137–147, Brussels, Belgium, 2010. OCG.

[GE03] Isabelle Guyon and André Elisseeff. An Introduction to Variable and Feature Selection. *J. Mach. Learn. Res.*, 3 :1157–1182, 2003.

[GFHK09] M Gutlein, E Frank, M Hall, and A Karwath. Large-scale attribute selection using wrappers. In *Computational Intelligence and Data Mining, 2009. CIDM '09. IEEE Symposium on*, pages 332–339, 2009.

[Gha12] Mohammad Ghasemazar. Robust Optimization of a Chip Multiprocessor's Performance Under Power and Thermal Constraints. In *Proceedings of the 2012 IEEE 30th International Conference on Computer Design (ICCD 2012)*, ICCD '12, pages 108–114, Washington, DC, USA, 2012. IEEE Computer Society.

[Gol89] David E Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1989.

[GP75] Jean D Gibbons and John W Pratt. P-Values : Interpretation and Methodology. *The American Statistician*, 29(1) :pp. 20–25, 1975.

[GRE+01] M R Guthaus, J S Ringenberg, D Ernst, T M Austin, T Mudge, and R B Brown. MiBench : A free, commercially representative embedded benchmark suite. In *Workload Characterization, 2001. WWC-4. 2001 IEEE International Workshop on*, pages 3–14, 2001.

[Hal98] Mark A Hall. Correlation-based feature selection for machine learning. Technical report, 1998.

[HGV+06] Wei Huang, Shougata Ghosh, Siva Velusamy, Karthik Sankaranarayanan, Kevin Skadron, and Mircea R Stan. Hotspot : Acompact Thermal Modeling Methodology for Early-stage VLSI Design. *IEEE Trans. Very Large Scale Integr. Syst.*, 14(5) :501–513, 2006.

[HO02] Richard Herveille and Others. WISHBONE system-on-chip (SoC) interconnection architecture for portable IP cores. *Revision : B*, 3 :4–32, 2002.

[HRYT00] Michael Huang, Jose Renau, Seung-Moon Yoo, and Josep Torrellas. A framework for dynamic energy efficiency and temperature management. In *Proceedings of the 33rd annual ACM/IEEE international symposium on Microarchitecture*, pages 202–213. ACM, 2000.

[IM03] C Isci and M Martonosi. Runtime power monitoring in high-end processors : methodology and empirical data. In *Microarchitecture, 2003. MICRO-36. Proceedings. 36th Annual IEEE/ACM International Symposium on*, pages 93–104, 2003.

[Inc] Xilinx Inc. Zynq-7000 All Programmable SoC.

[Inf] InfraTec. Infrared camera series VarioCAM hr head xxx.

[ITR13] ITRS. International Technology Roadmap for Semiconductors : Reports and Ordering Information, 2013.

[KP] Paul Kaufmann and Marco Platzner. A Hardware / Software Infrastructure for Performance Monitoring on LEON3 Multicore Platforms.

[KX04] S Kaxiras and P Xekalakis. 4T-Decay Sensors : A New Class of Small, Fast, Robust, and Low-Power, Temperature/Leakage Sensors. In *Low Power Electronics and Design, 2004. ISLPED '04. Proceedings of the 2004 International Symposium on*, pages 108–113, 2004.

[Lev04] John Levon. OProfile manual. *Victoria University of Manchester*, 2004.

[Lia09] Qi Liang. Performance Monitor Counter data analysis using Counter Analyser, 2009.

[LIM13] Charles LIMONARD. The future of power management in the mobile computing market, 2013.

[LMMM08] Jieyi Long, Seda Ogrenci Memik, Gokhan Memik, and Rajarshi Mukherjee. Thermal Monitoring Mechanisms for Chip Multiprocessors. *ACM Trans. Archit. Code Optim.*, 5(2) :9 :1—-9 :33, 2008.

[LPB+11] Suzanne Lesecq, Diego Puschini, Edith Beigné, Pascal Vivet, and Yeter Akgul. Low-cost and robust control of a DFLL for multi-processor system-on-chip. In *Proceedings of the 18th World Congress of IFAC*, 2011.

[LRLZ13] Xin Li, Mengtian Rong, Tao Liu, and Liang Zhou. Research of thermal sensor allocation and placement based on dual clustering for microprocessors. *SpringerPlus*, 2(1) :253, December 2013.

[LSH05] Kyeong-Jae Lee, K Skadron, and Wei Huang. Analytical model for sensor placement on microprocessors. In *Computer Design : VLSI in Computers and Processors, 2005. ICCD 2005. Proceedings. 2005 IEEE International Conference on*, pages 24–27, October 2005.

[LWT+12] Charles Lively, Xingfu Wu, Valerie Taylor, Shirley Moore, Hung Ching Chang, Chun Yi Su, and Kirk Cameron. Power-aware predictive models of hybrid (MPI/OpenMP) scientific applications on multicore systems. *Computer Science - Research and Development*, 27(4) :245–253, 2012.

[Mak10] K A A Makinwa. Smart temperature sensors in standard {CMOS}. *Procedia Engineering*, 5 :930–939, 2010.

[MB08] Andreas Merkel and Frank Bellosa. Task activity vectors : A new metric for temperature-aware scheduling. In *In Third ACM SIGOPS EuroSys Conference*, 2008.

[MBM+09] Almir Mutapcic, Stephen Boyd, Srinivasan Murali, David Atienza, Giovanni De Micheli, and Rajesh Gupta. Processor speed control with thermal constraints. *IEEE Transactions on Circuits and Systems I : Regular Papers*, 56(9) :1994–2008, 2009.

[MBTC13] Imen Mansouri, Pascal Benoit, Lionel Torres, and Fabien Clermidy. Fine-grain dynamic energy tracking for system-on-chip. *IEEE Transactions on Circuits and Systems. Part II, Express Briefs*, 60(6) :4, 2013.

[MJP93] Microsoft Windows Bitmap Format, 1993.

[MM06] R Mukherjee and S O Memik. Systematic temperature sensor allocation and placement for microprocessors. In *Design Automation Conference, 2006 43rd ACM/IEEE*, pages 542–547, 2006.

[MMNL08] Seda Ogrenci Memik, Rajarshi Mukherjee, Min Ni, and Jieyi Long. Optimizing thermal sensor allocation for microprocessors. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 27(3) :516–527, 2008.

[Mur83] Fionn Murtagh. A survey of recent advances in hierarchical clustering algorithms. *The Computer Journal*, 26(4) :354–359, 1983.

[MX09] Tie Meng and Cheng Xu. A cross-coupled-structure-based temperature sensor with reduced process variation sensitivity. *Journal of Semiconductors*, 30(4) :045002, 2009.

[MxM10] Matrix-Matrix Multiplication Timings, 2010.

[NCR10] A Nowroz, R Cochran, and S Reda. Thermal monitoring of real processors : Techniques for sensor allocation and full characterization. In *Design Automation Conference (DAC), 2010 47th ACM/IEEE*, pages 56–61, 2010.

[NMH12] Chalbi Najoua, Boubaker Mohamed, and Bedoui Mohamed Hedi. Accurate dynamic power model for FPGA based implementations. *IJCSI International Journal of Computer Science*, 9(2) :84–89, 2012.

[NR11] Abdullah Nazma Nowroz and Sherief Reda. Thermal and power characterization of field-programmable gate arrays. In *Proceedings of the 19th ACM/SIGDA international symposium on Field programmable gate arrays*, pages 111–114. ACM, 2011.

[OMM08] Umit Y Ogras, Radu Marculescu, and Diana Marculescu. Variation-adaptive feedback control for networks-on-chip with multiple clock domains. In *Proceedings of the 45th annual Design Automation Conference*, pages 614–619. ACM, 2008.

[Ope02] OpenCore. WISHBONE System-on-Chip (SoC) Interconnection Architecture for Portable IP Cores Revision : B.3, 2002.

[Opt] Optris. Camera infrarouge optris PI 40xx.

[PeP+95] J Periaux, G Winter (eds.), Edited J P, G Winter, and John Wiley. Genetic Algorithms In Engineering And Computer Science, 1995.

[Pet11] Thomas Petazzon. Snowball, a new community Linux development platform, 2011.

[PHW+08] B S Paskaleva, M M Hayat, Zhipeng Wang, J S Tyo, and S Krishna. Canonical Correlation Feature Selection for Sensors With Overlapping Bands : Theory and Application. *Geoscience and Remote Sensing, IEEE Transactions on*, 46(10) :3346–3358, October 2008.

[PMV13] Mihai Pricopi, Thannirmalai Somu Muthukaruppan, and Vanchinathan Venkataramani. Power-Performance Modeling on Asymmetric Multi-Cores. In *ACM International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES)*, 2013.

[POW] DOCEA POWER. Thermal Exploration and Validation.

[PP12] Roberta Piscitelli and Andy D. Pimentel. A Signature-Based Power Model for MPSoC on FPGA. *VLSI Design*, 2012 :1–13, 2012.

[RAHP12] C Ruething, A Agne, M Happe, and C Plessl. Exploration of ring oscillator design space for temperature measurements on FPGAs. In *Field Programmable Logic and Applications (FPL), 2012 22nd International Conference on*, pages 559–562, 2012.

[RCN11] S Reda, R J Cochran, and A N Nowroz. Improved Thermal Tracking for Processors Using Hard and Soft Sensor Allocation Techniques. *Computers, IEEE Transactions on*, 60(6) :841–851, 2011.

[RFP13] O Roeva, S Fidanova, and M Paprzycki. Influence of the population size on the genetic algorithm performance in case of cultivation process modelling. In *Computer Science and Information Systems (FedCSIS), 2013 Federated Conference on*, pages 371–376, September 2013.

[RVCC06] Ravishankar Rao, Sarma Vrudhula, Chaitali Chakrabarti, and Naehyuck Chang. An optimal analytical solution for processor speed control with thermal constraints. In *Proceedings of the 2006 international symposium on Low power electronics and design*, pages 292–297. ACM, 2006.

[SAR12] S. Sharifi, R. Ayoub, and T. S. Rosing. TempoMP : Integrated prediction and management of temperature in heterogeneous MPSoCs. *2012 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 593–598, 2012.

[SAS02] Kevin Skadron, Tarek Abdelzaher, and Mircea R Stan. Control-Theoretic Techniques and Thermal-RC Modeling for Accurate and Localized Dynamic Thermal Management. In *Proceedings of the 8th International Symposium on High-Performance Computer Architecture*, HPCA '02, pages 17—-, Washington, DC, USA, 2002. IEEE Computer Society.

[SBM+10] F Sebastiano, L J Breems, K A A Makinwa, S Drago, D M W Leenaerts, and B Nauta. A 1.2-V 10- μW NPN-Based Temperature Sensor in 65-nm CMOS With an Inaccuracy of 0.2 ˆ{\circ} C (3 σ) From - 70 ˆ{\circ} C to 125 ˆ{\circ} C. *Solid-State Circuits, IEEE Journal of*, 45(12) :2591–2601, 2010.

[SD95] Ching-Long Su and Alvin M Despain. Cache Design Trade-offs for Power and Performance Optimization : A Case Study. In *Proceedings of the 1995 International Symposium on Low Power Design*, ISLPED '95, pages 63–68, New York, NY, USA, 1995. ACM.

[Sem13] Freescale Semiconductor. Power Management Integrated Circuit (PMIC) for i.MX50/53 Families, 2013.

[SGG+08] V Salapura, K Ganesan, A Gara, M Gschwind, J C Sexton, and R E Walkup. Next-Generation Performance Counters : Towards Monitoring Over Thousand Concurrent Events. In *Performance Analysis of Systems and software, 2008. ISPASS 2008. IEEE International Symposium on*, pages 139–146, 2008.

[SGH+07] S Santra, P K Guha, M S Haque, S Z Ali, and F Udrea. Si Diode Temperature Sensor beyond 300°C. In *Semiconductor Conference, 2007. CAS 2007. International*, volume 2, pages 415–418, October 2007.

[Shu10] Jack Shue. Thermal Imaging of Power MOSFETs under Thermal Runaway Conditions, 2010.

[SII13] H Sasaki, S Imamura, and K Inoue. Coordinated power-performance optimization in manycores. In *Parallel Architectures and Compilation Techniques (PACT), 2013 22nd International Conference on*, pages 51–61, September 2013.

[SJS13] P Sibi, S Allwyn Jones, and P Siddarth. Analysis of different activation functions using back propagation neural networks. *Journal of Theoretical and Applied Information Technology*, 47(3) :1264–1268, 2013.

[SJT11] Karthik T Sundararajan, Timothy M Jones, and Nigel Topham. Smart cache : A self adaptive cache architecture for energy efficiency. *2011 International Conference on Embedded Computer Systems Architectures Modeling and Simulation*, pages 41–50, 2011.

[SKUY15] Gaurav Singla, Gurinderjit Kaur, Ali K Ogras Unver, and Umit Y. Predictive Dynamic Thermal and Power Management for Heterogeneous Mobile Platforms. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2015*, 2015.

[SMKR97] V Szekely, C Marta, Z Kohari, and M Rencz. CMOS sensors for on-line thermal monitoring of VLSI circuits. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 5(3) :270–276, September 1997.

[SOM+07] Makoto Saen, Kenichi Osada, Satoshi Misaka, Tetsuya Yamada, Yoshitaka Tsujimoto, Yuki Kondoh, Tatsuya Kamei, Yutaka Yoshida, Ei Nagahama, Yusuke Nitta, and Others. Embedded SoC resource manager to control temperature and data bandwidth. In *2007 IEEE International Solid-State Circuits Conference. Digest of Technical Papers*, 2007.

[SR10] S Sharifi and T S Rosing. Accurate Direct and Indirect On-Chip Temperature Sensing for Efficient Dynamic Thermal Management. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 29(10) :1586–1599, October 2010.

[SSH+03] Kevin Skadron, Mircea R Stan, Wei Huang, Sivakumar Velusamy, Karthik Sankaranarayanan, and David Tarjan. Temperature-aware microarchitecture. *ACM SIGARCH Computer Architecture News*, 31(2) :2–13, 2003.

[SYN15] SYNOPSYS. PrimeTime PX : Signoff Power Analysis, 2015.

[Sys] DIAS Infrared Systems. PUROVIEW 768N.

[SZS10] Bing Shi, Yufu Zhang, and Ankur Srivastava. Dynamic Thermal Management for Single and Multicore Processors Under Soft Thermal Constraints. In *Proceedings of the 16th ACM/IEEE International Symposium on Low Power Electronics and Design*, ISLPED '10, pages 165–170, New York, NY, USA, 2010. ACM.

[Tec14] Linear Technology. 6-Bit ?? ADC with Easy Drive Input Current Cancellation and I2C Interface Title of Citation, 2014.

[vdBCGB07] J W van den Brand, C Ciordas, K Goossens, and T Basten. Congestion-Controlled Best-Effort Communication for Networks-on-Chip. In *Design, Automation Test in Europe Conference Exhibition, 2007. DATE '07*, pages 1–6, 2007.

[VLSS] S. Velusamy, J. Lach, M. Stan, and K. Skadron. Monitoring temperature in FPGA based SoCs. *2005 International Conference on Computer Design*, pages 634–637.

[Wil12] Thomas Wilhalm. Intel Performance Counter Monitor - A better way to measure CPU utilization, 2012.

[WJMC04] Qiang Wu, Philo Juang, Margaret Martonosi, and Douglas W Clark. Formal online methods for voltage/frequency control in multiple clock domain microprocessors. In *ACM SIGPLAN Notices*, volume 39, pages 248–259. ACM, 2004.

[WM14] Ratapong Wongpiang and Pornsiri Muenchaisri. Comparing Heuristic Search Methods for Selecting Sequence of Refactoring Techniques Usage for Code Changing. In *International MultiConference of Engineers and Computer Scientistis (IMECS2014)*, volume I, 2014.

[WMW09] Yefu Wang, Kai Ma, and Xiaorui Wang. Temperature-constrained Power Control for Chip Multiprocessors with Online Model Estimation. In *Proceedings of the 36th Annual International Symposium on Computer Architecture*, ISCA '09, pages 314–324, New York, NY, USA, 2009. ACM.

[WRF+10] M Ware, K Rajamani, M Floyd, B Brock, J C Rubio, F Rawson, and J B Carter. Architecting for power management : The IBM #x00AE ; POWER7 #x2122 ; approach. In *High Performance Computer Architecture (HPCA), 2010 IEEE 16th International Symposium on*, pages 1–11, January 2010.

[Xila] Xilinx. MicroBlaze Processor Reference Guide EDK 12.1.

[Xilb] Xilinx. Spartan-6 FPGA Configurable Logic Block User Guide.

[XYC88] Lei Xu, Pingfan Yan, and Tong Chang. Best first strategy for feature selection. In *Pattern Recognition, 1988., 9th International Conference on*, pages 706–708 vol.2, November 1988.

[ZA12] R Zamani and A Afsahi. A study of hardware performance monitoring counter selection in power modeling of computing systems. In *Green Computing Conference (IGCC), 2012 International*, pages 1–10, 2012.

[ZLSC10] Qiusha Zhu, Lin Lin, Mei-Ling Shyu, and Shu-Ching Chen. Feature Selection Using Correlation and Reliability Based Scoring Metric for Video Semantic Detection. In *Semantic Computing (ICSC), 2010 IEEE Fourth International Conference on*, pages 462–469, September 2010.

[ZXD+08] Xiuyi Zhou, Yi Xu, Yu Du, Youtao Zhang, and Jun Yang. Thermal Management for 3D Processors via Task Scheduling. In *Parallel Processing, 2008. ICPP '08. 37th International Conference on*, pages 115–122, September 2008.