Titre: Title:	Caractérisation automatisée de la consommation de puissance des processeurs pour l'estimation au niveau système
Auteur: Author:	Fellipe Medeiros Monteiro
Date:	2013
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Medeiros Monteiro, F. (2013). Caractérisation automatisée de la consommation de puissance des processeurs pour l'estimation au niveau système [Master's thesis, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/1229/

UNIVERSITÉ DE MONTRÉAL

CARACTÉRISATION AUTOMATISÉE DE LA CONSOMMATION DE PUISSANCE DES PROCESSEURS POUR L'ESTIMATION AU NIVEAU SYSTÈME

FELLIPE MEDEIROS MONTEIRO

DÉPARTEMENT DE GÉNIE INFORMATIQUE ET GÉNIE LOGICIEL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION

DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES

(GÉNIE INFORMATIQUE)

AOÛT 2013

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé:

CARACTÉRISATION AUTOMATISÉE DE LA CONSOMMATION DE PUISSANCE DES PROCESSEURS POUR L'ESTIMATION AU NIVEAU SYSTÈME

présenté par : MEDEIROS MONTEIRO Fellipe

en vue de l'obtention du diplôme de : Maîtrise ès Sciences Appliquées

a été dûment accepté par le jury d'examen constitué de :

M. BELTRAME Giovanni, Ph.D., président

M. BOIS Guy, Ph.D., membre et directeur de recherche

M. ABDI Samar, Ph.D., membre et codirecteur de recherche

Mme NICOLESCU Gabriela, Doct., membre

REMERCIEMENTS

Je tiens à remercier chaleureusement le professeur Guy Bois pour son support, ses conseils et son encadrement tout au long de ma maîtrise. Je remercie également Marc-André Cantin, Michel Rogers-Vallée, Laurent Moss et Hubert Guérard pour leur précieuse aide, le temps qu'ils m'ont accordé et leur apport au projet grâce à leur expertise dans le domaine des systèmes embarqués.

Mes remerciements vont également au ReSMIQ et à l'École Polytechnique de Montréal de m'avoir accordé les bourses nécessaires à la continuité de mes études. Je ne peux passer sous silence l'équipe du GRM (Groupe de Recherche en Microélectronique) notamment Réjean Lepage et Jean Bouchard pour le soutien technique et leur disponibilité.

Un gros merci en particulier à mes amis Gabriel Cartier, Vincent Legault, Jean-François Cartier et Gabriel Oshiro pour leurs sages conseils et pour toutes nos enrichissantes discussions. Finalement, je remercie mes parents, Candido Monteiro et Margareth Medeiros, ma sœur, Mirella Monteiro, ainsi que ma copine, Amaya Olivo, pour leur appui indéfectible dans tous mes projets particulièrement dans mes études universitaires.

RÉSUMÉ

De nos jours, la consommation de puissance est une contrainte clé et une métrique de performance essentielle lors du design des systèmes numériques. La dissipation de chaleur excessive sur les circuits intégrés diminue relativement leurs performances. Également, plus que jamais, nous avons le besoin d'augmenter le temps de vie des batteries de nouvelles électroniques portables. Avec les techniques de design classiques, RTL « Register Transfer Level », une estimation de puissance précise est possible seulement aux dernières étapes du processus de développement. Pour remédier à cette problématique, on a récemment proposé dans la littérature de hausser le niveau d'abstraction de la conception de systèmes embarqués à l'aide de la méthodologie de niveau système « Electronic System Level » (ESL). Dans cette perspective, ce travail propose une méthodologie capable de caractériser automatiquement la consommation de puissance des processeurs configurable de type *« soft-processors »* et de générer un modèle efficace pour l'estimation de l'énergie consommée au niveau système. À l'aide de ce modèle, une étude comparative entre trois techniques d'estimation est donc présentée. Les résultats de cinq programmes tests montrent une estimation de puissance huit mille fois plus rapide que les techniques d'estimation conventionnelles et une erreur moyenne de seulement ±3.98 % pour le processeur LEON3 et de ±10.70 % pour le processeur Microblaze.

ABSTRACT

Nowadays, power consumption is a key constraint and a digital system design essential metric of performance. Excessive heat dissipation of integrated circuits relatively decreases the performance of the system. Also, more than ever, we need to increase the battery lifetime of new portable electronics. With classical design techniques as RTL « Register Transfer Level », precise power estimation is only possible in the final stages of the development process. To solve this problem, the literature recently proposed to raise the abstraction level of embedded systems design, using ESL « Electronic System Level » methodology. In this context, this project proposes a methodology to automatically characterize configurable soft-processors power consumption and generate an effective power model for energy consumption estimation at system level. Using this model, a comparative study between three estimation techniques is also presented. The results of five benchmarks show that our power estimation is eight thousand times faster than conventional estimation techniques and an average error of only ±3.98 % for the LEON3 processor and ±10.70 % for the Microblaze processor.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

ESL	« Electronic System Level »
FPGA	« Field Programmable Gate Array »
MIPS	« Micropocessor without Interlocked Pipeline Stage »
SoC	« System On Chip »
SPARCv8	« Scalable Processor ARChitecture version 8 »
RISC	« Reduced Instruction Set Computer »
EDK	« Embedded Development Kit »
GPL	« General Public License »
MAC	« Multiplier and Accumulator »
RTL	« Register Transfer Level »
XML	« Extensible Markup Language »
CB	« Coût de Base »
CI	« Coût Inter-instructions »
EQM	« Erreur Quadratique Moyenne »
ADL	« Architecture Description Language »
BRAM	« Block Random Access Memory »
DSP	« Digital Signal Processor »
ASIC	« Application Specific Integrated Circuit »
FAST	« FPGA-Accelerated Simulation Technologies »
HTLP	« Hierarchical Transaction Level Power »
UPF	« Unified Power Format »
EDA	« Electronic Design Automation »

FLPA	« Functional Level Power Analysis »
ILPA	« Instruction Level Power Analysis »
PLD	« Programmable Logic Device »

CHAPITRE 1 INTRODUCTION

Au fil des dernières années, la société moderne a été témoin de l'importante évolution qu'a subi l'électronique numérique. Nous pouvons facilement percevoir cette évolution dans les appareils portables que nous utilisons quotidiennement. De nos jours, les cellulaires ne sont pas seulement des appareils téléphoniques, ils contiennent une caméra photographique, une caméra vidéo, un GPS ainsi qu'une connexion internet. Le cellulaire est devenu un ordinateur portable aussi petit qu'un porte-monnaie. La diminution des composantes matérielles permet aux ingénieurs de diminuer la taille des appareils électroniques et d'augmenter leur performance. Techniquement, ce progrès est dû principalement à la constante diminution de la taille des transistors, élément de base d'un système numérique. Ceci permet la conception des puces avec beaucoup plus de transistors, avec une plus haute fréquence d'opération et avec plus de fonctionnalités, sans toutefois augmenter la taille de ces mêmes puces.

Dû à cette évolution, la consommation d'énergie des systèmes numériques a considérablement augmenté. Selon [1], la consommation de puissance est une contrainte clé et une métrique de performance essentielle lors du design de systèmes numériques. En conséquence, les optimisations au niveau logiciel ou bien matériel lors du design sont devenues fondamentales pour respecter les exigences au niveau de la consommation d'énergie. Par exemple, dans le cas des appareils mobiles, les chercheurs tentent d'obtenir plus de puissance de calcul tout en conservant le même budget d'énergie pour ainsi ne pas diminuer le temps de vie des batteries [2].

Les effets de l'augmentation de la consommation d'énergie sont multiples. Soulignons en autres :

- **La réduction du temps de vie du circuit :** Une grande dissipation de puissance et de chaleur peut endommager le système, ce qui diminuerait son temps de vie.
- **L'augmentation du coût pour refroidir la puce :** Une augmentation d'énergie consommée implique une plus grande chaleur dissipée, ce qui demande un système de

refroidissement plus efficace. Ceci peut impliquer une augmentation du coût final du produit.

- **La diminution du temps de vie des batteries :** Ceci est un élément important pris en considération par le marché. Les utilisateurs préfèrent les produits qui ont une plus grande autonomie.

1.1 Problématique

Avec les techniques de design classiques RTL *« Register Transfer Level »*, une estimation de puissance précise d'un système embarqué est seulement possible aux dernières étapes du processus de développement. Sans mentionner que pour obtenir cette estimation il faut synthétiser, placer, faire le routage de l'implémentation, simuler le système et finalement utiliser un outil pour obtenir l'estimation à bas niveau. Ceci est un processus très long. À titre d'exemple, selon [3] la simulation de 2.78 millisecondes de temps d'exécution d'une multiplication de matrice par le processeur embarqué Microblaze de Xilinx [4] prend au tour de 3 heures avec le simulateur ModelSim de Mentor Graphics [5]. L'analyse de l'activité du circuit lors de la simulation par XPower Analyzer de Xilinx [6], outil qui calcule l'énergie consommée par le circuit, prend une autre heure de plus. Indépendamment du temps requis, à ce point du projet, un changement devient une tâche très couteuse et fastidieuse. Et, la mise au point de ce changement est souvent remise en question.

Pour remédier à cette problématique, on a récemment proposé dans la littérature de hausser le niveau d'abstraction de la conception de systèmes embarqués à l'aide des méthodologies de niveau systèmes [7]. Plus spécifiquement, nous référerons à la conception niveau système (en anglais *« Electronic System Level »* abrégé par ESL). Cette méthodologie vise la conception d'une plate-forme virtuelle, c'est-à-dire un modèle fonctionnel d'une architecture logicielle/matérielle, qui abstrait des détails d'implémentation. Il est possible de simuler très rapidement cette implémentation, toujours à haut niveau, et d'avoir un aperçu de son comportement et de sa performance.

La modélisation à un haut niveau d'abstraction est intéressante pour faire une vérification exhaustive du fonctionnement du système. Toutefois, à partir de ce modèle il n'est pas possible de synthétiser le circuit afin d'obtenir un prototype. Il faut toujours utiliser la méthodologie RTL dans un deuxième temps, cependant à ce moment le design sera déjà défini et des changements tardifs, normalement, ne seront pas nécessaires.

Néanmoins, la modélisation au niveau système fournit quelques outils capables d'estimer l'énergie consommée par les composantes matérielles du circuit, mais encore beaucoup de travail reste à faire pour les améliorer. Ces outils permettent aux ingénieurs de rapidement comprendre l'impact de leur modèle dans la consommation de puissance et ainsi explorer plusieurs architectures logicielles/matérielles pour trouver un bon compromis entre consommation de puissance et performance. Ceci est illustré sur la Figure 1-1.

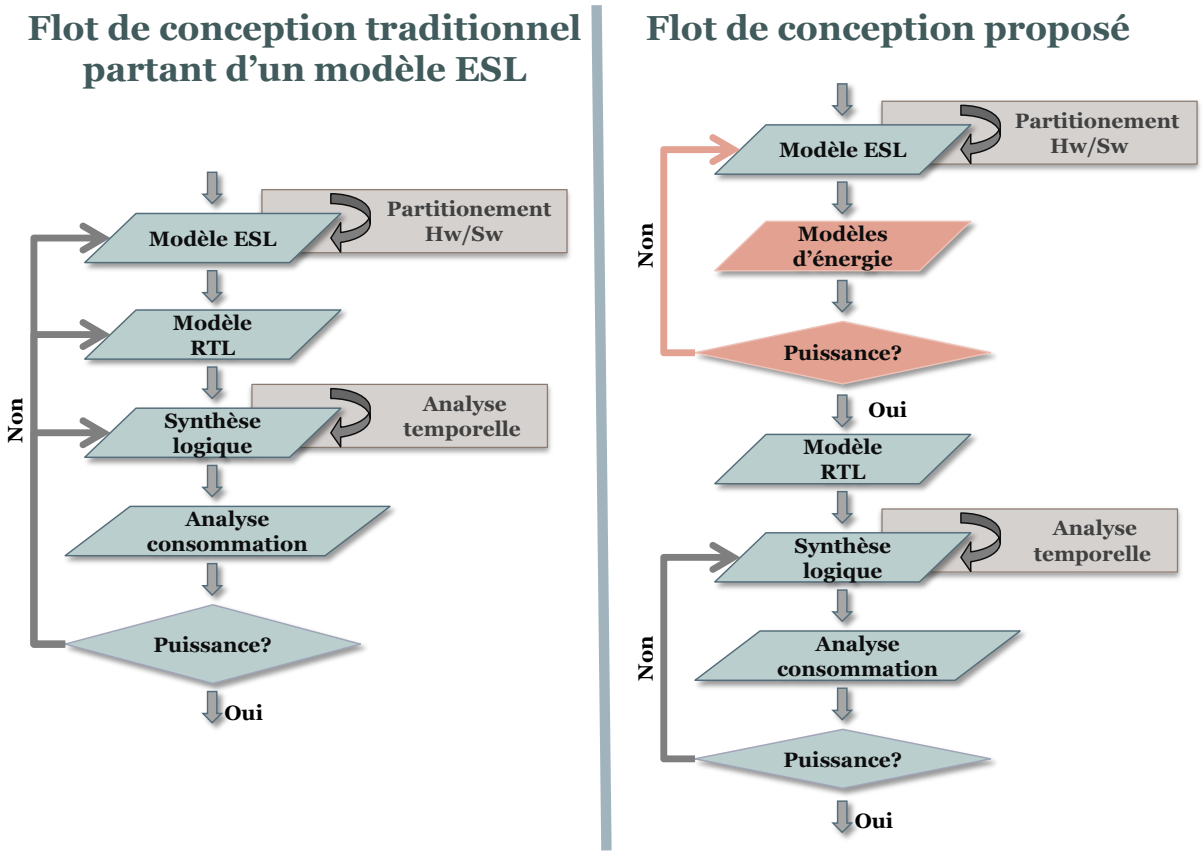

Figure 1-1 Flot de conception ESL traditionnel versus flot de conception proposé

1.2 Objectif

Les circuits configurables FPGA *« Field Programmable Gate Array »* sont devenus très populaires dans le développement des systèmes embarqués. Étant donné que les systèmes embarqués ont une durée de vie de plus en plus courte, la possibilité de reconfigurer le circuit fait du FPGA un choix intéressant pour les ingénieurs lorsque le volume le permet. Cette popularisation a mis en évidence les processeurs configurables de type *« soft-processors »*. Cette gamme de processeurs est implémentée sur les ressources configurables disponibles sur un FPGA et permet le développement de système sur puce (de l'anglais *« System On Chip »* abrégé par SoC).

Dans cette perspective, l'objectif principal de ce travail est d'implémenter un outil capable de caractériser automatiquement la consommation de puissance des *« soft-processors »* et de générer un modèle d'énergie efficace pour l'estimation de l'énergie consommée au niveau système. Ce modèle d'énergie sera basé sur la consommation de puissance de chaque instruction du processeur.

Un second objectif est d'utiliser le modèle d'énergie généré pour comparer trois techniques d'estimation de l'énergie consommée. Cela nous permettra d'évaluer ces techniques et de proposer la plus appropriée lors d'une estimation de puissance au niveau système.

1.3 Méthodologie

Ce projet se divise en quatre phases, soit la revue de littérature, la familiarisation avec un projet récent, le développement et l'évaluation.

Phase 1 : Revue de littérature

Grâce à la revue de littérature, il sera possible d'identifier les avancées des autres équipes de recherche dans ce domaine. Ceci constituera un point essentiel de départ pour la continuité de ce

projet. Au terme de cette activité, nous aurons rassemblé quelques travaux similaires, ceux-ci seront utilisés comme base de comparaison en vue de déterminer la qualité de notre travail.

Phase 2 : Familiarisation

La phase de familiarisation consistera à étudier un premier projet qui a été fait sur l'estimation de puissance des systèmes sur puce à notre laboratoire dans le passé [8]. À cette étape, une lecture exhaustive du code informatique déjà implémenté sera nécessaire. Cette lecture permettra d'apprendre exactement comment l'ancien estimateur de puissance a été conçu. De même, il sera possible d'identifier les faiblesses de l'algorithme et d'éventuelles améliorations à apporter au code et à la méthodologie utilisée.

Phase 3 : Développement

Le développement consistera, dans un premier temps, en une étude préliminaire du modèle à être implémenté et, dans un deuxième temps, en implémenter l'outil en tant que tel. L'étude préliminaire est consacrée à démontrer que le temps d'exécution requis par l'outil pour générer le modèle d'énergie est convenable. Pour ce faire, un petit modèle de test sera développé. Ce modèle considérera seulement une partie réduite du jeu d'instruction du processeur, mais en tenant compte des proportions, le temps requis pour générer un modèle incluant toutes les instructions du processeur pourra être estimé.

Phase 4 : Évaluation

Cette activité consistera à évaluer la qualité du modèle d'énergie généré. Une comparaison avec un modèle de référence déterminera la performance de la méthodologie de caractérisation appliquée par l'outil automatisé. Également, cette dernière étape déterminera quelle technique d'estimation d'énergie consommée s'avère la meilleure. La performance des techniques sera évaluée en fonction de la précision, de l'effort demandé et du temps de développement requis. Les trois techniques se serviront du seul modèle d'énergie généré par l'outil automatisé pour faire ses estimations.

1.4 Contribution

Précisément, ce travail vise à contribuer à l'état de l'art par plusieurs éléments. Tout d'abord, cet outil logiciel automatisera la caractérisation de l'énergie consommée par les *« soft-processors »*. Il sera capable de profiler le jeu d'instruction du processeur et de générer le modèle d'énergie sans l'intervention de l'usager. Selon [2], la caractérisation manuelle d'un processeur MIPS *« Micropocessor without Interlocked Pipeline Stage »* peut prendre jusqu'à trois mois de travail pour un ingénieur. Avec notre outil, le temps requis sera réduit à quelques semaines. L'une des contributions les plus importantes de ce projet est la portabilité de l'outil. La qualité du logiciel sera priorisée, les concepts de la programmation orientée objet seront utilisés afin d'obtenir un logiciel modulaire et générique. De telles manières, le port vers d'autres processeurs sera simple et nécessitera peu de changements.

De plus, le modèle d'énergie généré servira à comparer trois techniques d'estimation d'énergie au niveau système, soit *instruction sans transition*, *instruction avec transition unique* et *instruction avec transition*. Une telle étude permettra de conclure les avantages et les désavantages de chacune de ces techniques.

Plus précisément, nous allons appliquer le flot automatisé sur les processeurs Microblaze et LEON3. À partir du modèle d'énergie généré, une étude comparative entre les trois techniques mentionnées sera présentée.

Au niveau système, le modèle d'énergie généré pourra éventuellement être intégré dans une plate-forme virtuelle telle que SpaceStudio [9, 10]. Cette dernière permet en autres de faire du co-design logiciel/matériel, de faire l'exploration architecturale et d'obtenir de métriques de performances de façon non intrusive. L'intégration du modèle d'énergie dans le système de profilage de SpaceStudio signifierait l'ajout d'une nouvelle métrique de performance pour la plate-forme virtuelle. Une estimation de puissance efficace serait possible de manière simple et rapide.

Ce mémoire est divisé comme suit. Le Chapitre 2 présente une revue de littérature de l'état de l'art. Les méthodes d'estimation de puissance à plusieurs niveaux d'abstraction sont abordées. Au Chapitre 3 on se concentrera sur les techniques d'estimation de puissance à haut niveau d'abstraction spécifique aux processeurs embarqués. De plus, les trois techniques d'estimation de puissance qui seront étudiés dans ce travail seront décrites. Le Chapitre 4 présente les détails d'implémentation de l'outil qui génère automatiquement le modèle d'énergie des processeurs configurables de type *« soft-processors »*. Puis, le Chapitre 5 décrit notre environnement de test et présente les résultats obtenus, soit la qualité du modèle d'énergie et les performances de chaque technique d'estimation. Finalement, le Chapitre 6 effectue une synthèse des travaux et propose des travaux futurs.

CHAPITRE 2 REVUE DE LITTÉRATURE

Avec l'augmentation de la complexité des systèmes sur puce, la consommation de puissance des périphériques ainsi que des processeurs est devenue une préoccupation pour les concepteurs de circuits intégrés. Les processeurs, en particulier, sont essentiels aux systèmes embarqués utilisés dans les équipements électroniques tels que les caméras digitales, les téléphones mobiles, les plateformes de jeu portable entre autres. Le rapport entre la qualité et la performance de ces dispositifs ne se résume plus à la performance de leurs fonctionnalités, mais aussi au temps de vie de leurs batteries. La demande pour la réduction de la taille et du poids des appareils est directement associée à la taille de la batterie, normalement, cette dernière détermine sa grandeur. De plus, il y a un intérêt à diminuer la dissipation de chaleur sur une puce. Pour faire face à cette réalité et à tous ces nouveaux défis, des méthodologies de conception qui visent la diminution de la consommation de puissance ont été développées. Ces méthodologies font partir de plusieurs champs de recherche explorés par les scientifiques. Il existe quatre branches d'études principales qui s'adressent à ce problème [11] : les bibliothèques ciblant un développement de basse consommation [12, 13] (communément désigné par le terme anglais *« low-power library development »),* les techniques d'optimisation pour une basse consommation [14, 15] (communément désigné par le terme anglais *« low-power optimization techniques »*), les techniques d'estimation de puissance [11, 16] (communément désigné par le terme anglais *« power estimation techniques »*) et les outils de synthèse de basse consommation [17] (communément désigné par le terme anglais *« low-power synthesis tools »*). Ces quatre approches proposent, chacune à sa manière, des solutions pour la consommation d'énergie. Dans ce travail, nous allons uniquement nous concentrer sur la troisième approche, soit les techniques d'estimation de puissance.

Ce chapitre passe en revue les travaux pertinents à la présente recherche liée au domaine de l'estimation de puissance. Cette revue de littérature permettra donc de mieux située le travail. Nous passerons en revue les travaux portant sur les méthodes d'estimation de puissance à plusieurs niveaux d'abstraction. Les méthodes présentées sont majoritairement spécifiques aux composants implémentés sur FPGA, toutefois un certain nombre de travaux peuvent aussi être

appliqués à la conception ASIC *« Application Specific Integrated Circuit »*. Par la suite, le Chapitre 3 détaillera les aspects des méthodologies d'estimation de puissance spécifiques aux processeurs.

2.1 Les techniques d'estimation de puissance à plusieurs niveaux

Un système embarqué est constitué d'une partie matérielle et d'une partie logicielle. La partie matérielle est composée de couches hiérarchisées où chaque couche supérieure cache des détails de la couche inférieure immédiate. Chaque couche (ou niveau d'abstraction) représente la même fonctionnalité du modèle exprimé par un format de modélisation différent (Figure 2-1). Par exemple, la description au niveau système généralement comporte une description fonctionnelle du design, tandis que la description RTL consiste en une description architecturale et inclut le flot de données entre les registres ainsi que les opérations logiques appliquées sur les signaux. Par contre, la description au niveau porte logique contient le schéma de connexions entre les portes logiques et les bascules. Finalement, une description au niveau transistor décrit comment les transistors sont connectés entre eux en différentes couches de métaux, silicium et autres matériaux [18].

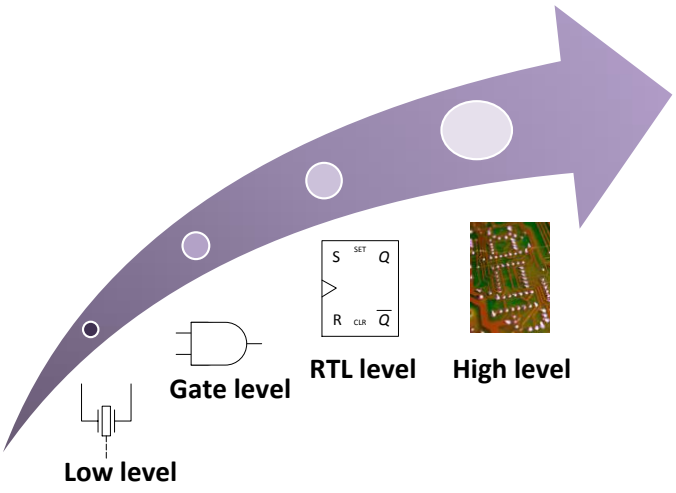

Figure 2-1 Les différents niveaux de modélisation

À chaque niveau de modélisation, différentes techniques pour réduire la consommation du système peuvent être appliquées. À titre d'exemple, on peut penser aux techniques connues comme « *multiple voltage domains* », « *clock gating* » et « *dynamic frequency scaling* ». Toutefois, aux niveaux plus bas la modélisation du matériel est plus limitée et un changement, pour le plus simple qu'il soit, requiert beaucoup d'effort pour être appliqué et pour être validé dû à la grande quantité d'information que constitue le modèle à ce niveau. Ainsi, une petite optimisation peut devenir une tâche très complexe. D'après une étude réalisée par Mentor Graphics et LSI Logic [19, 20], les techniques disponibles au niveau RTL peuvent conduire à une réduction de 20 % de la consommation de puissance; celles aux niveaux logiques permettent 10 % de réduction; alors que celles appliquées à la modélisation de bas niveau permettent de réduire la consommation de seulement 5 %. À l'opposé, la description au niveau système a un très grand potentiel d'optimisation. Les techniques à ce niveau peuvent atteindre une réduction de la consommation de 80 % (Figure 2-2). Pourtant, lorsqu'on augmente le niveau d'abstraction, les opportunités de réduire la consommation de puissance se multiplient significativement. Ainsi, actuellement, l'industrie et la communauté scientifique mettent un grand effort dans l'étude et développement des outils d'estimation efficace au niveau système. En effet, les outils d'estimation viennent complémenter les outils d'optimisation de puissance, ce qui permettra à l'industrie d'améliorer leur flot de conception afin d'étudier la consommation d'énergie d'une plateforme à partir de sa description de haut niveau. À ce niveau le design est encore très flexible et un changement majeur est possible pour un faible coût.

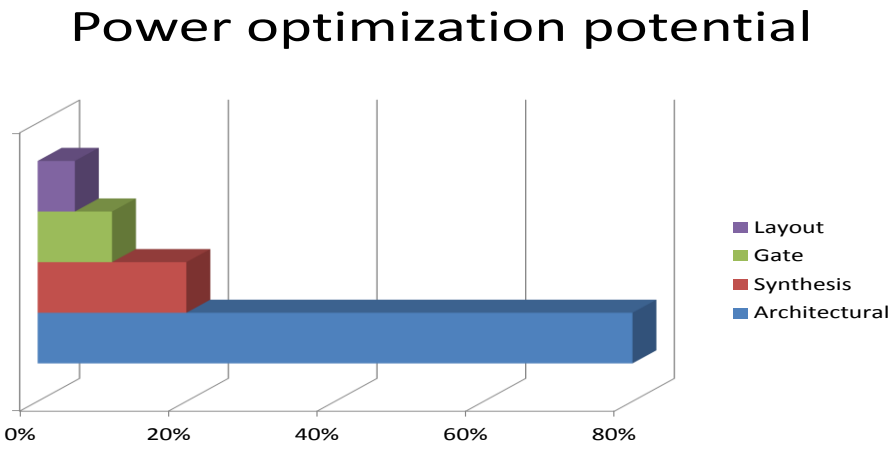

Figure 2-2 Potentiel de réduction de la puissance à différents niveaux [20]

Dans les prochaines sous-sections, nous allons passer en revue plusieurs travaux de recherche qui portent sur les techniques d'estimation de puissance. Afin de faciliter la compréhension du lecteur, ces travaux ont été séparés en différentes catégories.

2.1.1 Mesure physique

La modélisation de la consommation d'énergie est une approche très effective pour estimer la puissance consommée sur un design en particulier. Toutefois, elle ne sera jamais aussi précise qu'une mesure physique. Cette technique consiste à mesurer le courant qui circule sur la puce, normalement, à l'aide d'un multimètre, en sachant le voltage d'entrée et le voltage de sortie la puissance moyenne consommée peut être facilement obtenue.

Évidemment, cette technique d'estimation peut être utilisée seulement dans les dernières étapes de développement d'un système numérique. Ce dernier doit être déjà implémenté sur la puce FPGA ou bien, dans le cas d'un ASIC, un prototype de la puce doit être déjà fabriqué. Dans ces cas, les mesures sont importantes pour de question de test et vérification. Un de grands avantages d'une telle méthode est la rapidité avec laquelle les résultats sont obtenus. Il n'est pas nécessaire

d'exécuter une longue simulation qui demande beaucoup de calculs avec des modèles complexes. Dans [21], la mesure physique a été utilisée pour comparer la consommation d'un FPGA de la compagnie Xilinx et d'un *« Programmable Logic Device »* (PLD) de la compagnie Altera [22]. L'expérimentation a testé les mêmes designs dans les deux composants. En moyenne, le PLD de Altera a consommé 33 % moins de puissance que le FPGA de Xilinx. De même, en [23] la même méthode a été employé pour comparer la consommation de trois filtres digitaux différents implémentés sur un FPGA. Dans ces deux expériences, le but n'était pas d'utiliser la consommation de puissance en tant que métrique lors d'une exploration architecturale, mais plutôt de comparer la consommation entre deux scénarios différents. Malgré tout, le résultat fourni par cette technique se limite à la consommation de puissance moyenne sur un certain intervalle de temps. De plus, dans le cas d'un design implémenté sur FPGA, la consommation de chaque composant ne peut pas être obtenue séparément. Le résultat obtenu se réfère à la consommation de tous les périphériques implémentés sur la puce FPGA.

Il est possible de trouver dans la littérature, des outils améliorés qui mesurent la consommation physiquement. En [24], les auteurs présentent un outil capable de mesurer la puissance consommée au cycle près. L'outil consiste en un ensemble de convertisseurs analogique numérique, une interface de communication USB, un système automatique d'acquisition de données ainsi qu'un logiciel de contrôle. De plus, l'outil peut obtenir le minimum, le maximum et la moyenne de l'énergie consommée par cycle.

2.1.2 Bas niveau

Naturellement, d'autres approches ont été étudiées, développées et proposées pour remplacer la mesure physique. La majorité des techniques d'estimation actuelles a comme base les travaux de Farid N. Najm. Ce dernier a publié plusieurs travaux sur ce sujet. Une de ces plus importantes publications [11], qui est devenu le point de départ de plusieurs recherches, classe les techniques d'estimation bas niveau comme étant probabiliste ou bien statistique. Le premier groupe de technique se base sur la probabilité de chaque entrée appliquée au circuit pour estimer la puissance. Plus spécifiquement, ces techniques propagent la probabilité de transition de chaque

signal d'entrée sur toute la logique interne du circuit pour ainsi connaître l'activité de tous les nœuds. Afin d'obtenir l'effet de toutes les combinaisons de signaux d'entrée, une simulation exhaustive de chaque entrée possible est demandée, ce qui est peu pratique. Des alternatives et améliorations à cette approche sont la simulation probabiliste [25, 26] et la corrélation spatio-temporelle [27]. Typiquement, pour de grands circuits, une méthode probabiliste aura une erreur autour de 10 % sur l'estimation totale de la consommation de puissance du circuit. Cependant, l'erreur d'estimation de puissance consommée par porte logique est potentiellement beaucoup plus grande et probablement inacceptable [28].

Déjà, les techniques statistiques consistent à simuler répétitivement plusieurs fois le circuit à partir des modèles de simulation traditionnels. Les vecteurs d'entrée de la simulation sont générés aléatoirement par l'usager. Normalement, la sortie de la simulation est surveillée et analysée par une méthode de Monte-Carlo, de sorte qu'elle se poursuit jusqu'à une métrique d'évaluation de la précision soit rencontrée. Un avantage de ces méthodes est la possibilité de spécifier la précision voulue avant de débuter les simulations. Ceci permet d'établir un compromis entre la vitesse et la précision de la méthode. Pour obtenir seulement une estimation de la consommation moyenne du circuit, l'exécution peut être très rapide. Toutefois, dans un tel scénario, il ne sera pas possible d'obtenir une estimation individuelle des nœuds internes. Selon [29], une estimation statistique qui utilise une méthode de Monte-Carlo permet d'obtenir une meilleure précision avec la même vitesse d'exécution que les estimations probabilistes. De plus, la technique statistique est implémentée plus facilement et peut être intégrée aux outils de simulation logique déjà existants.

Une grande problématique des techniques d'estimation de puissance de bas niveau est la modélisation de la consommation de micro-impulsion. La transition logique d'une bascule se fait simultanément avec la transition logique de l'horloge à laquelle elle est connectée. Ainsi une bascule ne fait qu'une transition logique par cycle. On ne peut en dire autant des portes logiques. Celles-ci peuvent éprouver de multiples transitions dans un même cycle dû aux délais de signaux qui se propagent de la sortie de diverses bascules jusqu'à l'entrée des portes logiques. Ces transitions sont des micro-impulsions mieux connues sous le nom anglais de « *glitch* ». Évidemment, ces micro-impulsions causent une consommation de puissance. En accord avec

[28], ceci peut représenter 20 % de la consommation totale d'un petit circuit ou même 70 % de la consommation totale d'un grand circuit dépendant de sa structure. Ce facteur est très coûteux à calculer, parce qu'il dépend des connexions entre les signaux à l'intérieur du circuit. Conséquemment, plusieurs techniques d'estimation de puissance ignorent cette problématique et ainsi sous-estiment la consommation réelle. Toutefois, d'autres travaux ont étudié cette problématique et ont développé des modèles pour calculer la consommation de puissance due à ces micro-impulsions [30, 31].

2.1.3 Niveau RTL

Les techniques d'estimation au niveau RTL peuvent être regroupées en deux grandes catégories : les techniques analytiques et les techniques d'estimation par macromodèles. Il est important de noter que plusieurs variations de ces méthodes existent afin de s'adapter mieux à différentes parties du design (arbre d'horloge, entrée/sorties, mémoire et logique de contrôle).

2.1.3.1 Les techniques analytiques

En général, les techniques analytiques tentent de corréler la consommation d'énergie dynamique à des variables du design, telle que la quantité de portes logiques, la quantité d'éléments mémoire, l'activité de signaux entre autres. L'approche analytique définit la consommation d'énergie dynamique par une équation mathématique où les inconnues représentent ces variables. Cependant, à partir d'une description matérielle au niveau RTL plusieurs de ces variables ne sont généralement pas encore connues. Ainsi, l'usager doit estimer les valeurs de ces variables. Un exemple d'approche analytique est fourni en [32], ce travail applique l'équation suivante pour calculer la consommation de puissance moyenne d'un circuit :

$$P = \sum_{i}^{n} GE_i(E_{typ} + C_L^i * V_{dd}^2)f * A_{int}^i \qquad (2.1)$$

où GE_i est le nombre de porte logique qu'un bloc fonctionnel i contient, E_{typ} est l'énergie moyenne consommée par une porte logique lorsqu'elle est active, C_L^i et V_{dd} sont respectivement

la capacité moyenne et le voltage appliqué à porte logique, f est la fréquence et finalement A_{int}^{i} est l'activité moyenne des portes logiques qui font partie du bloc fonctionnel i. Enfin, pour calculer la consommation moyenne de puissance, la sommation est appliquée sur chaque bloc fonctionnel du design.

Une méthode analytique d'estimation de puissance très connue et utilisée dans l'industrie de la conception FPGA est l'approche par table (communément désigné par le terme anglais *« spreadsheet based approach »*). Il s'agit d'une approche très simple et facile d'emploi. Un grand avantage est que l'usager n'a pas à apprendre l'utilisation d'un outil complexe afin d'obtenir une estimation de la consommation de puissance. De plus, cette approche peut être utilisée au tout début du cycle de développement lorsque plusieurs décisions importantes sont prises ayant une grande influence dans les étapes subséquentes. Normalement, les informations qui doivent être fournies à un *« spreadsheet »* en général sont la fréquence d'opération, l'activité du circuit et la quantité de blocs mémoires. Cette méthode est utile lors de la planification du projet, mais elle ne peut pas fournir de valeurs précises sur la consommation du matériel. En début de projet les concepteurs n'ont souvent pas une idée claire de tous les blocs fonctionnels qui composeront un grand circuit. Les *« spreadsheets »* permettent alors d'obtenir rapidement une estimation des ressources requises pour l'implémentation du système ainsi que sa demande énergétique. Cette information est fondamentale pour planifier le système d'alimentation, ainsi que le système de refroidissement permettant la dissipation thermique. Toutefois, une utilisation efficace de cette méthode exige beaucoup d'expérience dans le développement de systèmes numériques. Faire une estimation minimalement précise de plusieurs paramètres du design au tout début du développement d'un système embarqué est une tâche qui demande un ensemble de connaissances qui est acquise seulement après une longue expérience. Il existe plusieurs outils commerciaux qui permettent d'appliquer l'approche par table, notamment l'outil « XPower Estimator » [33] de la compagnie Xilinx et l'outil « PowerPlay Estimator » [34] de la compagnie Altera. Dans les deux cas, les outils ciblent la conception FPGA. Ces outils sont capables d'estimer le courant électrique ainsi que la puissance consommée pour différentes familles de FPGA.

Les techniques analytiques exigent cependant que l'usager puisse estimer l'activité du circuit. Une mauvaise estimation de ce paramètre est évidemment une source d'erreur. Ceci a motivé l'arrivée d'un ensemble de techniques analytiques dotées de la théorie de Shanon. Cette dernière estime l'activité des blocs logiques basés sur l'entropie des signaux d'entrée et de sortie [35, 36], sans toutefois, simuler les blocs logiques du circuit. Ces techniques se basent seulement sur les caractéristiques de la séquence d'entrée et sur quelques détails de la composition interne du circuit. D'autres techniques plus évolués [37] se sont dotés de la théorie de la décision afin de trouver l'architecture possédant le meilleur compromis performance versus puissance consommée.

Actuellement, les vendeurs de FPGA fournissent une gamme d'outils d'estimation de puissance au niveau RTL plus sophistiqué que les *« spreadsheets »*. À part, « XPower Analyzer » de Xilinx et « PowerPlay Power Analysis » de Altera pour la conception FPGA, il existe aussi « SmartPower » de Actel [38]. Pour la conception ASIC, il en existe plusieurs dont « PrimeTime PX Power Analysis » de Synopsys [39]. XPower fait partie de la première génération des outils destinés à l'estimation de la consommation de puissance des FPGAs fondée sur les cellules SRAM commercialisées par Xilinx. L'outil base ces calculs d'estimation sur un modèle d'analyse discrète (aussi connu par *« Lumped capacitance model »* en anglais) [40]. La méthode employée par cet outil peut être considérée comme une variation de l'approche analytique. Par exemple, dans le cas de XPower, la description RTL du design doit être synthétisée. Le processus de synthèse peut cibler une liste d'interconnexions structurales, qui contient une description purement structurale du design, ou bien une liste d'interconnexions post-placement et routage. D'une part, la première liste permet une estimation extrêmement rapide, d'autre part la deuxième liste permet une estimation beaucoup plus précise. Ceci dit, l'estimation fournie par cette gamme d'outils consiste en une estimation de la puissance dynamique et une estimation de la puissance statique. La consommation statique est due au courant de fuite. Elle peut être calculée de manière précise à travers une équation mathématique qui a comme variables la température, le voltage d'alimentation et la superficie du FPGA utilisée. La puissance dynamique quant à elle est due à l'activité dans le circuit. Elle est linéairement proportionnelle à la capacitance de commutation, au carré du voltage d'alimentation, à l'activité du circuit et à la fréquence d'opération;

$$P = C * V^2 * \alpha * f \quad (2.2)$$

Encore une fois, l'activité du circuit peut changer grandement la précision du résultat. La méthode la plus appropriée pour obtenir l'activité avec une bonne précision est la simulation. Dépendamment de comment la description RTL a été synthétisée, la simulation peut être structural ou temporelle. Étant donné que la simulation temporelle fournit le comportement exact des signaux, incluant les micro-impulsions, l'activité obtenue est plus réaliste. Par contre, la simulation structurale s'exécute plus rapidement.

Les méthodes utilisées par les outils des autres compagnies que nous avons citées plus haut sont similaires à celles de XPower.

2.1.3.2 Les macromodèles

Une problématique de l'analyse est d'établir un compromis entre le temps et la précision de l'estimation. Pour obtenir une estimation précise, il faut bien évaluer l'activité du circuit. Une méthode efficace est la simulation logique qui normalement est un processus lent. Afin de remédier à cette problématique, il est possible d'avoir recours à la macromodélisation. La stratégie appliquée par cette méthode est de mesurer la consommation d'une implémentation existante lors de l'exécution de plusieurs scénarios différents et produire un modèle basé sur ces mesures.

Trois techniques de macromodélisation, bien connues dans la littérature, sont le *3DTab* [41], l'*EqTab* [42] et l'*e-HD* [43].

Un macromodèle *3Dtab* estime la consommation de puissance selon trois propriétés des signaux : la probabilité des signaux d'entrée P_{in}, la densité de transition des signaux d'entrée D_{in} et la densité de transition des signaux de sortie D_{out}. Le modèle est représenté par une table de correspondance en trois dimensions, tel que chaque triplet (P_{in}, D_{in}, D_{out}) correspond à une valeur

de puissance. Ainsi, la première étape pour construire un macromodèle consiste à caractériser le circuit. Premièrement, pour chaque paire valide (P_{in}, D_{in}), plusieurs séquences d'entrée doivent être générées. Ensuite, pour chacune de ces paires la consommation de puissance du circuit est déterminée par une technique d'estimation de plus bas niveau. Finalement, le D_{out} est évalué et la moyenne de toutes les valeurs de puissance avec le même triplet (P_{in}, D_{in}, D_{out}) est sauvegardée dans un tableau [44]. Ainsi, l'estimation de la consommation de puissance consiste à simuler la description RTL et collecter les statistiques P_{in}, D_{in} et D_{out} qui correspondent à une entrée dans le tableau. Si le triplet ne correspond pas directement à une entrée dans le tableau, une interpolation est utilisée pour trouver la valeur la plus proche.

Au lieu de compter sur la densité de transition des signaux d'entrée et de sortie, le macromodèle *EqTab* considère la contribution individuelle de chaque bit d'entrée et chaque bit de sortie. Ainsi la consommation de puissance est modélisée par l'équation suivante :

$$P = c_0 + c_1 * D_{in}(0) + c_m * D_{in}(m) + c_{m+1} * D_{out}(1) + c_{m+n} * D_{out}(n) \qquad (2.3)$$

où, $D_{in}(x)$ et $D_{out}(x)$ sont la densité de transition d'entrée et de sortie mesurées au $x^{ième}$ bit alors que n et m représentent respectivement la taille de vecteurs de bits d'entrée et de sortie. Une table de correspondance indexée par le couple (P_{in}, D_{in}) est créée et la valeur correspondante à chaque couple d'entrée est multipliée par les coefficients de l'équation 2.3. Ainsi, l'estimation de puissance consiste à simuler le design en RTL, déterminer (P_{in}, D_{in} et D_{out}), trouver les coefficients dans la table de correspondance et finalement appliquer l'équation pour faire l'estimation de puissance.

Essentiellement, le macromodèle *e-HD* consiste en une équation qui exprime la puissance en fonction de deux propriétés de signaux différentes de celles déjà présentées, la distance de *Hamming* et le nombre de bits stables entre deux séquences d'entrée successives. La distance de *Hamming* correspond au nombre de positions où deux suites de symboles de même longueur se différencient. La technique n'utilise pas les signaux de sorties.

Au cours des dernières années, les macromodèles présentés ci-haut ont été employés dans plusieurs travaux plus élaborés. Par exemple, l'outil « PrEsto » [45] génère automatiquement, à partir de la statistique de plusieurs séquences de valeurs d'entrées, un modèle linéaire de puissance. Ce macromodèle donne la possibilité d'estimer la puissance consommée à chaque cycle d'exécution. Cet outil a été utilisé pour estimer la consommation de puissance du processeur LEON3 et d'un ARM cortex A8. Un simulateur FAST (abréviation anglaise de *« FPGA-Accelerated Simulation Technologies »*) a été utilisé pour recueillir les données nécessaires aux estimations.

Bien que la macromodèlisation soit une technique très appliquée dans la littérature [46] chaque méthode de macromodélisation fait certaines hypothèses qui conduisent à une sorte de limitations intrinsèque, ce qui affecte la précision et qui empêchent une utilisation plus répandue de ces dernières.

2.1.4 Niveau système

Plus récemment, plusieurs travaux portant sur les méthodologies d'estimation de puissance au niveau système ont été publiés. Les défis (complexité des modèles, temps requis pour une estimation et coût d'un changement) rencontrés par les architectes de système numérique pour réduire la consommation de puissance à partir des techniques de bas niveau ont motivé l'étude plus approfondie sur les possibilités d'estimer la consommation de puissance directement au niveau système.

Dans les méthodologies de conception au niveau système, la spécification des processus se fait généralement à un haut niveau d'abstraction, les calculs de chaque module se font à un niveau comportemental et leurs communications se font à un niveau transactionnel [47]. À ce niveau d'abstraction, le modèle n'a aucune connaissance de l'implémentation matérielle future. Cette spécification est un premier modèle du design. Les méthodes de raffinement et de synthèse permettent l'obtention d'une implémentation finale du modèle où les modules s'ordonnancent et se communiquent par un protocole précis au cycle près.

L'absence d'information architecturale au niveau système limite le potentiel d'une estimation précise de la consommation de puissance à ce niveau. Toutefois, au fil des dernières années, plusieurs approches ont été proposées afin de repousser cette limitation.

Une spécification de haut niveau définit un module comme étant une unité fonctionnelle qui exécute plusieurs instructions ou transactions successivement. Dans cette perspective, une première solution au manque d'information est de représenter le comportement d'un module par une machine à états. Les transactions ou bien les instructions qui couvrent l'ensemble d'actions du module sont identifiées et chacune se voit assigner un état différent. À travers une caractérisation, une consommation moyenne est assignée à chacun des états. La caractérisation est faite à partir d'une implémentation du module de plus bas niveau. Ainsi une technique d'estimation de bas niveau doit absolument être utilisée. La granularité de transactions sur lesquelles le modèle de puissance s'appuie peut, évidemment, avoir une influence sur la précision de la méthode. Narayanan et al. [48] utilise un arbre nommé HTLP (en anglais *« Hierarchical Transaction Level Power »*) qui est intégré dans l'environnement de simulation de haut niveau SystemC-TLM d'un design particulier. Cet arbre représente la structure de transactions possibles, chaque nœud correspond à une transaction et pour chaque transaction une puissance moyenne est assignée. Ainsi, à partir d'une trace des transactions, la puissance consommée peut être estimée. La même approche est appliquée par Caldari et al. [49] en un bus AMBA AHB. Toutefois, au lieu d'assigner une puissance moyenne à chaque transaction, le bus a été décomposé en plusieurs blocs logiques et une équation mathématique décrit le comportement énergétique de chacun de ces blocs.

Une autre façon d'aborder ce problème est d'utiliser les bibliothèques de macromodèle ciblant les plates-formes virtuelles en SystemC. Un bon exemple pour illustrer cette approche est la plate-forme PowerSC [50]. Cette dernière permet d'estimer la puissance consommée par un design à partir de sa description en SystemC. PowerSC intègre une bibliothèque contenant plusieurs modèles de puissance (e.g. des composants tel qu'additionneur et multiplicateur) ciblant une technologie en particulier. De plus, elle fournit un moyen facile d'obtenir l'activité de signaux

lors d'une simulation de la plate-forme virtuelle SystemC. Ceci est possible parce que les objets SystemC ont été spécialisés afin de surveiller leur propre valeur pendant la simulation. Plus spécifiquement, les objets SystemC ont la capacité de mettre automatiquement à jour plusieurs informations chaque fois qu'une transition doit être effectuée. Ainsi, à la fin de la simulation il est possible d'obtenir des statistiques telles que le nombre de transitions sur chaque signal et la probabilité de transition sur chaque signal. Cette information est, donc, fournie comme entrée à la bibliothèque contenant les modèles de puissance et l'énergie consommée est finalement estimée. En [51], une approche très similaire est employée. À partir du code C ou SystemC fournit en entrée à l'outil (et préalablement modifié ou annoté), l'activité des signaux peut être facilement sauvegardée. Toutefois, dans ce cas, la puissance de chaque modèle SystemC doit être déterminée à partir d'une équation mathématique fournie par l'usager. Cette gamme de technique est importante afin d'avoir une idée de la consommation du circuit et ainsi prendre de décision tôt dans le développement.

Finalement, des travaux récents [52] proposent l'utilisation de la synthèse de haut niveau pour estimer la puissance consommée d'un modèle au niveau système. La synthèse de haut niveau permet de générer un bloc matériel au niveau RTL à partir d'une description en un langage de haut niveau. La méthodologie propose la modélisation du design avec le langage ESTEREL [53]. À l'aide de l'outil Esterel Studio, les modèles RTL sont obtenus par la synthèse comportementale. Ensuite, une corrélation est établie entre les variables définies dans la description de haut niveau et les signaux correspondants dans l'implémentation RTL. Enfin, une simulation du modèle de haut niveau permet d'extraire l'activité des signaux RTL. La corrélation entre les signaux implique notamment que seulement la simulation du modèle de haut niveau est nécessaire. Ce qui diminue grandement le temps de simulation requis. L'activité des signaux RTL est finalement fournie en entrée à l'estimateur de puissance RTL PowerTheater. Ce logiciel a été acheté en 2009 par la compagnie Synopsys [54].

L'industrie de la conception assistée par ordinateur pour l'électronique (abrégé en anglais par l'acronyme EDA pour *« Electronic Design Automation »*) a répondu à la demande pour de systèmes numériques moins dispendieux énergétiquement avec plusieurs solutions dans le flot de

conception ciblant les designs à basse consommation. Afin de contribuer à cette vague, un consortium formé par les grandes compagnies de l'industrie EDA a récemment défini le standard UPF [55] (sigle anglais pour *« Unified Power Format »*). Ce standard fournit un format cohérent pour spécifier divers aspects de la gestion d'énergie d'un système électronique, notamment le réseau d'alimentation, les commutateurs, l'isolement, la conservation et d'autres. Les langages de description matérielle, tel que VHDL et Verilog fournissent un riche ensemble de fonctionnalités nécessaires pour capturer la spécification fonctionnelle de systèmes numériques, toutefois aucune fonctionnalité ne permet de définir les aspects de la gestion d'énergie (par exemple comment chaque élément du système doit être alimenté). Enfin, le but est d'intégrer les contraintes UPF dans le flot de conception RTL classique.

Dans une perspective de développement au niveau système plusieurs de concepts et de composants définis par UPF ne s'appliquent pas [56]. Une solution à ce manque de détails à haut niveau est fournie par la plateforme nommée *PwARCH* [56]. Celle-ci génère les contraintes UPF à partir de la description en niveau système.

2.1.5 Émulation

La grande majorité de méthodologies d'estimation de puissance se résume typiquement en trois séquences d'opérations: simulation du circuit pour l'obtention de l'activité de signaux (ou bien les statistiques des entrées de composants), génération d'un modèle de puissance pour chaque composant basé sur les valeurs obtenues lors de la simulation et addition de la consommation individuelle de chaque composant pour calculer la consommation totale. Normalement, l'étape de simulation, un processus qui demande beaucoup de temps, doit être répétée plusieurs fois. Ce qui peut nuire à l'implémentation pratique d'une approche qui requiert plusieurs longues simulations. Une solution à cela est la technique d'émulation [57-59], celle-ci est déjà largement utilisée pour accélérer la vérification fonctionnelle. La technique d'émulation consiste à ajouter dans l'implémentation finale du design un module matériel responsable pour estimer la puissance consommée par le circuit. Le module matériel surveille plusieurs variables du circuit et estime la puissance consommée à partir du modèle de puissance qu'il implémente. L'ajout du module

matériel évite d'avoir à simuler le système embarqué, l'activité des signaux étant obtenue pendant l'exécution (donc en temps réel). Alors qu'une accélération considérable est atteinte avec une telle technique, un défi s'impose dû à l'augmentation de la taille du circuit lorsque le module matériel pour l'estimation de puissance est ajouté. Néanmoins, pour certains cas, si la taille du circuit devient problématique, il est possible de mettre le modèle d'analyse de la puissance en logiciel. Un inconvénient de cette technique est que l'estimation de puissance ne peut pas être obtenue au début du cycle de développement. Les résultats obtenus par une telle technique confirment si le budget en terme énergétique a été respecté.

CHAPITRE 3 ESTIMATION DE LA CONSOMMATION DE PUISSANCE DE PROCESSEURS AU NIVEAU SYSTÈME

Ce chapitre poursuit la revue des travaux portant sur les méthodologies d'estimation de puissance au niveau système. Cependant, nous allons approfondir les méthodologies qui ciblent spécifiquement les processeurs embarqués. Ce chapitre présente également les trois techniques d'analyse de puissance au niveau instruction qui seront étudiées plus en détail tout au long de ce travail.

3.1 Niveau fonctionnel

Les méthodologies d'estimation de puissance au niveau système ciblant spécifiquement les processeurs embarqués peuvent être classées en trois catégories : l'analyse de puissance au niveau fonctionnel (en anglais *« Functional Level Power Analysis »* abrégé par FLPA), l'analyse de puissance au niveau instruction (en anglais *« Instruction Level Power Analysis »* abrégé par ILPA) ainsi que les modèles hybrides. L'analyse de puissance au niveau fonctionnel a été introduite dans [60]. La modélisation FLPA, divise le processeur en plusieurs blocs fonctionnels, par exemple l'unité d'extraction (en anglais « *fetch unit* »), l'unité arithmétique logique, la banque de registre entre autres. L'objectif est d'identifier les blocs qui ont un impact sur la consommation de puissance totale du processeur. Après les avoir bien identifiés, chacun de ces derniers doit être décrit individuellement par une expression arithmétique en fonction des paramètres algorithmiques et architecturaux. Les paramètres algorithmiques sont spécifiques à l'algorithme exécuté, comme le taux de *« cache-miss »* et de *« cache-hit »*, tandis que les paramètres architecturaux sont choisis par le concepteur notamment la fréquence d'opération du processeur ainsi comme la longueur des mots. Ensuite, la consommation individuelle de chaque bloc doit être calculée pour plusieurs combinaisons différentes de ces paramètres. Ces différentes combinaisons sont obtenues par l'exécution de plusieurs séquences de code assembleur. Finalement, les données acquises sont analysées afin d'obtenir l'expression arithmétique de chaque bloc fonctionnel.

Au cours des dernières années, la technique FLPA a été étendue et améliorée. Un aspect qui différencie les travaux qui appliquent la FLPA est la séparation appropriée de l'architecture du processeur en blocs fonctionnels. Pour souligner un exemple réel, en [61] la méthodologie FLPA est appliquée pour caractériser un DSP (de l'anglais *« Digital Signal Processor »*) à architecture VLIW (abréviation anglaise pour *« Very Long Instruction Word »)* de Texas Instruments [62], un processeur spécialisé en application multimédia. L'architecture de ce dernier a été divisé en six blocs fonctionnels; l'arbre d'horloge, la cache d'instruction, la cache de données, l'unité arithmétique logique, l'unité d'extraction et la mémoire interne. Également, la méthodologie FLPA est utilisée en [63] pour générer un modèle d'énergie du « soft-processor » PowerPc405. Le résultat a été intégré dans un modèle SystemC du processeur permettant l'estimation de puissance à partir d'une simulation de haut niveau. Enfin, basé sur la méthodologie FLPA, SoftExplorer [64] est un outil d'analyse qui inclut une bibliothèque de modèles de puissance couvrant plusieurs blocs fonctionnels d'un processeur. Un profilage rapide du code est nécessaire pour déterminer les paramètres d'entrée des modèles de puissance. Et, une estimation relativement précise se fait dans un délai très court.

3.2 Niveau instruction

Le problème d'estimation de la consommation d'énergie au niveau instruction a d'abord été étudié par Tiwari [65] au cours des années 90. Il a introduit la méthodologie ILPA. Celle-ci associe chaque instruction du processeur à un coût d'énergie fixe normalement nommé le coût de base de l'instruction. Ce coût est dû au traitement de base nécessaire pour exécuter l'instruction en question. Le coût de base d'une instruction peut varier en fonction de différents opérandes et des différentes valeurs d'adresse. Bien que les techniques plus appropriées peuvent donner la valeur exacte du coût de base pour n'importe quelle valeur d'opérande ou d'adresse, en pratique cette variation est, seulement, un effet secondaire [65]. En général, le coût énergétique total d'un programme est la somme des coûts de base de chaque instruction exécutée. Néanmoins, lors de l'exécution d'un programme certains effets inter-instructions se produisent et si seulement le coût de base est pris en considération, la contribution de ces effets inter-instructions ne sera pas incluse dans l'énergie totale consommée. Le coût inter-instructions est défini comme l'énergie dissipée lors de la transition entre deux instructions différentes exécutées consécutivement par le

processeur. Ces transitions provoquent le changement de l'état de plusieurs ressources du processeur. Ce qui implique une grande augmentation de l'énergie consommée. Utiliser uniquement le coût de base des instructions, pour modéliser la consommation de puissance d'une séquence d'instructions, tend à sous-estimer la consommation [66]. Ainsi, l'ajout du coût inter-instructions pour chaque paire d'instructions consécutives mène à une estimation d'énergie beaucoup plus précise. D'autres coûts inter-instructions, normalement moins significatives, sont aussi observés notamment l'effet de *« cache-miss »* et l'effet de blocage du pipeline [67].

Les méthodologies d'estimation d'énergie ILPA mettent l'accent sur les avantages de l'utilisation de différents niveaux d'abstractions. Chaque niveau possède ses propres caractéristiques et ajoute l'information pertinente sur différentes étapes du processus d'estimation de puissance. En outre, les modèles d'estimation basé sur l'instruction abstraient tout ce qui concerne l'implémentation de bas niveau, réduit le nombre de paramètres et facilite l'extraction de données requises. Toutefois, chaque modèle a ses propres particularités. Dans la littérature, il est possible de trouver plusieurs modèles très détaillés qui tiennent compte des diverses sources de consommation d'énergie des processeurs (les effets inter-instructions). En même temps, d'autres modèles sont très simplistes et ignorent des concepts considérés redondants. L'étude [3] propose que les techniques très minutieuses finissent par surestimer la valeur calculée.

Les coûts de base de l'instruction et les coûts inter-instructions sont idéalement obtenus à travers un processus de caractérisation du processeur. En effet, ce processus permet de profiler l'ensemble d'instruction du processeur. Ce dernier exécute chaque instruction ou bien chaque paire d'instructions plusieurs fois pendant un certain laps de temps. Pour chacune de ces exécutions la puissance consommée par le processeur doit être calculée, ensuite les valeurs obtenues sont analysées et un modèle d'énergie peut enfin être généré. Une diversité de travaux [1, 68-70] suit ce flot classique pour obtenir un modèle d'énergie basé sur l'instruction. Ils se différencient principalement par la méthode employée pour simuler le design, par la méthode utilisée pour mesurer la consommation d'énergie du processeur ainsi que par les effets inter-instructions analysés.

Afin d'augmenter le niveau de détails de la modélisation au niveau instruction en vue d'obtenir une meilleure précision, la technique [71] a modélisé l'effet des opérandes des instructions sur la consommation d'énergie d'un processeur de la famille ARM7. Le modèle présenté en [72] est encore plus minutieux et analyse l'effet des paramètres (opérande) ainsi que l'effet du blocage de pipeline dans la consommation du même processeur.

Notamment, la grande problématique de la méthodologie ILPA, soulignée par plusieurs travaux [60, 61, 73], est que la quantité de mesures lors du processus de caractérisation du processeur est directement proportionnelle au nombre d'instructions de ce dernier. Ainsi l'analyse de l'effet inter-instructions de toutes les paires d'instructions devient une tâche extrêmement compliquée à être réalisée due au temps demandé. Pour accélérer le processus de caractérisation, [74] propose de diviser les instructions en différents groupes. Ces deniers regroupent les instructions dont les opérations élémentaires sont relativement similaires. Également, une deuxième approche pour accélérer ce processus est de l'automatiser. Wendt et al. [2] présente un outil capable de caractériser l'ensemble d'instructions d'un processeur ARM de manière automatique sans l'innervation de l'usager.

Il est important de mentionner que ce travail donne suite à un premier projet de recherche sur l'estimation de la consommation de puissance des périphériques au niveau système réalisé dans notre laboratoire [8, 69]. Entre autres, ce travail a contribué au développement d'un modèle de puissance au niveau instruction pour le processeur Microblaze. Bien que les résultats obtenus soient pertinents, la méthodologie appliquée était manuelle et difficile à étendre sur d'autres processeurs.

3.2.1 Les techniques d'estimation d'énergie choisies

Comme il a été mentionné, le même problème fut abordé dernièrement dans plusieurs études avec diverses approches différentes. Cette diversité nous a amenés à choisir trois techniques distinctes d'estimation de puissance basées sur les instructions pour évaluation et comparaison.

D'une manière générale, nous pouvons affirmer que l'énergie totale consommée par un programme E_{total} est la somme de l'énergie consommée à chaque cycle $E_{cycle}(n)$ sur tous les cycles n_{total}.

$$E_{total} = \sum_{n=0}^{n_{total}} E_{cycle}(n) \tag{3.1}$$

Toutefois, l'énergie consommée par cycle peut être définie de différentes manières. Ceci nous amène à une diversité de technique d'estimation.

3.2.1.1 Instruction sans transition

Cette technique d'estimation est très simple. L'énergie par cycle E_{cycle} est seulement basée sur le coût de base de l'instruction, abrégé par CB_i, ceci étant l'énergie consommée par le processeur lors de l'exécution de l'instruction i. Une addition linéaire du coût de base de chaque instruction exécutée donne l'estimation de l'énergie consommée.

$$E_{cycle} = CB_i \tag{3.2}$$

Ainsi,

$$E_{total} = \sum_{n=0}^{n_{total}} CB_i\ (n) \tag{3.3}$$

Cette technique a déjà été utilisée par plusieurs projets de recherches [1], [74], entre autres. Elle est illustrée sur la Figure 3-1.

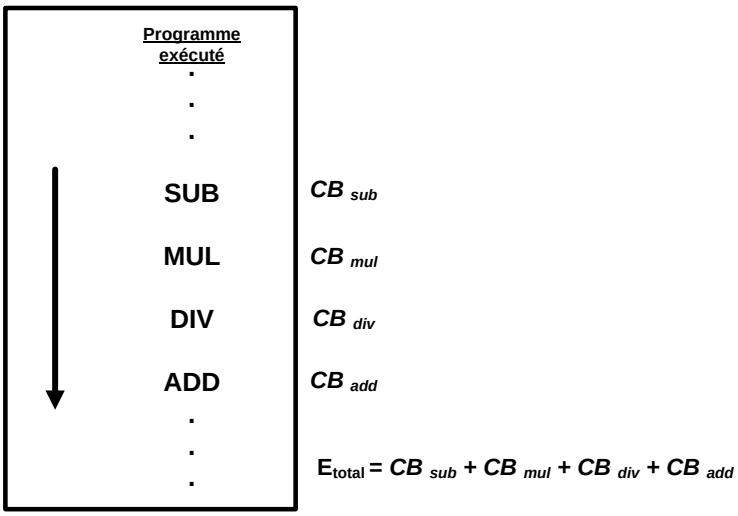

Figure 3-1 Estimation de l'énergie consommée par instruction sans transition

3.2.1.2 Instruction avec transition unique

Cette deuxième technique d'estimation décompose l'énergie par cycle en deux parties. La première partie est le coût de base de l'instruction CB_i. La deuxième partie est le coût de transition, ou bien le coût inter-instructions, abrégé par $CI_{i,j}$ où i est l'instruction exécutée et j est l'instruction qui la suit, ceci correspond à l'énergie consommée lors de la transition entre deux instructions différentes exécutées consécutivement par le processeur. Bref, l'énergie par cycle E_{cycle} est le coût de base de l'instruction CB_i plus le coût inter-instructions $CI_{i,j}$.

La grande difficulté d'une telle approche est la modélisation du coût inter-instructions. Dû à la grande quantité d'instructions dans le jeu d'instruction d'un processeur, la quantité de couples d'instructions possibles est très grande. Pour un processeur avec n instructions, nous avons n^2 coûts inter-instructions possibles. Ainsi, la technique, *instruction avec transition unique*, fait l'approximation suivante :

$$\forall j, \qquad CI_{i,j} = CI_{i,NOP}$$

Quelle que soit l'instruction j, le coût inter-instructions $CI_{i,j}$ sera arrondi par le coût inter-instructions $CI_{i,NOP}$. Donc, pour un processeur avec n instructions dans son jeu d'instruction,

nous aurions seulement n coûts inter-instructions possibles, au lieu de n^2. Par conséquent, l'énergie par cycle E_{cycle} est le coût de base de l'instruction, CB_i, plus le coût inter-instructions entre l'instruction i et l'instruction NOP, $CI_{i,NOP}$.

$$E_{cycle} = CB_i + CI_{i,NOP} \tag{3.5}$$

Finalement,

$$E_{total} = \sum_{n=0}^{n_{total}} CB_i(n) + CI_{i,NOP}(n) \tag{3.6}$$

Cette technique est aussi connue comme le modèle NOP [70]. En effet, cette approche se base sur l'hypothèse que l'effet inter-instructions est principalement dû au changement des instructions et non aux instructions qui y sont impliquées. Cette hypothèse nous mène vers une modélisation de la consommation de puissance au niveau instruction qui ne dépend pas de toutes les paires d'instructions du processeur. Les effets inter-instructions sont pris en compte, sans toutefois trop augmenter la complexité du modèle. La figure suivante, Figure 3-2, illustre l'utilisation de cette technique.

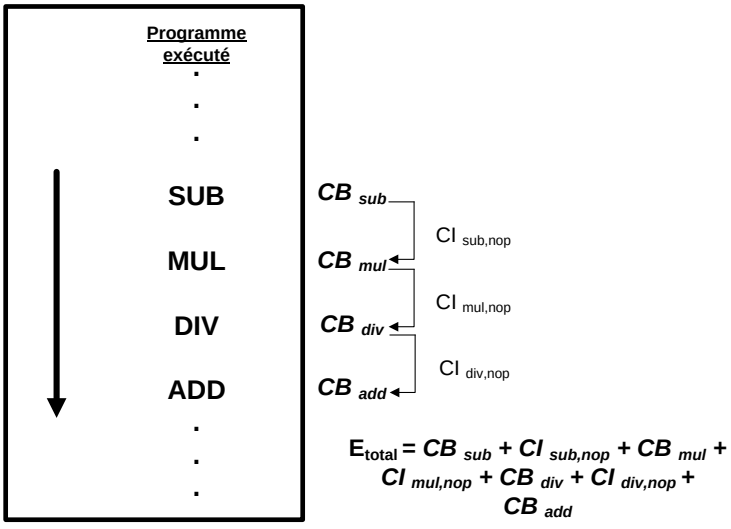

Figure 3-2 Estimation de l'énergie consommée par instruction avec transition unique

Les projets [8, 70] ont également utilisé une approche très similaire à celle-ci.

3.2.1.3 Instruction avec transition

La troisième et dernière technique d'estimation que nous allons étudier dans ce travail est l'*instruction avec transition*. Cette dernière est très semblable à la dernière technique présentée. L'énergie par cycle est, encore une fois, décomposée en deux parties. La première partie est le coût de base de l'instruction CB_i, et la deuxième partie est le coût inter-instructions $CI_{i,j}$. Toutefois, cette technique ne fait aucune approximation du coût inter-instructions. Alors, l'énergie par cycle E_{cycle} est le coût de base de l'instruction CB_i plus le coût inter-instructions $CI_{i,j}$.

$$E_{cycle} = CB_i + CI_{i,j} \tag{3.7}$$

Ceci implique que,

$$E_{total} = \sum_{n=0}^{n_{total}} CB_i(n) + CI_{i,j}(n) \tag{3.8}$$

La Figure 3-3 illustre l'utilisation de cette technique. Elle a déjà été employée par [2], [72] entres autres.

Sans aucune approximation, tous les coûts de transition de l'ensemble des instructions du processeur doivent être calculés. Ceci est une tâche très coûteuse en temps où il y a un très grand risque d'erreurs, c'est pourquoi un processus de modélisation automatique s'avère absolument indispensable. En raison de cette nécessité, nous avons développé un processus automatique de modélisation de la consommation de puissance des processeurs configurables de type *« softcore »*. Ce processus génère un modèle d'énergie qui contient tous les coûts de base et tous les coûts inter-instructions du processeur. Ainsi, ce même modèle d'énergie permet l'application des trois techniques d'estimation présentées. Le Chapitre 4 décrit en profondeur la méthodologie employée pour modéliser la consommation de puissance. Par la suite, au Chapitre 5, nous allons nous servir du modèle d'énergie obtenue pour évaluer et comparer ces trois techniques.

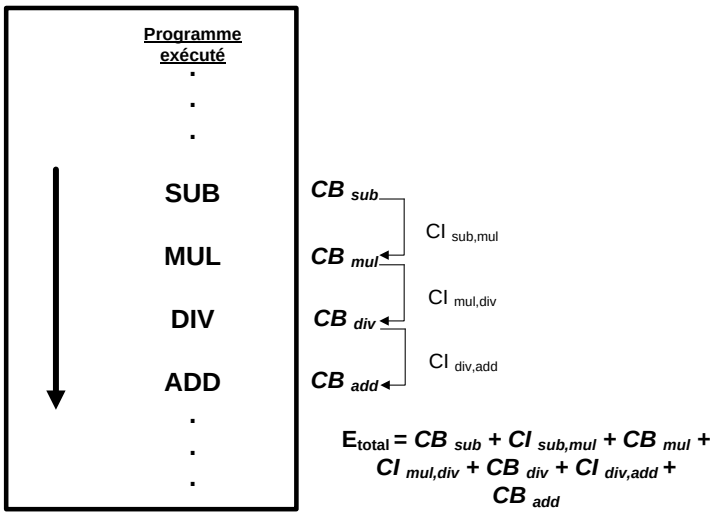

Figure 3-3 Estimation de l'énergie consommée par instruction avec transition

3.3 Hybride

La méthode hybride combine la méthodologie FLPA et la méthodologie ILPA. Cette méthode sépare les instructions du processeur en plusieurs classes (modélisation ILPA) et la consommation de puissance de chaque classe est décrite par l'activité de blocs fonctionnels (modélisation FLPA) [75]. Actuellement, les processeurs extensibles sont très populaires dans l'industrie, ces processeurs sont un bon compromis entre haute performance et haute flexibilité tout en gardant un temps de développement très court. Les techniques d'estimation de consommation hybrides s'avèrent une bonne approche pour l'estimation de puissance de cette gamme de processeurs. En [76], le processeur extensible Xtensa de Tensilica [77] a été caractérisé à l'aide d'un macro-modèle linéaire hybride. Ce dernier est en effet la sommation de deux fonctions linéaires. La première est une fonction linéaire dépendante des paramètres au niveau instructions, celle-ci modélise les instructions de base du processeur Xtensa ainsi que les effets secondaires que le matériel personnalisé peut causer, déjà la deuxième est une fonction linéaire dépendante des paramètres au niveau fonctionnel, celle-ci modélise spécifiquement la consommation du matériel personnalisé.

CHAPITRE 4 MÉTHODOLOGIE POUR UNE CATACTÉRISATION AUTOMATISÉE

Ce chapitre explique les aspects déterminants de ce mémoire ainsi que les détails d'implémentation de l'outil qui génère automatiquement le modèle d'énergie des processeurs configurables de type *« soft-processors »*. La première partie décrit les processeurs que nous avons choisis de caractériser. La deuxième partie contient une vue d'ensemble de la méthodologie de caractérisation appliquée par le flot de profilage. De plus, la complexité de la méthodologie et les outils impliqués sont énumérés. Finalement, nous présentons la structure du flot développé ainsi que les choix qui ont mené à la version finale de ce dernier.

4.1 Les processeurs

La méthodologie présentée est applicable dans la caractérisation de diverses catégories de processeurs. Dans notre cas, on se restreint aux processeurs configurables de type *« soft-processors »* synthétisables sur les FPGAs de la compagnie Xilinx. Ce choix est dû uniquement au fait que les anciens travaux de recherches de notre laboratoire portaient déjà sur les FPGA de Xilinx. Nous avons décidé de continuer avec la même technologie. Ainsi, ce projet a profilé deux processeurs différents. Le premier fut le processeur Microblaze de Xilinx[4]. Ceci est un processeur RISC *« Reduced Instruction Set Computer »* optimisé pour être implémenté sur les cartes FPGA de Xilinx. Ce processeur fait partir de l'environnement de développement de systèmes embarqués EDK *« Embedded Development Kit »* [78]. Le deuxième processeur profilé est le LEON3 [79] de la famille d'architecture SPARCv8 *« Scalable Processeur ARChitecture version 8 »*. Ce dernier est un processeur décrit en VHDL synthétisable. Son code source est disponible sur la licence GNU GPL *« General Public Lincense »*, ce qui permet une utilisation illimitée pour la recherche sans aucune charge.

4.1.1 Microblaze

Ce processeur ayant été utilisé précédemment dans un autre projet [8, 69] au même laboratoire de recherche, nous l'avons considéré puisque cela nous permettra d'analyser les améliorations du présent travail par rapport au précédent, en comparant les méthodologies appliquées et les résultats obtenus.

La consommation de puissance du Microblaze a aussi été étudiée en [1], ce qui nous servira comme deuxième source de comparaison lors de l'analyse de résultats.

Le Microblaze est un processeur hautement configurable. Il y a des attributs fixes et aussi un ensemble d'attributs qui peuvent être activés à la guise de l'utilisateur.

L'ensemble d'attributs fixes est le suivant :

- Registre de trente-deux (32) bits à des fins générales
- Instructions sur 32 bits avec 3 opérandes et 2 modes d'adressage
- Bus d'adresse de 32 bits

Dans notre cas, nous avons ajouté plusieurs fonctionnalités au processeur, les principales sont les suivantes :

- Pipeline de 5 étages
- Dispositif de décalage
- Multiplicateur 32 bits
- Diviseur 32 bits
- Unité de calcul à point flottant « FPU »

L'unité point flottante a été ajoutée dans la configuration du Microblaze, mais cette unité ne sera pas utilisée dans nos tests puisque les instructions qui utilisent la FPU ne seront pas caractérisées. La raison pour laquelle nous avons ajouté cette unité dans le processeur est que nous avons observé une augmentation de la consommation d'énergie du processeur lorsque ce dernier possédait une FPU. En effet, nous avons tour à tour exécuté un même code logiciel sur un processeur sans FPU et avec FPU. Puis nous avons comparé les résultats. Le code exécuté avec FPU a eu une consommation d'énergie plus grande (Figure 4-1). Cette différence s'explique par le fait que même lorsque la FPU n'est pas utilisée, il y a une activité dans cette dernière ce qui augmente la consommation de puissance. Ceci est dû à l'implémentation du processeur. En vue d'obtenir les pires cas lors de notre estimation, nous avons décidé de garder l'unité de calcul à point flottant, même sans l'utiliser.

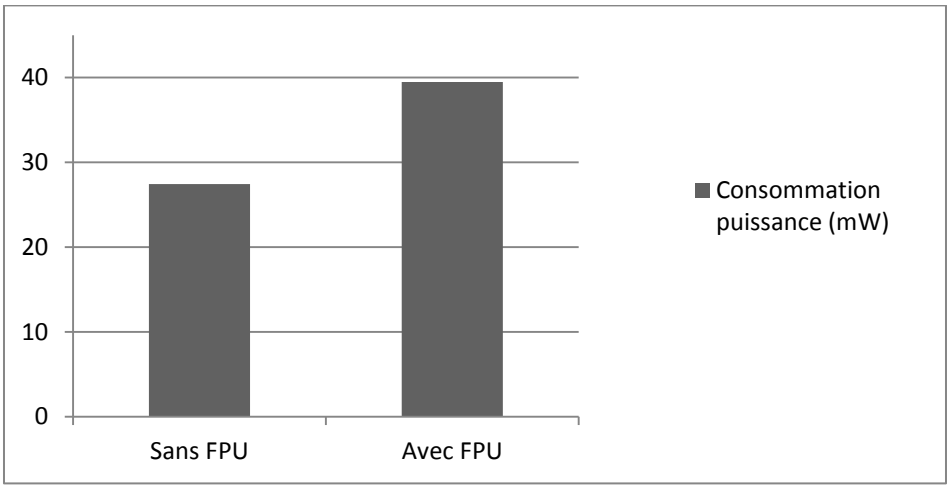

Figure 4-1 Comparaison entre la consommation du Microblaze avec FPU et sans FPU

Notez également que ce processeur ne comporte pas de mémoire cache.

4.1.2 LEON3

Le processeur LEON est largement utilisé dans l'industrie, principalement dans le domaine de l'Aérospatiale pour le développement de satellites. Nous l'avons sélectionné étant donné sa popularité, tant dans le milieu de la recherche que celui de l'industrie.

La famille de processeurs LEON est fournie et supportée par la compagnie Aeroflex. Tout comme le Microblaze, ce processeur est hautement configurable et particulièrement adéquat pour la conception de systèmes à base de FPGAs. Le LEON3 est une version améliorée du LEON2. De plus, il existe des versions tolérantes aux pannes pour le LEON2 , LEON3 et même le LEON4.

La configuration du processeur LEON3 employé dans ce projet est la suivante :

- Instructions sur 32 bits qui peuvent être divisées en six catégories
- 8 fenêtres de registre
- Pipeline de 7 étages
- Multiplicateur 32 bits
- Diviseur 32 bits
- Instruction MAC (de l'anglais *« Multiplier and Accumulator »*)

Le même comportement que celui du processeur Microblaze a été observé lorsqu'on ajoute une unité FPU. L'ajout d'une FPU augmente la consommation (Figure 4-2) de la puissance consommée même si cette unité n'est jamais utilisée. Par contre, des difficultés techniques (libraire de simulation manquante), afin d'ajouter une FPU dans notre version du LEON3, nous ont empêchés de faire cette analyse en début de projet. Par conséquent, la version du LEON3 utilisée dans ce projet ne contient pas de FPU.

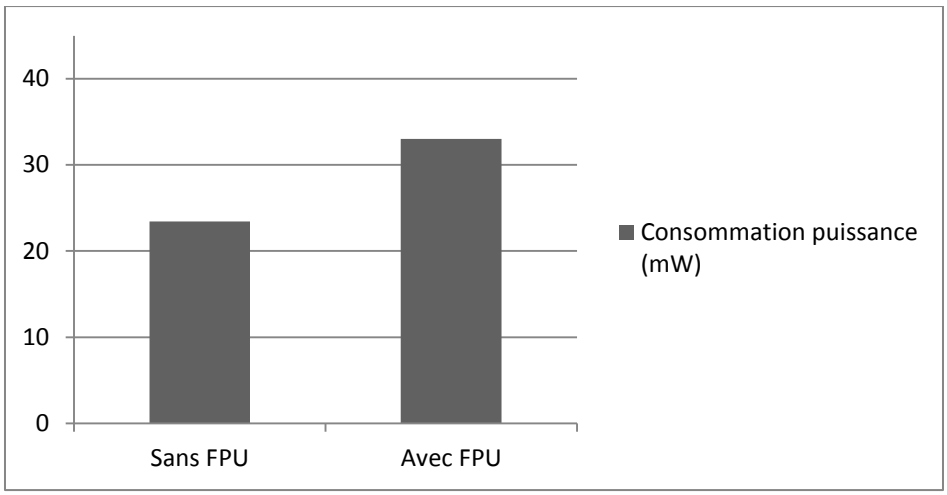

Figure 4-2 Comparaison entre la consommation du LEON3 avec FPU et sans FPU

Ici également, la cache ne fait pas partie de la configuration du processeur.

4.2 Aperçu général

À la section 3.2.1, trois techniques d'estimation de puissance au niveau instruction ont été présentées. On a conclu que les coûts de base CB et les coûts inter-instructions CI sont les valeurs requises par ces techniques pour estimer la puissance à ce niveau d'abstraction. L'obtention de ces valeurs est possible à travers la caractérisation de toutes les instructions et de tous les couples d'instructions. Pour caractériser une instruction en particulier, le processeur doit exécuter cette instruction, répétitivement, un certain nombre de fois. De manière similaire pour un couple d'instructions, le processeur doit exécuter le couple en question répétitivement plusieurs fois. Cette exécution peut se faire par simulation ou bien directement sur une carte FPGA. Dans les deux cas, il faut mettre en place une méthode capable d'estimer la puissance consommée par le processeur lors de l'exécution de la boucle.

La méthodologie appliquée dans ce travail est basée sur la simulation. Elle suit les propositions de plusieurs projets décrits dans la revue de littérature [1], [2], [8], [65], [74] et [72]. Après une analyse approfondie de ces derniers, nous avons été en mesure d'adapter leur méthode à nos

propres besoins. Un aperçu général de la méthodologie employé par ce projet est présenté à la Figure 4-3.

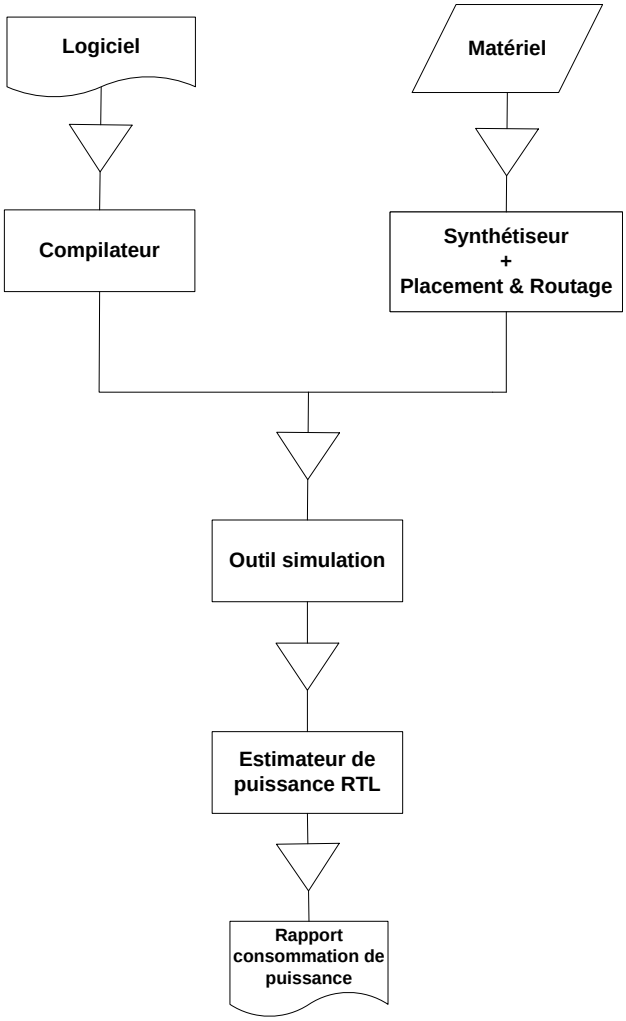

Figure 4-3 Méthodologie appliquée

Le point de départ de la méthodologie est un code logiciel et une spécification du système embarqué à base de FPGA sur lequel s'exécute le processeur, de type « soft-processor », qui doit être profilé. Le code logiciel contient une instruction ou un couple d'instructions dans une boucle. Dans ce qui suit, ce code logiciel sera appelé *scénario*. Ce scénario va faire l'objet d'une compilation croisée (en anglais « *cross compiler* ») à l'aide d'un compilateur ciblant le processeur en question. Parallèlement, une liste d'interconnexions post-placement et routage va

être généré à partir de la spécification du système embarqué. Par la suite, on effectuera une simulation RTL post-placement et routage. À la fin, un outil d'estimation de puissance RTL calculera la consommation d'énergie du processeur lors de l'exécution du scénario donné en entrée. Avec la consommation d'énergie du processeur pour chaque scénario, on est ainsi en mesure de calculer tous les CB et tous les CI.

4.2.1 Effort de caractérisation

Si on applique la méthodologie décrite ci-dessus afin de caractériser le jeu d'instruction d'un processeur, ces étapes doivent être répétées pour chaque scénario. Pour chaque instruction, un scénario est requis afin de calculer CB. Pour chaque couple d'instructions, un scénario est requis afin de calculer CI. Si on suppose que n est le nombre total d'instructions d'un processeur, le nombre exact de scénarios requis est donné par :

$$Nombre\ scénarios = n + \left(n * \left(\frac{n-1}{2}\right)\right) \tag{4.1}$$

Conséquemment, plus grand est le jeu d'instruction d'un processeur plus l'effort est grand pour une caractérisation complète. En ce qui concerne le processeur Microblaze nous avons choisi 102 instructions à caractériser, ce qui nous donne 5253 scénarios. De même, pour le processeur LEON3 nous avons sélectionné 66 instructions ce qui nous donne 2211 scénarios. Les instructions ont été choisies de manière aléatoire dans l'ensemble d'instructions du mode utilisateur de chaque processeur.

$$Nombre\ scénarios\ Microblaze = 102 + \left(102 * \left(\frac{102-1}{2}\right)\right) = 5253 \tag{4.2}$$

$$Nombre\ scénarios\ Leon3 = 66 + \left(66 * \left(\frac{66-1}{2}\right)\right) = 2211 \tag{4.3}$$

Puisqu'une caractérisation manuelle risque d'être laborieuse à réaliser, nous proposons un flot de profilage automatique qui accélérera grandement la caractérisation des processeurs configurables de type *« soft-processors »*.

Notez qu'une approche possible afin de diminuer la quantité de scénarios et ainsi minimiser l'effort pourrait être de diviser les instructions par catégories : on forme un groupe pour les instructions ADD, un second pour les instructions OR et ainsi de suite. Ainsi, un seul scénario est requis par groupe. Évidemment, cette pratique diminue la qualité des résultats et pour cette raison, nous ne l'avons pas adopté. Lors de notre projet, nous n'avons pas quantifié l'effet des groupements. Toutefois, en [3], l'effet de cette technique a été étudié sur le processeur Microblaze. Il a été démontré que l'erreur due au groupement peut atteindre les 15 %.

4.2.2 Les outils

Du côté matériel, nous utilisions les outils fournis par Xilinx, soit les logiciels en ligne de commande qui viennent avec ISE Design Suite 13.4 [80], pour la génération de la liste d'interconnexions post-placement et routage. La simulation est exécutée avec Modelsim 6.6f de Mentor Graphics [5]. Il existe plusieurs outils commerciaux de simulation pour les designs FPGA, toutefois nous avons décidé de poursuivre avec les outils utilisés dans le travail précédent. Puis finalement, nous utilisions l'estimateur de puissance RTL XPower Analyzer de Xilinx [6]. Cet outil analyse la liste d'interconnexions post-placement et routage et le fichier de sortie de la simulation contenant l'activité de tous les signaux du circuit pour estimer la consommation de puissance globale. Donc, la qualité de ces deux fichiers est très importante pour la précision de l'estimation. Le résultat donné par l'outil, sous forme d'un fichier texte, est très détaillé. Il fournit séparément la consommation de chaque périphérique du système embarqué. À la Figure 4-4, nous pouvons voir un exemple du rapport généré par cet outil. Dans ce cas précis, nous nous sommes intéressés à la consommation totale du processeur.

Il est important de noter que XPower joue un rôle très important dans la précision de la caractérisation, parce que toutes les estimations qui nous servent de point de départ pour le calcul de CB et CI, sont réalisées avec cet outil. La précision de ce logiciel a été souvent discutée dans la littérature. Quelques travaux ont déjà mis en question la qualité des résultats [24, 40]. Par

contre, d'autres travaux plus récents montrent une amélioration dans les résultats [3] et [81]. En [3], Viktor K. Prasanna affirme que l'erreur moyenne de XPower est de seulement 6.8%.

3. Details
3.1. By Hierarchy

By Hierarchy	Power (mW)	Logic Power (mW)	Signal Power (mW)
Hierarchy total	115.34	88.96	26.38
LEON3_VP_top	2.84 / 115.34	0.00 / 88.96	2.84 / 26.38
prom	0.49	0.10	0.39
leon31_wrapper_inst	0.00 / 21.79	0.00 / 7.50	0.00 / 14.30
LEON3S_instance	0.23 / 21.79	0.00 / 7.50	0.23 / 14.30
rf0	0.00 / 3.92	0.00 / 3.75	0.00 / 0.17
s1.dp.x1	1.30	1.23	0.07
s1.dp.x0	2.62	2.52	0.10
p0	2.48 / 17.64	0.01 / 3.75	2.47 / 13.90
mgen.mul0	0.00 / 0.35	0.00 / 0.21	0.00 / 0.15
xm3232.m3232	0.00 / 0.35	0.00 / 0.21	0.00 / 0.15
pipe2.arch0.dwm	0.35	0.21	0.15
mgen.div0	0.22	0.04	0.18
iu0	7.90	1.83	6.07
c0mmu	0.29 / 6.70	0.00 / 1.67	0.29 / 5.03
icache0	1.14	0.33	0.81
dcache0	3.14	0.89	2.25
a0	2.13	0.45	1.68

Figure 4-4 Rapport de consommation de puissance généré par XPower

4.3 L'outil *Power Profiler*

Cette section présente plus en détail les éléments de l'outil de caractérisation développé dans le cadre de ce projet et introduit à la section 4.2. Nous allons regarder plus en détail les différentes raisons qui ont motivé nos choix et les différentes avenues possibles. Tel qu'illustre à Figure 4-5, l'outil *Power Profiler* consiste en un ensemble de modules indépendants contrôlés par un module principal, le noyau. Ce dernier a la responsabilité de gérer chaque étape du processus de caractérisation.

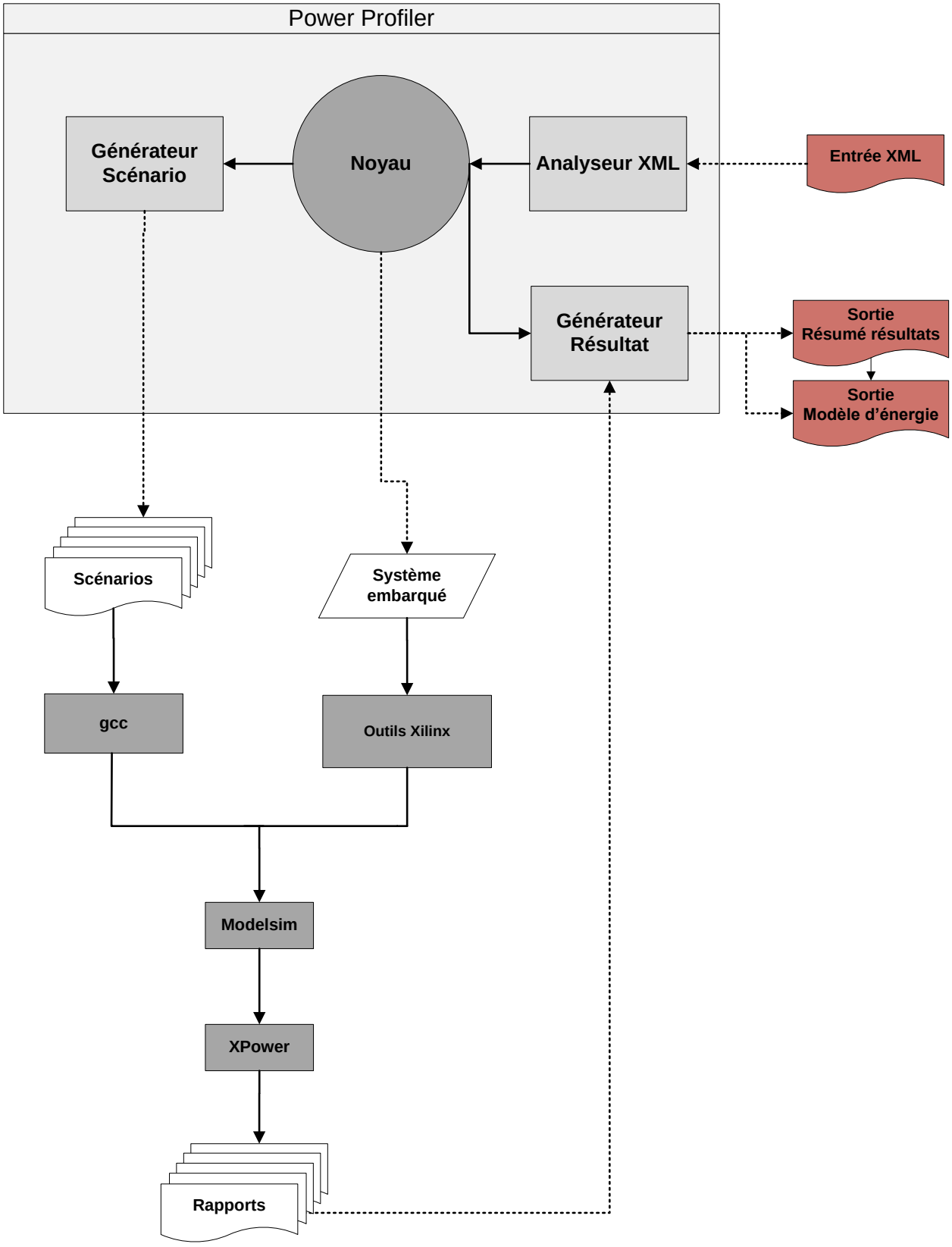

Figure 4-5 Vue d'ensemble de l'outil de caractérisation

4.3.1 Analyseur XML

Comme nous pouvons le remarquer, le point de départ du flot de profilage est un fichier XML *« Extensible Markup Language »*, nous avons choisi ce format de fichier à cause de sa simplicité et facilité de manipulation. En vue d'éliminer toute intervention de l'usager dans l'outil, nous avons créé ce fichier d'entrée comportant une structure basée sur le langage XML spécifique aux besoins du flot. Ce fichier contient toute l'information nécessaire au bon fonctionnement de l'outil et lui permet d'être autonome. Le fichier se divise en deux parties. La première partie comprend l'ensemble de paramètres modifiables. Étant donné que la portabilité était un requis que nous avons établi au début du projet, l'outil ne devrait pas être limité à un FPGA spécifique, ou bien à une famille de FPGA. Donc, la première partie du fichier permet à l'usager de cibler le FPGA de son choix et aussi de définir la fréquence à laquelle le processeur fonctionnera. C'est aussi à ce moment que l'usager indique le processeur à être caractérisé.

```
<systemTopLevel>system</systemTopLevel>
<processorInstance>processor_microblaze</processorInstance>
<architectureFPGA>virtex5</architectureFPGA>
<device>xc5vlx50t</device>
<speed>-1</speed>
<package>ff1156</package>
<processor>microblaze</processor>
<frequency>100</frequency>
```

Figure 4-6 Les paramètres modifiables dans le fichier d'entrée

Dans la deuxième partie du fichier, l'usager doit rentrer les informations propres au processeur choisi, soit les registres d'usage général et les instructions à caractériser. Comme le nom suggère, ces registres ne sont pas réservés à un usage en particulier, ils agissent comme une mémoire qui peut garder divers types de données temporairement. Ces registres sont très importants lors de la génération de scénario, ceci est expliqué à la section 4.3.3 de ce chapitre. La Figure 4-7 montre comment les registres sont définis dans le fichier d'entrée.

```
<registers type="General purpose registers">
        <register>r3</register>
        <register>r4</register>
                ...
        <register>r30</register>
        <register>r31</register>
</registers>
```

Figure 4-7 Les registres du processeur dans le fichier d'entrée

Pour terminer, l'usager doit indiquer les instructions appartenant au jeu d'instruction du processeur qu'il veut caractériser. Bien sûr, le but est d'avoir tout le jeu d'instruction du processeur, ceci nous permettre de générer un modèle d'énergie exhaustif. La Figure 4-8 montre comment chaque instruction doit être représentée dans le fichier, c'est-à-dire par un nœud dans le fichier XML. Ce nœud contient toujours deux attributs, la mnémonique de l'instruction et le groupe de l'instruction, par exemple arithmétique, logique, branche entre autres. Ces informations sont essentielles au module générateur de scénarios. Le nœud instruction doit toujours avoir trois nœuds fils identifiant les deux opérandes sources et l'opérande destination de l'instruction. L'opérande source numéro un $< operandOne >$ peut prendre les valeurs ra ou $none$. L'opérande source numéro deux $< operandTwo >$ peut prendre les valeurs rb, IMM ou $none$. Et, l'opérande destination $< destination >$ peut prendre les valeurs rd ou $none$. ra, rb ou rd indiquent que l'opérande est un registre, alors que IMM indique que l'opérande est une valeur immédiate. Dans ce dernier cas, le nœud accepte un attribut de type $xbit$ qui indique sur combien de bits est la valeur immédiate. Cette structure de données en XML permet une interprétation rapide du format assembleur de chaque instruction du processeur en question.

```
<instruction mnemonic = "add" group = "arithmetic">
        <operandOne>ra</operandOne>
        <operandTwo>rb</operandTwo>
        <destination>rd</destination>
</instruction>
<instruction mnemonic = "bsrli" group = "logic">
        <operandOne>ra</operandOne>
        <operandTwo xbit = "5">imm</operandTwo>
        <destination>rd</destination>
</instruction>
<instruction mnemonic = "brd" group = "branchDelay">
        <operandOne>none</operandOne>
        <operandTwo>rb</operandTwo>
        <destination>none</destination>
</instruction>
<instruction mnemonic = "beq" group = "branch">
        <operandOne>ra</operandOne>
        <operandTwo>rb</operandTwo>
        <destination>none</destination>
</instruction>
```

Figure 4-8 Les instructions à caractériser dans le fichier d'entrée

À partir de l'exemple illustré à la Figure 4-8, il est facile d'interpréter le format des instructions en langage assembleur :

Instruction ADD	add rD, rA, rB
Instruction BSRLI	bsrli rD, rA, IMM
Instruction BRD	brd rB
Instruction BEQ	beq rA, rB

Ce format XML de description du jeu d'instruction est suffisant pour décrire des architectures de processeurs RISC typiques telles que le Microblaze et LEON3.

Finalement, une fois le fichier d'entrée créé, le module analyseur XML est responsable de sa lecture. Ce module est complètement indépendant des autres modules du flot. Ceci laisse la possibilité de modifier le format d'entrée, soit le format XML, sans toutefois, avoir besoin d'apporter des modifications majeures aux autres modules.

4.3.2 Plate-forme matérielle

Comme nous pouvons observer sur la Figure 4-5, le flot a besoin d'une plate-forme matérielle sur laquelle les scénarios seront exécutés. Pour l'instant, l'outil n'a pas la capacité de générer ce système embarqué automatiquement. Néanmoins, l'outil est capable de modifier les spécifications d'un système embarqué déjà existant. Ainsi, il faut définir, auparavant, une plate-forme matérielle de base sur laquelle se trouve le processeur à être caractérisé. Une fois le fichier XML lu, le module noyau appliquera les modifications nécessaires à cette plate-forme pour respecter les paramètres déterminé dans le fichier d'entrée. Ces modifications sont faites en mode automatique sans aucune intervention de l'usager.

La plate-forme matérielle de base utilisée dans ce projet comprend un processeur, une mémoire RAM, le bus et un périphérique développé par nous, nommé *stop_simulation*. Un aperçu général de la plate-forme matérielle de chaque processeur est présenté sur la Figure 4-9 et sur la Figure 4-10.

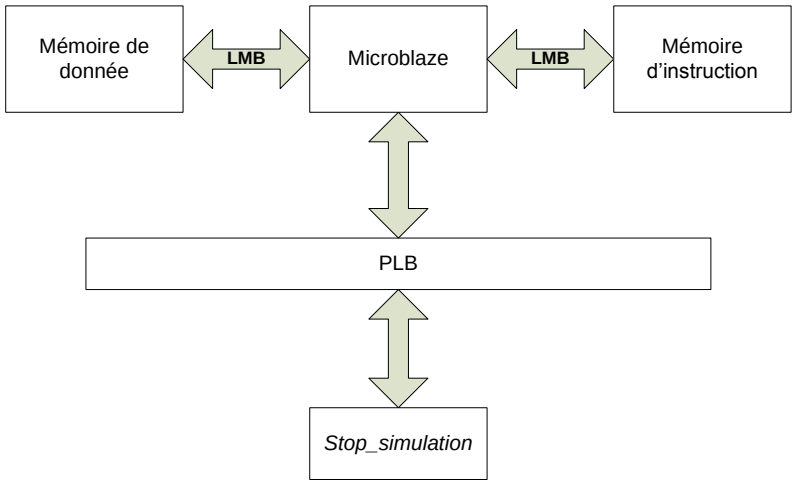

Figure 4-9 Plate-forme matérielle Microblaze

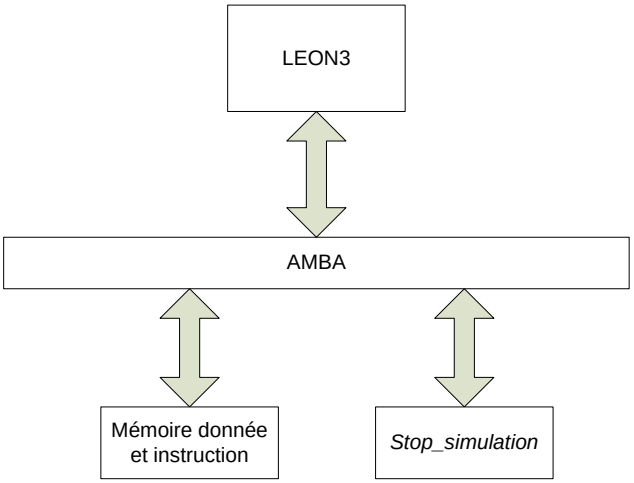

Figure 4-10 Plate-forme matérielle LEON3

Le *stop_simulation* est très important pour la précision des résultats, toutefois son fonctionnement est très simple. Il reçoit des valeurs à travers le bus et il les sauvegarde dans un registre interne. Lors d'une simulation, à travers un script DO (fichier de script Modelsim) l'outil peut observer les valeurs reçues par ce périphérique, et ainsi selon cette valeur, connaître le moment exact pour débuter et arrêter l'enregistrement de l'activité du circuit. Ceci nous permet d'obtenir l'estimation de puissance consommée par le processeur sur un intervalle de temps très précis.

4.3.3 Générateur de scénarios

Après avoir interprété le fichier XML d'entrée et modifié la spécification de la plate-forme matérielle, le bloc *Power Profiler* (Figure 4-5) générera les scénarios nécessaires au profilage des instructions du processeur. Le module générateur de scénarios est responsable de cette tâche.

Le scénario est en soi un fichier écrit en langage C avec plusieurs instructions en assembleur. Il est divisé en deux sections distinctes, « *setup* » et « *run* ». La section « *setup* » est responsable pour préparer le processeur à l'exécution de toutes les instructions spécifiées dans le fichier d'entrée. Ceci est fait par l'initialisation des registres d'usage général et l'initialisation de la mémoire de données. D'autre part, la section « *run* » est composée d'une même instruction ou d'un même couple d'instructions qui se répète plusieurs fois. Nous avons opté pour répéter l'instruction ou le couple d'instructions en question au lieu d'utiliser une boucle, parce que l'utilisation de cette dernière ajouterait un coût additionnel non négligeable en énergie lors de l'exécution du scénario.

Chaque famille de processeurs correspond à des fonctionnalités diverses et à des capacités différentes, tel que chacune a son propre format de donnée, son propre mode d'adressage et son propre jeu d'instruction. Ceci implique que chaque processeur ait une syntaxe unique du langage assembleur. Ainsi, le module générateur de scénarios doit bien gérer le format assembleur des instructions de chaque processeur ciblé par l'outil. Ce qui explique l'importance de la structure décrivant chaque instruction dans le fichier d'entrée XML. Les informations de chaque nœud XML et ses attributs vont être employés par ce module lors de la génération de scénarios. L'implémentation de ce dernier est, donc, très spécifique à la famille du processeur. Au **Error! Reference source not found.**, cette problématique est illustrée par un exemple très simple où pour la même opération la syntaxe du langage assembleur est très différente. Dans le cas de l'addition, l'ordre des paramètres passés à l'instruction n'est pas le même, de plus, le deuxième paramètre du processeur LEON3 peut être aussi bien une valeur immédiate qu'un numéro de registre. De plus, pour l'instruction *branche si égal*, la syntaxe assembleur du processeur Microblaze requiert le numéro de deux registres, alors que la syntaxe du processeur LEON3 demande seulement l'adresse à brancher.

Tableau 4-1 Comparaison de la syntaxe du langage assembleur des processeurs

	Microblaze		**LEON3**	
	Syntaxe assembleur	*Sémantique*	*Syntaxe assembleur*	*Sémantique*
addition	***add*** rd, ra, rb	rd **:=** ra+ rb	***add*** ra, rb_IMM, rd	rd **:=** ra + rb ou rd **:=** ra + IMM
branche si égal	***beq*** ra, rb	pc **:=** pc + rb **si ra =0**	***be*** label	pc **:=** label **si *icc[2] = 1***

La Figure 4-11 illustre un exemple de la section « *setup* » pour le processeur Microblaze. Il est à noter que cette section est identique dans tous les scénarios. Toutefois, dans le cas de différents processeurs cette section changera, dû à la syntaxe du langage assembleur qui diffère, mais aussi dû aux subtilités de chaque processeur qui oblige une initialisation particulière.

La Figure 4-13 est un exemple de la section « *run* » d'un scénario ciblant le processeur Microblaze avec le couple d'instruction *add* et *bsrli*. Il est possible d'observer que le registre de destination est toujours le même. Cette approche évite les exceptions lors de l'exécution d'une instruction due à une mauvaise valeur d'opérande source. En gardant le même registre destination, les valeurs des autres registres ne changeront pas au cours de l'exécution d'un scénario. Les registres utilisés comme opérandes sources garderont la même valeur reçue lors de leur initialisation. D'autre part, les opérandes sources, soit un registre ou bien une valeur immédiate, ne sont pas toujours les mêmes. Dans la majorité de cas, ils sont choisis aléatoirement et ne sont jamais égaux dans deux instructions qui se suivent. En effet, la variation de la valeur d'entrée de l'instruction est une problématique bien connue dans la littérature, et sera abordée à la fin de ce chapitre.

Finalement, un dernier aspect très important de la section « *run* » est le nombre de fois que le couple d'instructions se répète. Étant donné une instruction A et une instruction B, le couple AB doit être répété N fois. Lorsque A est égale à B, la même instruction sera répétée 2N fois. La

valeur de N doit être suffisamment grande pour que l'estimation soit précise, et suffisamment petite pour que la simulation ne soit pas très longue. Donc, il faut déterminer un bon compromis entre la précision et le temps de simulation requis.

```
asm("SETUP:");
 asm("addi r3,r0,0x2aaaaaaa");
 asm("swi r3,r1,0x8"); // Initialization of memory at stack pointer + 8
 asm("addi r3,r0,0xd5555556");
 asm("swi r3,r1,0x4"); // Initialization of memory at stack pointer + 4
 asm("addi r3,r0,0x0"); // First register = Destination register
 asm("addi r4,r0,0x4"); // Second register = 0x04
 asm("addi r5,r0,0x8"); // Third register =  0x08
 asm("addi r7,r0,0x0");
 asm("addi r8,r0,0x1f");
 asm("addi r9,r0,0x3f");
 asm("addi r10,r0,0x7f");
 asm("addi r11,r0,0xff");
 asm("addi r12,r0,0x1ff");
 asm("addi r19,r0,0x3ff");
 asm("addi r20,r0,0x7ff");
 asm("addi r21,r0,0xfff");
 asm("addi r22,r0,0xffffe001");
 asm("addi r23,r0,0x3ffd");
 asm("addi r24,r0,0xffff8005");
 asm("addi r25,r0,0xfff5");
 asm("addi r26,r0,0xfffe0015");
 asm("addi r27,r0,0x3ffd5");
 asm("addi r28,r0,0xfff80055");
 asm("addi r29,r0,0xfff55");
 asm("addi r30,r0,0xffe00155");
 asm("brlid r31, 0x4"); // Last register = Program Counter
 asm("nop");
```

Figure 4-11 Section *« setup »* d'un scénario du processeur Microblaze

Afin d'assurer une valeur optimale de N, une comparaison de l'énergie consommée par l'instruction « *add* » a été faite en utilisant le même scénario avec quatre valeurs différentes de N. Nous avons fait l'hypothèse que l'énergie consommée par l'instruction « *add* » tend vers sa valeur exacte lorsque N augmente.

Le Tableau 4-2 illustre les résultats obtenus pour le processeur LEON3. La première colonne indique la valeur de N. La colonne 2 correspond au temps requis pour simuler (avec Modelsim) et estimer (avec XPower) la puissance consommée par le processeur. Finalement, la colonne 3 représente l'énergie consommée par l'instruction « *add* ». À partir de ces données, nous avons pu obtenir la courbe sur la Figure 4-12. Nous observons que la courbe tend vers une asymptote verticale ce qui signifie qu'à partir d'un certain moment il est plus difficile d'augmenter la précision de l'estimation en seulement augmentant la valeur de N. Basé sur cette analyse, nous avons choisi d'avoir N = 300 pour le processeur LEON3 afin de garder les temps de simulation et estimation requis raisonnables. Avec la même méthode d'analyse, nous avons choisi N = 500 pour le processeur Microblaze. Il est important de remarquer que le comportement de l'énergie en fonction de N a été analysé seulement pour l'instruction « *add* », nous n'avons pas fait cette analyse sur tout le jeu d'instruction des processeurs.

Tableau 4-2 Comparaison entre l'estimation de l'énergie avec différents N

N	Temps requis	Énergie consommée (nJ)
200	2 min 24 sec	1.2707
300	3 min 10 sec	1.2665
500	4 min 41 sec	1.2591
700	7min 21 sec	1.2542
1000	11min 55 sec	1.2505
1200	19min 55 sec	1.2498

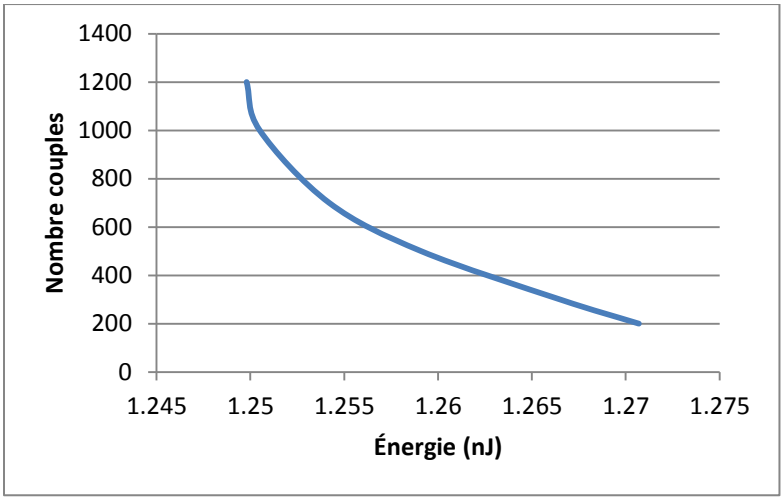

Figure 4-12 Énergie consommée par un même scénario avec différents N

```
asm("RUN:");
  asm("_INST1:");  asm("add r3, r30, r12");
  asm("_INST2:");  asm("add r3, r23, r29");
  asm("_INST3:");  asm("add r3, r27, r26");
  asm("_INST4:");  asm("add r3, r22, r4");
  asm("_INST5:");  asm("add r3, r29, r30");
  asm("_INST6:");  asm("add r3, r24, r9");
  asm("_INST7:");  asm("add r3, r23, r10");
  asm("_INST8:");  asm("add r3, r11, r12");
  asm("_INST9:");  asm("add r3, r23, r24");
  asm("_INST10:");  asm("add r3, r28, r28");
  asm("_INST11:");  asm("add r3, r19, r19");
  asm("_INST12:");  asm("add r3, r27, r10");
  asm("_INST13:");  asm("add r3, r29, r23");
  asm("_INST14:");  asm("add r3, r22, r9");
  asm("_INST15:");  asm("add r3, r30, r12");
  asm("_INST16:");  asm("add r3, r12, r22");
                          ...
  asm("_INST999:");  asm("add r3, r22, r6");
  asm("_INST1000:");  asm("add r3, r22, r26");
```

Figure 4-13 Section « *run* » d'un scénario du processeur Microblaze

4.3.4 Générateur de résultat

Ce module est le dernier du flot de caractérisation de la Figure 4-5. Il est responsable de la réception et de la gestion du résultat de chaque scénario. Ce résultat est un rapport de consommation de la plateforme matérielle (Figure 4-4). À partir de ces rapports de consommation, le module générateur produit un résumé de résultats de chaque scénario, et par la suite, il génère le modèle d'énergie du processeur.

Le Tableau 4-3 présente un exemple de rapports de consommation produit par l'outil pour le processeur Microblaze. La première colonne représente le nom du scénario, alors que la deuxième colonne correspond au temps d'exécution du scénario. La troisième colonne est la puissance consommée par le processeur lors de l'exécution du scénario. La quatrième colonne équivaut à l'énergie totale consommée par le scénario, incluant la section « *setup* » et « *run* ». Les colonnes 5 et 6 procurent respectivement l'énergie consommée et le nombre de couples d'instructions, uniquement pour la section « *run* » du scénario. Finalement, la dernière colonne correspond au couple du scénario en question.

Après avoir parcouru tous les scénarios, le module est en mesure de générer le modèle d'énergie. Ce modèle est une matrice carrée à deux dimensions. Si on considère l'ensemble I_n comme étant l'ensemble de *n* instructions à caractériser, la matrice carrée E_{nxn} correspondra au modèle d'énergie, où :

- $e_{ixj} = Coût\ de\ base\ (CB) de\ l'intruction\ i,\ si\ i = j$
- $e_{ixj} = Coût\ inter - intructions\ (CI)\ entre\ \ i\ et\ j,\ si\ i > j$
- $e_{ixj} = 0,\ si\ i < j$

Tableau 4-3 Résumé de rapports de consommation du processeur Microblaze

	Run(ns)	Power(mW)	Total Energy(nJ)	Couple Energy(nJ)	Number Couples	Couple
scenario_setup	3119.08	30.21	94.2273			
scenario_1	13119.1	63.02	826.764	732.537	500	add/add
scenario_2	13119.1	67.92	891.048	796.82	500	add/rsub
scenario_3	13119.1	63.79	836.866	742.639	500	add/addc
scenario_4	13119.1	68.3	896.033	801.806	500	add/rsubc
scenario_5	13119.1	62.66	822.041	727.814	500	add/addk
…	…	…	…	…	…	…
scenario_2734	18119.1	57.46	1041.12	946.8927	500	and/braid
scenario_2735	18119.1	54.58	988.939	894.7117	500	and/bralid
scenario_2736	14339.1	61.56	882.714	788.4867	500	and/beqi
scenario_2737	22259.1	48.72	1084.46	990.2327	500	and/bnei
scenario_2738	15559.1	59.08	919.23	825.0027	500	and/blti
…	…	…	…	…	…	…
scenario_5250	13119.1	52.23	685.209	590.9817	500	shi/nop
scenario_5251	13119.1	21.39	280.617	186.3897	500	swi/swi
scenario_5252	13119.1	52.24	685.341	591.1137	500	swi/nop
scenario_5253	13119.1	9.25	121.351	27.1237	500	nop/nop

À titre d'exemple la Figure 4-14 présente un modèle d'énergie pour l'ensemble des instructions $I_4 = [add, sub, mul, div]$.

	add	**sub**	**mul**	**div**
add	CB_{add}	$CI_{add\text{-}sub}$	$CI_{add\text{-}mul}$	$CI_{add\text{-}div}$
sub	0	CB_{sub}	$CI_{sub\text{-}mul}$	$CI_{sub\text{-}div}$
mul	0	0	CB_{mul}	$CI_{mul\text{-}div}$
div	0	0	0	CB_{div}

Figure 4-14 Exemple de modèle d'énergie

Le CB de chaque instruction est l'énergie consommée par la section « *run* » du scénario divisée par le nombre de fois que l'instruction est exécutée :

$$CB = \frac{\text{É}nergie\ couple}{2 * Nombre\ couples} \tag{4.4}$$

Le CI_{ij} est l'énergie consommée par la section « *run* » du scénario moins l'énergie consommée par chaque instruction individuellement, le tout divisé par le nombre de transitions effectuées.

$$CI_{i,j} = \frac{\text{É}nergie\ couple - \ Nombre\ couples * (CB_i + CB_j)}{(2 * Nombre\ couples) - 1} \tag{4.5}$$

4.3.5 Noyau

Le module responsable pour la gestion de toutes les étapes de la caractérisation exécutée par l'outil est le noyau (Figure 4-5). Il contrôle les opérations des autres modules, analyseur XML, générateur de scénarios et générateur de résultat. De plus, il doit contrôler les appels à tous les outils utilisés par le *Power Profiler*, soit le compilateur *gcc*, les outils Xilinx utilisés lors de la génération de la liste d'interconnexions de la plate-forme matérielle, le simulateur Modelsim et l'estimateur de puissance RTL XPower. Conséquemment, tous les paramètres d'entrée requis pour l'exécution de ces outils sont, lorsque nécessaire, générés par le noyau, toujours de manière automatique sans aucune intervention externe de l'usager.

4.3.6 Limitation du modèle

Nous pouvons noter que les trois techniques d'estimation de puissance choisies et décrites à la section 3.2.1, négligent la valeur des opérandes de l'instruction. Également, le modèle d'énergie généré par le flot ne considère pas l'effet des opérandes dans la consommation d'énergie des instructions. Tel que discuté précédemment, la dépendance entre la consommation de puissance et les paramètres des instructions est une problématique connue dans la littérature. Une étude approfondie réalisée par [69] avec un processeur Microblaze, a conclu que les valeurs passées en paramètre à une instruction, soit un numéro de registre, une valeur immédiate, une adresse entre autres, ne tend pas à faire une grande variation dans la consommation. L'étude démontre que

81.25 % (39/48) de cas d'études ont une variation maximale de 7 % dans la consommation de puissance lorsque les paramètres d'entrée changent continuellement. Également, Kavvadias et al. [72] démontre qu'avec le processeur ARM7 [82] il existe une faible dépendance entre la consommation du processeur et les valeurs d'entrées des instructions. Finalement, Klass et al. [70] est arrivé à la même conclusion avec le Motorola DSP56000.

Bien sûr, la dépendance entre la consommation d'énergie et les paramètres de l'instruction dépend de l'architecture de chaque processeur. Toutefois tous les projets mentionnés ont démontré que cette dépendance peut être négligée sans toutefois avoir un impact majeur dans les résultats. Basé sur ces études, notre modèle d'énergie ne considère pas les paramètres des instructions, cependant lors de la génération de scénarios l'outil met un effort particulier pour faire varier ces paramètres.

CHAPITRE 5 RÉSULTAT ET ANALYSE

Le processus de caractérisation à travers un flot automatisé présenté dans le chapitre précédent nous a permis de générer deux modèles d'énergie pour processeurs, un pour le Microblaze et l'autre pour le LEON3. Ces proesseurs, de type *« softcore »*, ciblent le FPGA *« xc5vlx50t»* de la famille Virtex5 de la compagnie Xilinx. Ces modèles d'énergie sont employés dans un environnement de test permettant l'estimation de l'énergie consommée de cinq bancs d'essai différents à travers trois techniques d'estimation distinctes. Ce chapitre présente d'abord l'environnement de test développé pour analyser les résultats de ce travail. On présente ensuite des résultats sur la qualité de l'estimation de chaque technique appliquée; soit *instruction sans transition, instruction avec transition unique* et *instruction avec transition*. De plus, on évalue l'accélération qui peut être atteinte par une estimation de puissance au niveau système et on compare la consommation de puissance des deux processeurs. Pour terminer, on analyse l'extensibilité et le caractère modulaire du flot de caractérisation.

5.1 Environnement de test

Pour permettre une étude comparative entre les trois techniques d'estimation (Section 3.2.1), un deuxième outil permettant la comparaison des trois techniques a dû être créé dans le but de déterminer celle qui assure un meilleur compromis entre la qualité des résultats et l'effort de développement. L'objectif de cet environnement de test est de comparer l'estimation de l'énergie consommée fourni par un estimateur RTL, tel que XPower, à l'estimation fournie par les trois techniques de haut niveau. L'estimation de XPower est donc l'estimation de référence et permettra de valider la qualité de ces trois techniques plus haut niveau d'abstraction. Encore une fois, ces résultats seront fondamentaux pour décider laquelle des techniques, s'il en existe une, permet de déterminer précisément la consommation de puissance au niveau instruction. La Figure 5-1 présente un aperçu général de l'environnement de test. Le flot développé responsable pour appliquer les techniques d'estimation au niveau instruction a été nommé *Power Estimator*.

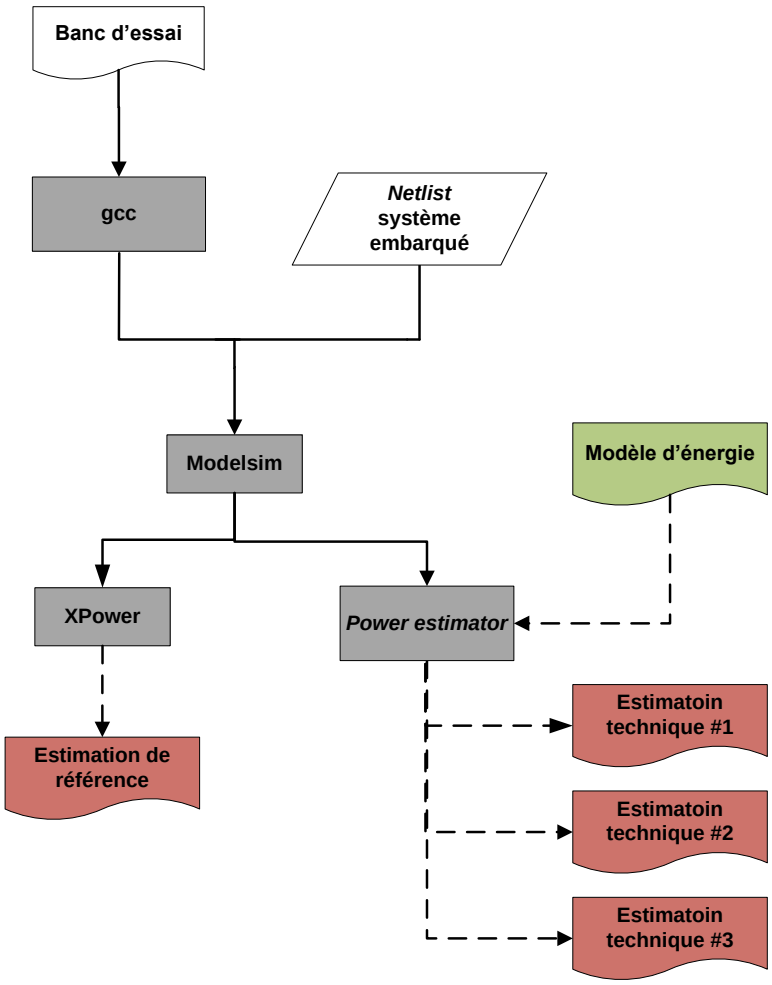

Figure 5-1 Environnement de test

Comme il peut être observé, le point de départ de ce flot est le projet logiciel qui constitue le banc d'essai (l'application) et la spécification de la plate-forme matérielle. Cette dernière a été décrite à la section 4.3.2. Le banc d'essai et la plate-forme matérielle sont simulés avec Modelsim en mode temporelle, soit après placement et routage. Par la suite, l'activité du système est analysée par XPower qui fournit l'estimation de référence. D'autres outils pourraient être utilisés pour obtenir l'estimation de référence. Étant donné que XPower a déjà été utilisée dans notre flot de caractérisation, nous l'avons choisi pour éviter les erreurs dues à l'incertitude ajoutée par l'estimateur de référence. En même temps, l'activité du système est aussi utile à l'outil *Power Estimator*, parce qu'elle contient la trace d'exécution du processeur. Cette trace est, en effet, la liste des instructions exécutée par le processeur en ordre chronologique. Ainsi, *Power Estimator*

se sert de cette trace et du modèle d'énergie pour appliquer les trois techniques d'estimation. La trace et le modèle d'énergie sont les seules données requises. La simulation avec Modelsim est le moyen employé dans l'environnement de test pour obtenir la trace d'exécution du processeur, toutefois dans un environnement de développement haut niveau, la trace pourra être obtenue à travers une simulation beaucoup plus rapides que celle de Modelsim. Indépendamment du format de la spécification d'entrée (comme par exemple une description en langage ADL *« Architecture Description Language »* comme LISA [83] ou une description en langage C/C++ come SystemC [84]), un environnement de développement de haut niveau permet de simuler le design et d'obtenir la trace du processeur plus rapidement.

5.1.1 Bancs d'essai

Pour assurer une vérification des modèles d'énergie et ainsi obtenir des résultats valides, cinq bancs d'essai différents ont été choisis.

- Le **Dhrystone** est un banc d'essai très populaire et largement utilisé pour mesurer les performances générales des divers systèmes. Essentiellement, ce banc d'essai demande peu de mémoire pour être gardé et consiste en une boucle principale exécutée plusieurs fois.

- Le **Coremark** est un banc d'essai simple et sophistiqué conçu spécifiquement pour tester les fonctionnalités du noyau des processeurs [85]. Il consiste en plusieurs tâches habituellement exécutées par des applications embarquées.

- Le **Quicksort,** un des algorithmes le plus populaires pour le tri rapide. D'autres travaux [2, 86] utilisent également cet algorithme lors de l'analyse de performance de leur modèle d'énergie.

- La résolution d'**Équations mathématiques du troisième degré.** Cet algorithme provient d'un ensemble de bancs d'essai, le MiBench [87]. Ce dernier regroupe plusieurs bancs d'essai pour les systèmes embarqués.

- La **recherche de sous chaînes**. Un logiciel qui exécute l'algorithme de recherche de sous-chaines pour plusieurs valeurs d'entrée. Cet algorithme fait aussi partie du banc d'essai MiBench.

5.2 Estimation niveau système

Dans cette section, nous présenterons les résultats qui ont été obtenus lors de l'exécution de chaque banc d'essai présenté à la section précédente. Naturellement, les résultats sont divisés en fonction des processeurs Microblaze et LEON.

5.2.1 Microblaze

Ce processeur a été simulé à une fréquence de 100MHz. Les tableaux qui suivront montrent les résultats de l'estimation de l'énergie et de la puissance consommées par ce processeur pour chaque banc d'essai.

Les deux premières colonnes des tableaux 4 à 8, *Puissance* et *Énergie* indiquent les valeurs estimées respectivement par chaque technique d'estimation. Ainsi, pour chaque estimateur (les trois premières lignes des tableaux 4 à 8), la puissance consommée représente le rapport entre l'énergie consommée estimée à haut niveau et le temps de simulation soit :

$$Puissance = \frac{\acute{E}nergie}{Temps} \tag{5.1}$$

Pour la quatrième ligne des tableaux 4 à 8, la formule opposée est utilisée, c'est-à-dire qu'on obtient l'énergie à partir de la puissance fournie par XPower.

Ensuite, pour les trois premières lignes des tableaux 4 à 8, la quatrième colonne, *% Erreur*, exprime l'erreur de mesure commise par chaque technique haut niveau par rapport à XPower. Finalement, la dernière colonne, *% Énergie calculée*, indique le pourcentage de l'énergie estimée. Tel qu'expliqué auparavant, le modèle d'énergie généré ne comporte pas toutes les instructions du processeur, mais il se peut que des instructions qui ne sont pas dans le modèle d'énergie soient exécutées par le processeur. Ces dernières ne sont pas dans l'estimation fournie par nos 3 estimateurs.

Tableau 5-1 Résultats estimation Dhrystone pour le Microblaze

Dhrystone				
	Puissance (mW)	**Énergie (uJ)**	**% Erreur**	**% Énergie calculée**
Instruction sans transition	24.13	111.3586	-36.57%	99.86%
Instruction avec transition unique	45.86	211.5925	20.52%	99.86%
Instruction avec transition	44.37	204.7088	16.60%	99.86%
XPower	**38.05**	**175.5627**		

Tableau 5-2 Résultats estimation Coremark pour le Microblaze

Coremark				
	Puissance (mW)	**Énergie (uJ)**	**% Erreur**	**% Énergie calculée**
Instruction sans transition	14.69	36.6834	-54.51%	98.90%
Instruction avec transition unique	33.30	83.1470	3.11%	98.90%
Instruction avec transition	26.96	67.3181	-16.52%	98.90%
XPower	**32.3**	**80.6399**		

Tableau 5-3 Résultats estimation Quicksort pour le Microblaze

Quicksort				
	Puissance (mW)	**Énergie (uJ)**	**% Erreur**	**% Énergie calculée**
Instruction sans transition	22.29	10.6965	-45.32%	99.88%
Instruction avec transition unique	43.48	20.8625	6.65%	99.88%
Instruction avec transition	41.48	19.9033	1.75%	99.88%
XPower	**40.77**	**19.5610**		

Tableau 5-4 Résultats estimation Équation du 3ième degré pour le Microblaze

Équation 3^{ième} degré				
	Puissance (mW)	**Énergie (uJ)**	**% Erreur**	**% Énergie calculée**
Instruction sans transition	42.94	112.6613	-32.16%	97.02%
Instruction avec transition unique	67.32	176.6241	6.36%	97.02%
Instruction avec transition	62.53	164.0720	-1.20%	97.02%
XPower	**63.29**	**166.0641**		

Tableau 5-5 Résultats estimation Recherche de sous chaînes pour le Microblaze

Recherche sous chaîne				
	Puissance (mW)	**Énergie (uJ)**	**% Erreur**	**% Énergie calculée**
Instruction sans transition	29.51	73.2416	-38.68%	99.99%
Instruction avec transition unique	56.14	139.3179	16.63%	99.99%
Instruction avec transition	50.24	124.6852	4.38%	99.99%
XPower	**48.13**	**119.4491**		

Plusieurs conclusions s'imposent en analysant l'erreur d'estimation de chaque technique. Tout d'abord, la technique d'estimation *instruction sans transition* comporte les % d'erreur les plus élevées. On remarque aussi que les erreurs sont toujours négatives. Ce qui est tout à fait normal étant donné que cette technique ne considère pas l'énergie consommée par le processeur lors de la transition entre deux instructions. Ainsi, cette technique tend à sous-estimer l'énergie consommée.

D'autre part, on note que la technique *instruction avec transition unique* fournit les valeurs d'estimation les plus élevées. Cette technique arrondit toutes les transitions d'une instruction i par la transition entre l'instruction i et l'instruction nop. L'erreur de cette approximation est positive pour quelques couples d'instructions et négative pour d'autres, ainsi, puisque l'erreur de l'approximation passe du positif au négatif, l'erreur s'annule à long terme. Conséquemment, même avec les estimations plus élevées, cette deuxième technique fournit une bonne estimation avec une erreur modérée.

Enfin, on remarque que la troisième technique *instruction avec transition* donne d'excellentes valeurs d'estimation avec de très petites erreurs pour trois bancs d'essai : Quicksort, Équation du 3^{ième} degré et Recherche de sous chaînes. Les erreurs assez accentuées pour le banc d'essai Coremark est expliqué par l'estimation du coût de base et l'estimation du coût inter-instructions associées aux instructions « *load* » et « *store* ». Les coûts de base de ces instructions, et conséquemment tous les coûts de transition associés à eux dans le modèle d'énergie du processeur Microblaze, sont sous-estimés. En effet, lors de la caractérisation de ces instructions, les paramètres de ces dernières ne varient pas beaucoup. À la section 4.3.6, il a été expliqué que la variation de paramètres des instructions a une influence sur la consommation, toutefois cela a été négligé dans notre modèle d'énergie. La non-variation de paramètres dans le cas des instructions « *load* » et « *store* » est due au fait que les accès mémoire doivent être alignés par de mots de 32 bits. Cependant, les valeurs auxquelles les registres ont été initialisés, dans la section « *setup* » des scénarios, empêchaient la génération de différentes combinaisons de registres sources permettant un accès mémoire qui respecte l'alignement. Conséquemment, le banc d'essai Coremark est sous-estimé parce que plus de la moitié des instructions exécutées par ce dernier sont des « *load* » et « *store* ». En ce qui concerne le banc d'essai Dhrystone, les deux techniques d'estimations *instruction avec transition unique* et *instruction avec transition* obtiennent une forte erreur. Étant donné que l'erreur se retrouve dans les deux estimations, on considère qu'elle est due aux incertitudes dans le modèle d'énergie.

On peut donc affirmer que le modèle d'énergie produit automatiquement par notre flot *Power Profiler* pour le processeur Microblaze est d'une très bonne qualité et qu'il permet l'obtention de

bonnes estimations avec les techniques *instruction avec transition unique* et *instruction avec transition*.

5.2.2 LEON

Le processeur LEON a été simulé à une fréquence de 75MHz. Les tableaux qui suivent montrent les résultats de l'estimation de l'énergie et de la puissance consommées par ce processeur pour chacun des bancs d'essai. Chaque tableau (9 à 13) possède les quatre colonnes *Puissance*, *Énergie*, *% Erreur* et *% Énergie calculée* tel qu'expliqué pour le processeur Microblaze.

Tableau 5-6 Résultats estimation Dhrystone pour le LEON3

Dhrystone				
	Puissance (mW)	**Énergie (uJ)**	**% Erreur**	**% Énergie calculée**
Instruction sans transition	20.93	72.1773	-18.12%	95.08%
Instruction avec transition unique	25.89	89.2916	1.29%	95.08%
Instruction avec transition	25.78	88.9142	0.86%	95.08%
XPower	**25.56**	**88.1547**		

Tableau 5-7 Résultats estimation Coremark pour le LEON3

Coremark				
	Puissance (mW)	**Énergie (uJ)**	**% Erreur**	**% Énergie calculée**
Instruction sans transition	18.61	60.8724	-24.70%	99.55%
Instruction avec transition unique	23.44	76.6621	-5.17%	99.55%
Instruction avec transition	24.70	80.7656	-0.10%	99.55%
XPower	**24.72**	**80.8440**		

Tableau 5-8 Résultats estimation Quicksort pour le LEON3

Quicksort				
	Puissance (mW)	**Énergie (uJ)**	**% Erreur**	**% Énergie calculée**
Instruction sans transition	21.58	25.1319	-22.22%	96.11%
Instruction avec transition unique	27.49	32.0238	-0.90%	96.11%
Instruction avec transition	27.93	32.5367	0.69%	96.11%
XPower	**27.74**	**32.3130**		

Tableau 5-9 Résultats estimation Équation du 3ième degré pour le LEON3

Équation 3^{ième} degré				
	Puissance (mW)	**Énergie (uJ)**	**% Erreur**	**% Énergie calculée**
Instruction sans transition	22.71	102.7515	-17.92%	96.11%
Instruction avec transition unique	28.96	130.9975	4.65%	96.11%
Instruction avec transition	29.09	131.6262	5.15%	96.11%
XPower	**27.67**	**125.1817**		

Tableau 5-10 Résultats estimation Recherche de sous chaînes pour le LEON3

Recherche sous chaîne				
	Puissance (mW)	**Énergie (uJ)**	**% Erreur**	**% Énergie calculée**
Instruction sans transition	24.85	83.6819	-13.94%	98.88%
Instruction avec transition unique	31.90	107.4303	10.48%	98.88%
Instruction avec transition	30.94	104.2249	7.18%	98.88%
XPower	**28.87**	**97.2395**		

Selon les tableaux, on remarque encore une fois que la technique *instruction sans transition* comporte les erreurs les plus élevées. De même, les erreurs sont toujours négatives. Ceci conforme aux résultats obtenus et discutés avec le Microblaze.

Par contre, contrairement au processeur Microblaze, la technique *instruction avec transition unique* ne fournit pas toujours les valeurs d'estimation les plus élevées. On peut aussi remarquer que les valeurs d'estimation fournies par cette dernière et les valeurs d'estimation fournies par la

technique *instruction avec transition* sont toujours très proches. Ce qui démontre que l'approximation établie par la technique *instruction avec transition unique* peut être convenable pour ce processeur. En somme, la précision de ces deux techniques est excellente, avec une erreur maximale de 10.48 % pour la technique *instruction avec transition unique* et de seulement 7.18 % pour la technique *instruction avec transition*. Ceci prouve que les incertitudes dans le modèle d'énergie du processeur LEON sont infimes et le modèle généré de manière automatique est d'une qualité remarquable.

En terminant, on peut évidemment constater que le pourcentage de l'énergie calculée de chaque banc d'essai pour chaque processeur n'est pas égal. En effet, chaque compilateur croisé cible l'ensemble d'instructions du jeu d'instructions respectif, ce qui génère des instructions différentes pour l'exécution d'une même tâche.

5.3 Discussion de résultats

5.3.1 Les techniques d'estimation

La section précédente a permis d'expliquer les résultats d'estimation obtenus par les trois techniques de haut niveau lors de la simulation de chaque banc d'essai. En se comparant à XPower, il a été démontré que le modèle d'énergie généré pour les deux processeurs nous permet d'obtenir une très bonne estimation en appliquant les techniques *instructions avec transition unique* et *instruction avec transition*.

Pour compléter ce chapitre, nous allons comparer plus en détails la performance de chaque technique d'estimation de haut niveau. Pour réaliser cette analyse, nous nous sommes dotés de deux indices statistiques. Parmi ceux-ci nous retrouvons tout d'abord, l'EQM « Erreur Quadratique Moyenne » qui est une mesure de l'erreur moyenne pondérée par le carré de l'erreur et caractérise la précision d'un ensemble de mesure. Parce qu'il s'agit d'une quantité au carré, l'erreur quadratique moyenne est plus influencée par les grandes erreurs que par les petites erreurs [88]. Cette erreur est définie à travers l'équation mathématique suivante :

$$EQM = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(V_E - V_R)^2} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(erreur)^2} \quad (5.2)$$

où V_E est la valeur estimée, V_R est la valeur de référence et N est le nombre de points de vérification, soit le nombre de bancs d'essai. De plus, nous avons aussi utilisé l'écart type des erreurs dans le but de faire l'analyse des résultats. Cette valeur mesure la dispersion des valeurs par rapport à la moyenne, ainsi il exprime le caractère, homogène ou hétérogène de l'ensemble d'erreurs. L'équation mathématique suivante définit cette valeur :

$$\text{É}cart\ type = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(e_i - \bar{e})^2} \quad (5.3)$$

où e_i est l'erreur de chaque estimation, $\bar{e}$ est la moyenne des erreurs et N est le nombre de points de vérification, soit le nombre de bancs d'essai.

Grâce à ces indices, nous avons pu résumer l'ensemble de résultats pour chaque processeur avec seulement deux valeurs. Les tableaux qui suivent présentent l'erreur quadratique moyenne et l'écart type de l'ensemble des erreurs de chaque technique pour chaque processeur.

Tableau 5-11 Analyse de l'erreur des estimations pour le Microblaze

	Erreur quadratique moyenne	**Écart type des erreurs**
Instruction sans transition	42.17 %	8.71 %
Instruction avec transition unique	12.59 %	7.49 %
Instruction avec transition	10.70 %	11.91 %

Tableau 5-12 Analyse de l'erreur des estimations pour le LEON3

	Erreur quadratique moyenne	**Écart type des erreurs**
Instruction sans transition	19.74 %	4.18 %
Instruction avec transition unique	5.67 %	5.90 %
Instruction avec transition	3.98 %	3.21 %

À la lumière de données présentées, on constate que pour les deux processeurs la technique *instruction sans transition* ne permet pas une estimation minimalement efficace. L'erreur quadratique moyenne dépasse les 40% pour le processeur Microblaze et reste au tour de 20% pour le processeur LEON. Ce qui est inacceptable, si on envisageait d'appliquer cette technique pour l'estimation d'énergie de processeurs dans un environnement de conception au niveau système. Toutefois, ce modèle nous permettre d'obtenir des résultats plus précis si on ajoute un facteur de correction à son estimation, par exemple l'erreur quadratique moyenne. Ainsi, si on additionne l'erreur quadratique moyenne à chaque estimation fournie par cette technique, on obtiendrait déjà de meilleurs résultats. Une approche similaire a été proposée par [89].

Pour ce qui est de la précision de deux autres techniques d'estimation, les résultats sont éloquents. Les erreurs quadratiques moyennes de ces deux techniques sont plus grandes pour le processeur Microblaze. Cependant, nous avons noté lors de la présentation des résultats à la section précédente que le modèle d'énergie de ce processeur était moins précis que celui du processeur LEON. Pour le processeur LEON, les résultats sont satisfaisants en plus d'avoir une excellente précision. Pour ce même processeur, la technique *instruction avec transition* démontre un résultat discrètement meilleur que la technique *instruction avec transition unique*. D'ailleurs, pour le processeur Microblaze, nous ne pouvons pas affirmer qu'une technique est meilleure que l'autre, étant donné la proximité de l'erreur quadratique moyenne et de l'écart type.

En ce qui concerne la technique avec le meilleur compromis entre précision et effort de caractérisation, la technique *instruction avec transition unique* semble avoir un faible avantage.

La qualité d'estimation et la précision de ces deux techniques sont convaincantes et très semblables pour les deux processeurs. Paradoxalement, l'effort de caractérisation est extrêmement plus grand pour générer un modèle d'énergie adapté à la technique *instruction avec transition* (ceci doit tenir compte de l'effet inter-instructions de toutes les paires d'instructions). L'effort de caractérisation pour générer le modèle d'énergie présenté dans ce travail est d'ordre n^2, n étant le nombre d'instructions (ceci est décrit à la section 4.2.1). D'un autre côté, un modèle d'énergie plus simple qui répond aux besoins de la technique *instruction avec transition unique* demande seulement la caractérisation de chaque instruction i et chaque couple $i - nop$, ce qui nécessite un effort de caractérisation d'ordre n, n étant toujours le nombre d'instructions.

Enfin, on conclut que la qualité des techniques ne se fait pas nécessairement au détriment de l'effort de caractérisation. Toutefois, le jugement minutieux de ce que l'usager favorise lors de l'analyse du compromis entre la précision et l'effort de caractérisation s'avère nécessaire.

5.3.2 Accélération du temps de simulation

Il est relativement long de simuler et faire l'estimation RTL de la consommation de puissance d'une plate-forme matérielle. Ce temps est de l'ordre de plusieurs minutes, voire quelques heures. Dépendamment de la plate-forme matérielle et de la performance du simulateur utilisé, ce temps peut devenir prohibitif s'il est nécessaire d'évaluer diverses implémentations. À travers les techniques d'estimation de puissance au niveau système, on cherche donc à minimiser le temps requis par le processus d'estimation. Les résultats présentés au Tableau 5-13 et au Tableau 5-14 représentent l'accélération qu'il est possible d'atteindre avec une estimation de puissance de haut niveau basé sur un modèle d'énergie matricielle en deux dimensions. Les temps nécessaires à l'exécution des estimations ont été mesurés sur un poste de travail Windows 7 Entreprise 64 bits avec un processeur Intel i7 à 2.67GHz et 6GB de mémoire RAM.

Tableau 5-13 Comparaison du temps d'estimation de la puissance consommée par chaque bancs d'essai pour le Microblaze

Banc d'essai	Temps simulé (millisecondes)	Temps estimation RTL (secondes)	Temps estimation niveau système (secondes)	Accélération
Dhrystone	4.770	11040	1.352	8165.7
Coremark	2.500	5760	0.688	8372.1
Quicksort	0.482	1140	0.138	8260.9
Équation 3^{ième} degré	2.630	7440	0.723	10290.5
Recherche sous chaîne	2.500	6420	0.697	9210.9

Tableau 5-14 Comparaison du temps d'estimation de la puissance consommée par chaque bancs d'essai pour le LEON3

Banc d'essai	Temps simulé (millisecondes)	Temps estimation RTL (secondes)	Temps estimation niveau système (secondes)	Accélération
Dhrystone	3.45	4860	0.511	9510.8
Coremark	3.27	4620	0.442	10452.5
Quicksort	1.16	1620	0.190	8526.3
Équation 3^{ième} degré	4.52	6420	0.759	8458.5
Recherche sous chaîne	3.37	4920	0.602	8172.8

Dans les deux tableaux, le *Temps simulé* correspond au temps que devrait prendre l'exécution sur le FPGA. D'autre part, le *Temps d'estimation RTL* est le temps pris par XPower pour estimer la puissance consommée par le système embarqué. Finalement, le *Temps d'estimation niveau système* est le temps requis par l'outil *Power Estimator* pour calculer l'estimation de la puissance consommée avec les techniques au niveau instruction, à partir de la trace d'exécution du processeur. La dernière colonne représente l'accélération (*Temps d'estimation RTL* divisé par *Temps d'estimation niveau système*).

Tel que mentionné plus haut, il est important de noter que le temps simulé de chaque banc d'essai n'est pas le même pour les deux processeurs, étant donné les jeux d'instructions différents et le temps d'accès à la mémoire d'instruction et à la mémoire de données différents. De plus, étant donnée la complexité des bancs d'essai Coremark et Dhrystone, ces derniers ne sont pas exécutés complètement.

Finalement, notons que le temps d'estimation RTL est de l'ordre des milliers de secondes alors que le temps pris pour les estimations de haut niveau est de l'ordre des millisecondes. Les estimations de haut niveau permettent donc d'accélérer grandement le calcul d'estimation de puissance. Cette accélération est de plus de huit mille fois pour tous les bancs d'essai (incluant les deux processeurs). Ceci est un gain remarquable et s'explique par la simplicité de l'algorithme de calcul dans le flot *Power Estimator*. Celui-ci doit seulement associer un coût de base pour chaque instruction et un coût inter-instructions pour chaque transition à l'aide du modèle d'énergie. De toute évidence, cet algorithme peut être facilement intégré dans n'importe quelle plate-forme virtuelle qui permettrait l'obtention de la trace d'exécution du processeur, tel que SpaceStudio [10].

5.3.3 La fréquence d'opération et le placement et routage

Tel que mentionné aux chapitres précédents, ce travail donne suite à un premier projet de recherche sur l'estimation de la consommation de puissance des périphériques au niveau système réalisé dans notre laboratoire. Ce dernier a été d'extrême importance et a servi comme point de départ pour ce mémoire. Ainsi, nous avons tenu pour acquises plusieurs conclusions de ce premier projet sans faire une analyse exhaustive pour les vérifier. Parmi ceux-ci, nous avons tout d'abord pris en compte les effets de la fréquence d'opération du processeur pour l'estimation de la consommation. Dans le fichier d'entrée du flot *Power Profiler*, la fréquence d'opération du processeur est indiquée et le processeur est caractérisé à cette fréquence. Toutefois, ceci ne veut pas dire que le modèle d'énergie généré est seulement valide lorsque la fréquence d'opération du processeur est identique à celle choisie pour la caractérisation. En effet, en accord avec [8], pour différentes fréquences la consommation d'énergie reste stable. La Figure 5-2 présente la

consommation d'énergie de l'instruction *addkc* à différentes fréquences. On observe que l'énergie reste stable même si la fréquence change. Ceci dit, le modèle d'énergie généré peut être employé pour l'estimation de l'énergie consommée indépendamment de la fréquence d'opération du processeur. Il est important de noter que l'énergie consommée reste stable, toutefois la puissance consommée change.

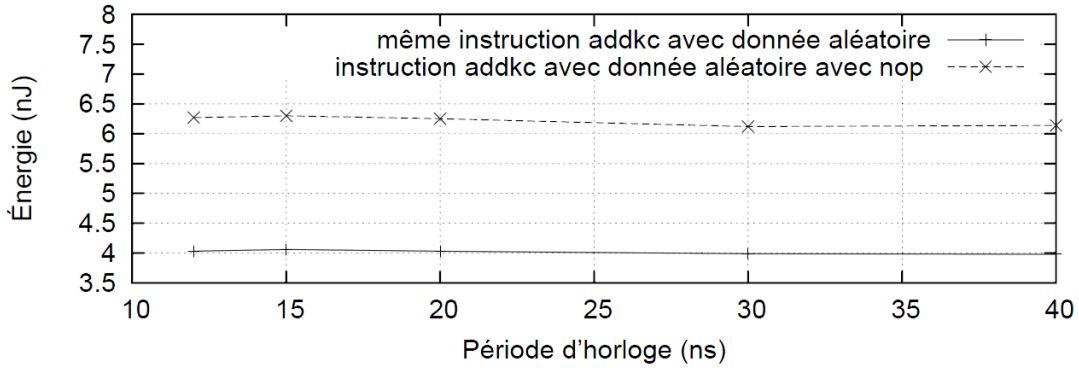

Figure 5-2 Énergie consommée par l'instruction addkc pour différentes fréquences [8]

Également, nous avons pris en compte les effets que le placement et routage peut causer sur la consommation du processeur. Ce même travail affirme que lors du placement et routage les périphériques plus volumineux en ressources, ceux qui sont plus complexes avec plus d'interconnexions, sont routés en premier pour obtenir les meilleures performances, car ses contraintes sont plus difficiles à rencontrer. Ainsi, il conclut que les périphériques plus complexes ont un placement et routage assez stable. Cette conclusion nous assure que les modèles d'énergies générés par *Power Profiler* ne se limitent pas à la plate-forme matérielle utilisée et au placement et routage obtenu lors du processus de caractérisation. Ces modèles peuvent être employés dans d'autres configurations architecturales sans que le placement et routage vienne affecter la précision de l'estimation.

Bien évidemment, le placement et routage est un processus très complexe. Une étude plus complète et approfondie de cette problématique est recommandée.

5.4 Extensibilité et adaptabilité

Au cours du cycle de développement de notre projet, nous avons mis un grand effort sur la qualité du logiciel dans le but de concevoir un outil extensible à d'autres processeurs configurables de type *« soft-processors »* et portable à d'autres technologies. Cette section démontre les possibilités d'adaptation de l'outil.

La Figure 5-3 montre le flot de tâches lors du processus de caractérisation. Sur cette illustration, le module responsable de chaque tâche n'est pas indiqué. Toutefois, toutes les tâches illustrées sur la figure sont accomplies par les modules de l'outil qui ont été présentés à la section 4.3. Le flot est illustré par treize tâches bien précises et indépendantes entre elles qui peuvent facilement être adaptées. À titre d'exemple, le tableau suivant propose des changements possibles à apporter à l'outil et indique respectivement les tâches à être modifiées:

Tableau 5-15 Tâches à modifier pour adapter l'outil *Power Profiler*

	Tâches à modifier
Support à un autre « soft-processor », ciblant toujours les FPGAs de la compagnie Xilinx.	5 et 8
Support aux FPGAs d'une autre compagnie.	5, 6, 7, 10 et 11
Support à un autre format de fichier d'entrée.	3

- Pour supporter un autre *« softcore »*, ciblant toujours les FPGAs de la compagnie Xilinx, il faut modifier la tâche responsable de changer les spécifications du système embarqué données en entrée (tâche 5) et la tâche responsable de la génération des scénarios (tâche 8). Cette dernière est totalement dépendante du processeur à caractériser.

- En vue de supporter les FPGAs d'autres compagnies, il faudrait premièrement modifier la tâche responsable de changer les spécifications du système embarqué donnée en entrée (tâche 5). Ensuite, il serait nécessaire d'adapter la tâche responsable de générer la liste d'interconnexion (tâche 6), la tâche qui génère le modèle HDL pour la simulation (tâche 7) et les tâches responsables de l'estimation de puissance (tâches 10 et 11). Normalement, l'outil qui génère le modèle de simulation HDL et l'outil qui estime la puissance du modèle RTL sont normalement fournis dans l'environnement de développement de la technologie qu'on cible.
- Finalement, afin de supporter un autre format de fichier d'entrée, seule la tâche qui interprète ce fichier doit être modifiée (tâche 3).

Nous pouvons en conclure que les tâches peuvent être facilement adaptées à d'autres environnements. Ainsi, l'outil est extensible et adaptable à d'autres processeurs et à d'autres technologies.

5.5 Comparaison de résultats

Lors du premier projet de recherche réalisé dans notre laboratoire [8], les bancs d'essai Dhrystone et Coremark ont aussi été utilisés. Pour le processeur Microblaze, les estimations obtenues avec une technique de type *instruction avec transition unique* avaient une erreur de 2.31 % et -11.38 % pour chaque banc d'essai respectivement. Bien que ces résultats soient identiques à ceux que nous avons obtenus, la caractérisation complétée dans ce travail était complètement manuelle tel que discuté à la section 3.2. En outre, Jingzhao et al. [1] présente aussi une méthodologie de caractérisation pour le processeur Microblaze et obtient une erreur maximale de 7.8 %. Par contre, les bancs d'essais utilisés ne sont pas les mêmes employés dans notre projet. Une autre fois, cette caractérisation n'a pas été faite avec un outil automatisé.

En conclusion, nous sommes en mesure d'obtenir des résultats relativement précis à partir d'une caractérisation automatisée.

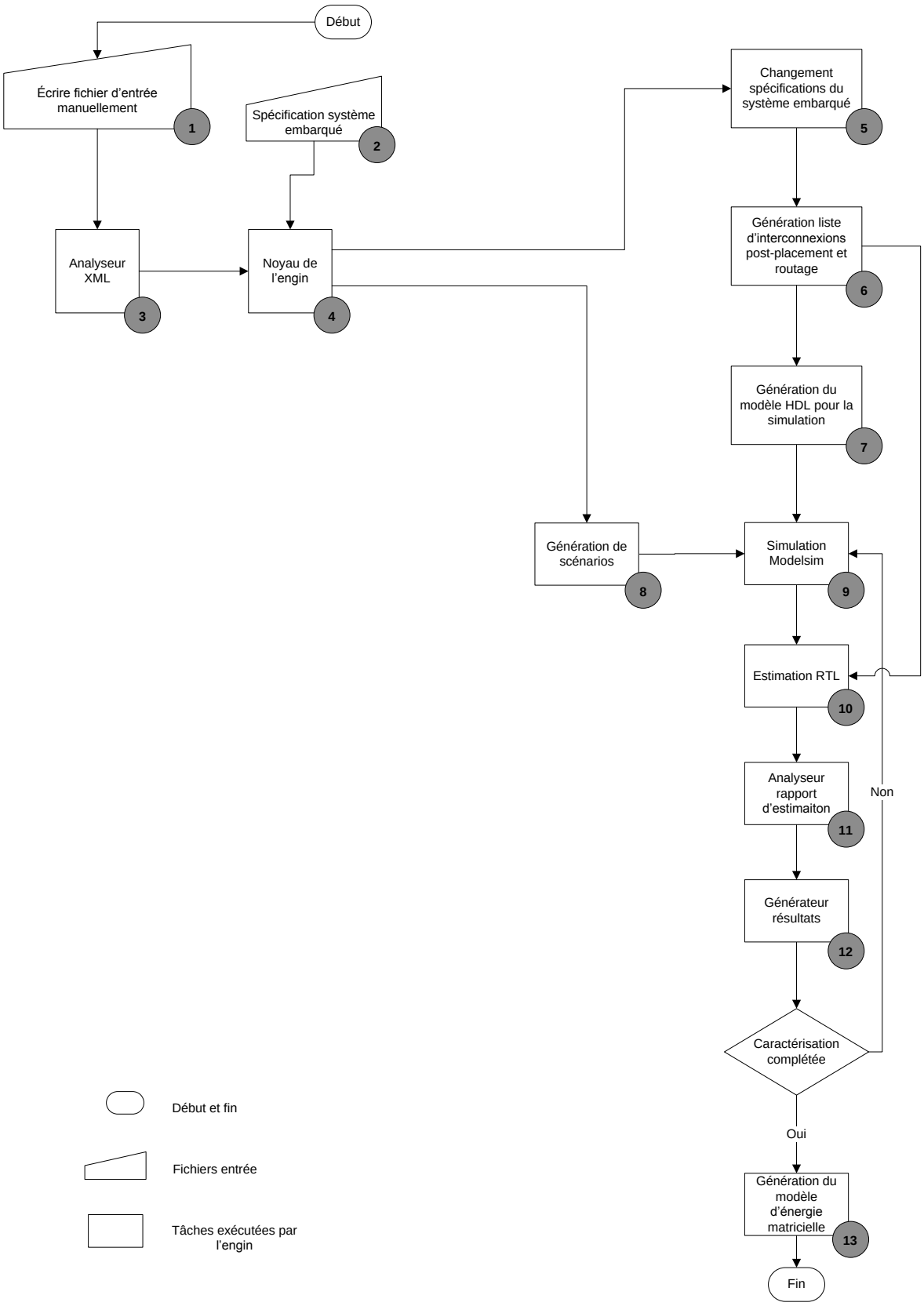

Figure 5-3 Flot de tâches de l'outil Power Profiler

CHAPITRE 6 CONCLUSION ET TRAVAUX FUTURS

Dans un contexte où l'industrie des systèmes embarqués ne cesse d'innover dans le but de créer de nouveaux produits afin d'attirer de nouveaux usagers et où le temps de mise en marché doit être le plus court possible, la consommation de puissance de circuits devient une problématique importante. Les diverses techniques actuellement étudiées pour contrôler l'augmentation de la consommation de puissance et aussi diminuer les impacts d'une consommation trop élevée sont devenues impératif lors de la conception de ces systèmes.

L'objectif principal de ce mémoire était donc de, premièrement, concevoir un outil capable de caractériser automatiquement la consommation de puissance des *« soft-processors »* et de générer un modèle d'énergie efficace pour l'estimation de la consommation au niveau système. Le deuxième objectif était d'évaluer trois techniques d'estimation d'énergie au niveau instructions et choisir celle qui s'avère avoir le meilleur compromis entre la qualité du résultat fourni et le temps de développement. Après avoir passé en revue plusieurs travaux dans la littérature, certains concepts de ceux-ci ont été retenus et ont été adaptés à nous propres besoins. De plus, un projet d'analyse de la consommation de la puissance des diverses composantes matérielles réalisées dans notre laboratoire de recherche nous a servi de point de départ.

En conclusion, nous avons développé un flot automatisé capable de caractériser la consommation de puissance des processeurs Microblaze et LEON3 et de générer un modèle d'énergie matricielle en deux dimensions. La méthodologie appliquée par l'outil a été minutieusement décrite dans ce mémoire. Cette première version de l'outil se restreint aux FPGAs de la compagnie Xilinx. Toutefois, il n'est pas limité à un FPGA spécifique, ni à une famille de FPGA. À travers, le fichier d'entrée l'usager peut cibler le FPGA et le processeur de son choix ainsi que l'ensemble d'instructions à caractériser. Également, l'outil a été développé de façon modulaire, ce qui lui permet d'être facilement adapté pour supporter la caractérisation d'autres processeurs embarqués et aussi pour cibler les FPGAs d'une autre compagnie. Grâce à cet outil automatisé, le temps pour caractériser la consommation de puissance d'un nouveau processeur n'est que de quelques

semaines contrairement aux quelques mois qu'une caractérisation manuelle peut facilement demander.

L'une des contributions les plus importantes de ce travail a été l'évaluation des trois techniques d'estimation de puissance de haut niveau. La qualité des résultats de la technique *instruction sans transition* a été très mauvaise et son application a tendance à sous-estimer la consommation. Une amélioration à cette estimation serait l'utilisation d'un facteur de correction. Par contre, les résultats obtenus avec la technique *instruction avec transition unique* et la technique *instruction avec transition* ont été très pertinents. Dans le cas du processeur Microblaze, les résultats de ces deux techniques ont été très semblables. Par contre, pour le processeur LEON3, la technique *instruction avec transition* a présenté un faible avantage. Toutefois, l'effort pour la génération du modèle d'énergie adapté à chaque technique est grandement différent. Les caractéristiques uniques de ces deux techniques leur confèrent donc certains avantages l'un par rapport à l'autre, sans toutefois pouvoir déclarer que l'une est plus appropriée que l'autre dans le contexte de notre travail. De plus, de façon générale, cette évaluation a démontré que le modèle d'énergie généré est d'une grande qualité puisque les résultats obtenus par ces techniques sont très bons par rapport aux estimations de l'outil XPower.

Somme toute, dans un environnement de développement ESL, le modèle d'énergie généré par le flot automatisé permet d'accélérer grandement l'estimation de puissance de processeurs embarqués. La méthodologie proposée permet d'atteindre une accélération de plus de huit mille fois sur le flot de simulation et estimation traditionnel, tout en sacrifiant, seulement, au tour de 10 % la précision pour le processeur Microblaze et 4 % pour le processeur LEON3.

6.1 Travaux futurs

En ce qui concerne les travaux futurs, il serait fort intéressant d'utiliser l'outil pour caractériser une architecture de type DSP. Ces architectures ont un riche mode d'adressage et la capacité de traiter les instructions en parallèle ce qui entrave la classification des instructions en différents

groupes. Ainsi un tel outil s'avère idéal pour caractériser le jeu d'instruction de cette gamme de processeurs.

Également, il serait envisageable d'étudier la contribution énergétique du pipeline d'un processeur par rapport à celle de la mémoire cache. Si la consommation de la cache se confirme comme étant d'une grande importance, il serait souhaitable de concevoir un modèle d'énergie qui modélise la mémoire cache du processeur. La mémoire cache est d'extrême importance pour la performance d'un système embarqué, elle se trouve dans pratiquement tous les processeurs qui envisagent un minimum de vitesse d'exécution. Dans la littérature, plusieurs projets étudient ce sujet comme par exemple [90] qui analyse la consommation d'une mémoire *« Block Random Access Memory »* (BRAM) sur les FPGAs de Xilinx en se basant sur les paramètres qui la définissent. Il pourrait être également intéressant d'ajouter dans le modèle d'énergie d'autres instructions plus spécialisées du processeur telles que les instructions de l'unité point flottante, pour s'assurer de la qualité des résultats. De plus,

Notamment, certaines optimisations pourraient être apportées à la méthode proposée pour augmenter son efficacité. Tout d'abord, le modèle d'énergie du Microblaze a présenté de résultats moins satisfaisants que ceux du processeur LEON. Il serait souhaitable de travailler la problématique avec les instructions « *load* » et « *store* » pour obtenir un meilleur résultat. Également, il serait intéressant de pouvoir mesurer la consommation du processeur directement sur la carte FPGA. Bien que l'estimation de XPower semble être très bonne, la mesure physique serait plus fidèle et considérablement plus rapide que l'estimation RTL.

Finalement, dans le cas de deux processeurs, le processus de caractérisation effectué par le flot automatisé prend environ dix jours pour se compléter. Une approche pour réduire ce temps est l'utilisation des « Application Programmable Interface » (APIs) pour le calcul parallèle. Pour y arriver, il serait intéressant et astucieux de dédier plusieurs machines exclusivement au processus de caractérisation, ceci diminuerait drastiquement le temps requis pour l'exécution.

RÉFÉRENCES

[1] O. Jingzhao and V. K. Prasanna, "Rapid energy estimation of computations on FPGA based soft processors," in *SOC Conference, 2004. Proceedings. IEEE International*, 2004, pp. 285-288.

[2] M. Wendt, M. Grumer, C. Steger, R. Weiss, U. Neffe, and A. Muehlberger, "Tool for Automated Instruction Set Characterization for Software Power Estimation," *Instrumentation and Measurement, IEEE Transactions on*, vol. 59, pp. 84-91, 2010.

[3] J. Ou and V. Prasanna, "Rapid Energy Estimation for Hardware-Software Codesign Using FPGAs," *EURASIP Journal on Embedded Systems*, vol. 2006, p. 098045, 2006.

[4] XILINX. (13/12/2012). *MicroBlaze Processor Reference Guide v13.2*. Available: http://www.xilinx.com/support/documentation/sw_manuals/xilinx13_2/mb_ref_guide.pdf

[5] M. Graphics. (04/02/2013). *ModelSim Advanced Simulation and Debugging*. Available: http://model.com/

[6] XILINX. (12/12/2012). *XPower Analyzer (XPA)*. Available: http://www.xilinx.com/products/design_tools/logic_design/verification/xpower_an.htm

[7] B. Bailey, Martin, G. et Piziali, A. , *ESL Design and Verification : A Prescription for Electronic System-Level Methodology*. San Francisco, CA.: Morgan Kaufmann, 2007.

[8] M. Rogers-Vallée, "Une méthode d'estimation de la consommation de puissance pour un système sur puce " M.Sc.A., École Polytechnique de Montréal, Montréal, Qc, Canada, 2011.

[9] L. Filion, Cantin, M.-A., Moss, L., Aboulhamid, M. et Bois, G. , "Space Codesign : A SystemC framework for fast exploration of hardware/software systems.," presented at the Design& Verification Conference and Exhibition, San Jose, CA, 2007.

[10] S. Codesign. (20/12/2012). *Space Codesign*. Available: http://www.spacecodesign.com/

[11] F. N. Najm, "Power estimation techniques for integrated circuits," in *Computer-Aided Design, 1995. ICCAD-95. Digest of Technical Papers., 1995 IEEE/ACM International Conference on*, 1995, pp. 492-499.

[12] K.-S. Chong, G. Bah-Hwee, and J. S. Chang, "A Low Energy FFT/IFFT Processor for Hearing Aids," in *Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium on*, 2007, pp. 1169-1172.

[13] N. Kulkarni, N. Nukala, and S. Vrudhula, "Minimizing area and power of sequential CMOS circuits using threshold decomposition," in *Computer-Aided Design (ICCAD), 2012 IEEE/ACM International Conference on*, 2012, pp. 605-612.

[14] M. Pedram, "Power minimization in IC design: principles and applications," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 1, pp. 3-56, 1996.

[15] S. M. Srinivas Devadas, "A Survey of Optimization Techniques Targeting Low Power VLSI Circuits," in *Design Automation, 1995. DAC '95. 32nd Conference on*, 1995, pp. 242-247.

[16] P. Landman, "High-level power estimation," in *Low Power Electronics and Design, 1996., International Symposium on*, 1996, pp. 29-35.

[17] N. Ranganathan, R. Namballa, and N. Hanchate, "CHESS: a comprehensive tool for CDFG extraction and synthesis of low power designs from VHDL," in *Emerging VLSI Technologies and Architectures, 2006. IEEE Computer Society Annual Symposium on*, 2006, p. 6 pp.

[18] S. Ahuja, A. Lakshminarayana, and S. K. Shukla, *Low Power Design with High-level Power Estimation and Power-aware Synthesis*: Springer New York, 2012.

[19] J. Bennett and K. Eder, "The Software Drained my Battery," in *National Microelectronics Institute: Yearbook 2011*, 2011, pp. 39-42.

[20] O. Neumann, "Challenges to the Low Power circuit design," ed: Mentor Graphics, 2009.

[21] A. Corporation, "Power consumption comparison: APEX 20K vs. Virtex devices,," ed. Altera Technical Brief, 1999.

[22] (24/04/2013). *Altera Corporation*. Available: http://www.altera.com/

[23] G. C. Cardarilli, A. Del Re, A. Nannarelli, and M. Re, "Power characterization of digital filters implemented on FPGA," in *Circuits and Systems, 2002. ISCAS 2002. IEEE International Symposium on*, 2002, pp. V-801-V-804 vol.5.

[24] L. Hyung Gyu, N. Sungyuep, and C. Naehyuck, "Cycle-accurate energy measurement and high-level energy characterization of FPGAs," in *Quality Electronic Design, 2003. Proceedings. Fourth International Symposium on*, 2003, pp. 267-272.

[25] D. Chih-Shun, T. Chi-ying, and M. Pedram, "Gate-level power estimation using tagged probabilistic simulation," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 17, pp. 1099-1107, 1998.

[26] L. Wang, M. Olbrich, E. Barke, T. Buchner, M. Buhler, and P. Panitz, "A theoretical probabilistic simulation framework for dynamic power estimation," in *Computer-Aided Design (ICCAD), 2011 IEEE/ACM International Conference on*, 2011, pp. 708-715.

[27] D. M. M. P. Radu Marculescu, "Efficient Power Estimation for Highly Correlated Input Streams," in *Design Automation, 1995. DAC '95. 32nd Conference on*, 1995, pp. 628-634.

[28] F. N. Najm, "Low-power design methodology: power estimation and optimization," in *Circuits and Systems, 1997. Proceedings of the 40th Midwest Symposium on*, 1997, pp. 1124-1129 vol.2.

[29] R. Burch, F. N. Najm, B. S. Ping Yang, and T. N. Trick, "A Monte Carlo approach for power estimation," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 1, pp. 63-71, 1993.

[30] T. Chi-ying, M. Pedram, and A. M. Despain, "Efficient estimation of dynamic power consumption under a real delay model," in *Computer-Aided Design, 1993. ICCAD-93.*

Digest of Technical Papers., 1993 IEEE/ACM International Conference on, 1993, pp. 224-228.

[31] G. I. Stamoulis and I. N. Hajj, "Improved Techniques for Probabilistic Simulation Including Signal Correlation Effects," in *Design Automation, 1993. 30th Conference on*, 1993, pp. 379-383.

[32] K. D. Muller-Glaser, K. Kirsch, and K. Neusinger, "Estimating essential design characteristics to support project planning for ASIC design management," in *Computer-Aided Design, 1991. ICCAD-91. Digest of Technical Papers., 1991 IEEE International Conference on*, 1991, pp. 148-151.

[33] XILINX. (29/04/2013). *Xilinx Power Estimator (XPE)*. Available: http://www.xilinx.com/products/design_tools/logic_design/xpe.htm

[34] ALTERA. (29/04/2013). *PowerPlay Early Power Estimators (EPE)*. Available: http://www.altera.com/support/devices/estimator/pow-powerplay.jsp

[35] M. Nemani and F. N. Najm, "High-level area and power estimation for VLSI circuits," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 18, pp. 697-713, 1999.

[36] D. Marculescu, R. Marculescu, and M. Pedram, "Information theoretic measures of energy consumption at register transfer level," presented at the Proceedings of the 1995 international symposium on Low power design, Dana Point, California, USA, 1995.

[37] V. Krishna and N. Ranganathan, "A methodology for high level power estimation and exploration," in *VLSI, 1998. Proceedings of the 8th Great Lakes Symposium on*, 1998, pp. 420-425.

[38] ACTEL. (30/04/2013). *SmartPower*. Available: http://www.actel.com/products/software/libero/smartpower.aspx

[39] SYNOPSYS. (30/04/2013). *Expanding the Synopsys PrimeTime Solution with Power Analysis*. Available: http://www.synopsys.com/Tools/Implementation/SignOff/CapsuleModule/ptpx_wp.pdf

[40] H. Lee, K. Lee, Y. Choi, and N. Chang, "Cycle-Accurate Energy Measurement and Characterization of FPGAs," *Analog Integrated Circuits and Signal Processing*, vol. 42, pp. 239-251, 2005.

[41] S. Gupta and F. N. Najm, "Power Macromodeling For High Level Power Estimation," in *Design Automation Conference, 1997. Proceedings of the 34th*, 1997, pp. 365-370.

[42] M. Anton, I. Colonescu, E. Macii, and M. Poncino, "Fast characterization of RTL power macromodels," in *Electronics, Circuits and Systems, 2001. ICECS 2001. The 8th IEEE International Conference on*, 2001, pp. 1591-1594 vol.3.

[43] G. Jochens, L. Kruse, E. Schmidt, and W. Nebel, "A new parameterizable power macro-model for datapath components," in *Design, Automation and Test in Europe Conference and Exhibition 1999. Proceedings*, 1999, pp. 29-36.

[44] F. Klein, G. Araujo, R. Azevedo, R. Leao, and L. C. V. dos Santos, "A multi-model power estimation engine for accuracy optimization," in *Low Power Electronics and Design (ISLPED), 2007 ACM/IEEE International Symposium on*, 2007, pp. 280-285.

[45] S. Dam, G. Y. Wu, N. A. Patil, and D. Chiou, "PrEsto: An FPGA-accelerated Power Estimation Methodology for Complex Systems," in *Field Programmable Logic and Applications (FPL), 2010 International Conference on*, 2010, pp. 310-317.

[46] F. Klein, G. Araujo, R. Azevedo, R. Leao, and L. C. V. dos Santos, "On the Limitations of Power Macromodeling Techniques," in *VLSI, 2007. ISVLSI '07. IEEE Computer Society Annual Symposium on*, 2007, pp. 395-400.

[47] L. Moss, "Profilage, caractérisation et partitionnement fonctionnel dans une plate-forme de conception de systèmes embarqués," Ph.D., École Polytechnique de Montréal, Montréal, Qc, Canada, 2010.

[48] V. Narayanan, L. Ing-Chao, and N. Dhanwada, "A power estimation methodology for systemC transaction level models," in *Hardware/Software Codesign and System Synthesis, 2005. CODES+ISSS '05. Third IEEE/ACM/IFIP International Conference on*, 2005, pp. 142-147.

[49] M. Caldari, M. Conti, M. Coppola, P. Crippa, S. Orcioni, L. Pieralisi, and C. Turchetti, "System-level power analysis methodology applied to the AMBA AHB bus [SoC applications]," in *Design, Automation and Test in Europe Conference and Exhibition, 2003*, 2003, pp. 32-37 suppl.

[50] F. Klein, G. Araujo, R. Azevedo, R. Leao, and L. C. V. dos Santos, "An efficient framework for high-level power exploration," in *Circuits and Systems, 2007. MWSCAS 2007. 50th Midwest Symposium on*, 2007, pp. 1046-1049.

[51] D. Greaves and M. Yasin, "TLM POWER3: Power estimation methodology for SystemC TLM 2.0," in *Specification and Design Languages (FDL), 2012 Forum on*, 2012, pp. 106-111.

[52] S. Ahuja, D. A. Mathaikutty, G. Singh, J. Stetzer, S. K. Shukla, and A. Dingankar, "Power estimation methodology for a high-level synthesis framework," presented at the Proceedings of the 2009 10th International Symposium on Quality of Electronic Design, 2009.

[53] F. Boussinot and R. De Simone, "The ESTEREL language," *Proceedings of the IEEE*, vol. 79, pp. 1293-1304, 1991.

[54] (14/05/2013). *Synopsys*. Available: http://www.synopsys.com/

[55] I. 1801™, "Unified Power Format (UPF 2.0) IEEE standard for design and verification of low power integrated circuits," ed, 2009.

[56] O. Mbarek, A. Pegatoquet, and M. Auguin, "A Methodology for Power-Aware Transaction-Level Models of Systems-on-Chip Using UPF Standard Concepts," in *Integrated Circuit and System Design. Power and Timing Modeling, Optimization, and Simulation*. vol. 6951, J. Ayala, B. García-Cámara, M. Prieto, M. Ruggiero, and G. Sicard, Eds., ed: Springer Berlin Heidelberg, 2011, pp. 226-236.

[57] J. Coburn, S. Ravi, and A. Raghunathan, "Power emulation: a new paradigm for power estimation," in *Design Automation Conference, 2005. Proceedings. 42nd*, 2005, pp. 700-705.

[58] M. A. Ghodrat, K. Lahiri, and A. Raghunathan, "Accelerating System-on-Chip Power Analysis Using Hybrid Power Estimation," in *Design Automation Conference, 2007. DAC '07. 44th ACM/IEEE*, 2007, pp. 883-886.

[59] A. Bhattacharjee, G. Contreras, and M. Martonosi, "Full-system chip multiprocessor power evaluations using FPGA-based emulation," in *Low Power Electronics and Design (ISLPED), 2008 ACM/IEEE International Symposium on*, 2008, pp. 335-340.

[60] G. Qu, N. Kawabe, K. Usami, and M. Potkonjak, "Function-level power estimation methodology for microprocessors," presented at the Proceedings of the 37th Annual Design Automation Conference, Los Angeles, California, USA, 2000.

[61] M. Ibrahim, M. Rupp, and A. H. Fahmy, "Power estimation methodology for VLIW Digital Signal Processors," in *Signals, Systems and Computers, 2008 42nd Asilomar Conference on*, 2008, pp. 1840-1844.

[62] (21/05/2013). *Texas Instruments*. Available: http://www.ti.com/

[63] S. K. Rethinagiri, R. Ben Atitallah, S. Niar, E. Senn, and J. Dekeyser, "Fast and accurate hybrid power estimation methodology for embedded systems," in *Design and Architectures for Signal and Image Processing (DASIP), 2011 Conference on*, 2011, pp. 1-7.

[64] J. Laurent, N. Julien, E. Senn, and E. Martin, "Functional level power analysis: an efficient approach for modeling the power consumption of complex processors," in *Design, Automation and Test in Europe Conference and Exhibition, 2004. Proceedings*, 2004, pp. 666-667 Vol.1.

[65] V. Tiwari, S. Malik, A. Wolfe, and M. T. C. Lee, "Instruction level power analysis and optimization of software," in *VLSI Design, 1996. Proceedings., Ninth International Conference on*, 1996, pp. 326-328.

[66] S. Nikolaidis and T. Laopoulos, "Instruction-level power consumption estimation embedded processors low-power applications," in *Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, International Workshop on, 2001.*, 2001, pp. 139-142.

[67] V. Tiwari, S. Malik, and A. Wolfe, "Power Analysis Of Embedded Software: A First Step Towards Software Power Minimization," in *Computer-Aided Design, 1994., IEEE/ACM International Conference on*, 1994, pp. 384-390.

[68] M. Wendt, M. Grumer, C. Steger, R. Weiss, U. Neffe, and A. Muhlberger, "Energy Consumption Measurement Technique for Automatic Instruction Set Characterization of Embedded Processors," in *Instrumentation and Measurement Technology Conference Proceedings, 2007. IMTC 2007. IEEE*, 2007, pp. 1-4.

[69] M. Rogers-Vallee, M. A. Cantin, L. Moss, and G. Bois, "IP characterization methodology for fast and accurate power consumption estimation at transactional level model," in *Computer Design (ICCD), 2010 IEEE International Conference on*, 2010, pp. 534-541.

[70] B. Klass, D. E. Thomas, H. Schmit, and D. E. Nagle, "Modeling Inter-Instruction Energy Effects in a Digital Signal Processor," presented at the Power-Driven Microarchitecture

Workshop, in conjunction with Intl. Symposium on Computer Architecture, Barcelona, Spain, 1998.

[71] S. Lee, A. Ermedahl, S. L. Min, and N. Chang, "An Accurate Instruction-Level Energy Consumption Model for Embedded RISC Processors," presented at the Proceedings of the ACM SIGPLAN workshop on Languages, compilers and tools for embedded systems, Snow Bird, Utah, USA, 2001.

[72] N. Kavvadias, P. Neofotistos, S. Nikolaidis, K. Kosmatopoulos, and T. Laopoulos, "Measurements analysis of the software-related power consumption in microprocessors," in *Instrumentation and Measurement Technology Conference, 2003. IMTC '03. Proceedings of the 20th IEEE*, 2003, pp. 981-986 vol.2.

[73] H. Blume, M. Schneider, and T. G. Noll, "Power estimation on functional level for programmable processors," in *Advances in Radio Science*, ed, 2004.

[74] A. R. Tzan Hsin Ma J. , "Estimativa de Consumo de Energia em Nível de Instrução para Processadores Modelados em ArchC," presented at the X Simpósio em Sistemas Computacionai, 2009.

[75] P. Zipf, H. Hinkelmann, D. Lei, M. Glesner, H. Blume, and T. G. Noll, "A Power Estimation Model for an FPGA-Based Softcore Processor," in *Field Programmable Logic and Applications, 2007. FPL 2007. International Conference on*, 2007, pp. 171-176.

[76] F. Yunsi, S. Ravi, A. Raghunathan, and N. K. Jha, "A hybrid energy-estimation technique for extensible processors," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 23, pp. 652-664, 2004.

[77] (21/05/2013). *Tensilica*. Available: http://www.tensilica.com/

[78] XILINX. (05/02/2013). *Platform Studio and the Embedded Development Kit (EDK)*. Available: http://www.xilinx.com/tools/platform.htm

[79] A. Gaisler. (01/02/2013). *Leon3 Processor*. Available: http://www.gaisler.com/index.php/products/processors/leon3?task=view&id=13

[80] XILINX. (10/02/2013). *ISE 13.4*. Available: http://www.xilinx.com/support/documentation/dt_ise13-4.htm

[81] L. Wang, M. French, A. Davoodi, and D. Agarwal, "FPGA dynamic power minimization through placement and routing constraints," *EURASIP J. Embedded Syst.*, vol. 2006, pp. 7-7, 2006.

[82] ARM. (25/02/2013). *ARM7 Processor Family*. Available: http://www.arm.com/products/processors/classic/arm7/index.php

[83] I. f. C. T. a. E. S. (ICE). (17/04/2013). *LISA - Description Language for Hardware and Software*. Available: http://www.ice.rwth-aachen.de/research/tools-projects/lisa/lisa

[84] Accellera. (15/04/2013). *Accellera and the Open SystemC Initiative (OSCI)* Available: http://www.accellera.org/

[85] EEMBC. (05/03/2013). *CoreMark*. Available: http://www.coremark.org/

[86] J. T. H. Ma, "Estimativa de Consumo de Energia em Nível de Instrução para Processadores Modelados em ArchC," M.Sc.A., UNICAMP, Campinas, São Paulo, Brasil, 2007.

[87] U. o. Michigan. (11/03/2013). *MiBench Version 1.0*. Available: http://www.eecs.umich.edu/mibench/

[88] E. Canada. (12/03/2013). *Erreur quadratique moyenne*. Available: http://www.meteo.gc.ca/verification/scores/rmse_f.html

[89] G. Beltrame, C. Brandolese, W. Fornaciari, F. Salice, D. Sciuto, and V. Trianni, "Dynamic modeling of inter-instruction effects for execution time estimation," in *System Synthesis, 2001. Proceedings. The 14th International Symposium on*, 2001, pp. 136-141.

[90] D. Elleouet, N. Julien, D. Houzet, J.-G. Cousin, and E. Martin, "Power Consumption Characterization and Modeling of Embedded Memories in XILINX VIRTEX 400E FPGA," presented at the Proceedings of the Digital System Design, EUROMICRO Systems, 2004.