Grenoble INP – ENSIMAG

École Nationale Supérieure dInformatique et de Mathématiques Appliquées

Rapport de projet de fin d'études

Effectué chez Thales et au Lip6

Métriques de consommation pour l'évaluation de l'empreinte écologique d'un programme sur un micro-contrôleur

Laramas Rémi
3e année – Option ISI

17 février 2025 – 15 août 2025

Thales SIX GTS et Lip6
4 avenue des Louveresses et 4 place Jussieu
92230 Gennevilliers et 75252 PARIS CEDEX 05

Responsable de stage
Lounes pour Thales et
Cassagne Adrien pour le Lip6
Tuteur de l'école
Denis Trystram

Contents

Glossaire

AID Axe Intelligence artificielle et science des données de recherche du Lip6. 6

ALSOC Architecture et Logiciels pour Systèmes embarqués sur Puce[1]. Équipe se concentrant sur la conception des systèmes multiprocesseurs intégrés sur puce. 6

ASN Axe Architecture, système et réseaux de recherche du Lip6. 6

AtomMan x7ti Processeur utilisé pour les expérimentations, doté de cœurs x8[2]. 5, 44

CodeCarbon Outil sous licence MIT qui calcule l'émission de CO2eq d'une application[3]. 57, 58

EnergAt Outil sous licence MIT d'analyse de consommation énergétique[4]. 12, 13, 14

i.mx93 Processeur produit par NXP utilisé pour les expérimentations[5]. 5, 44

IEA L'International Energy Agency, mentionnée pour les données globales d'impact carbone. 57

ISO14050 Une des normes suivies par Thales dans la promotion de l'écologie au sein de Thales. 9

Lip6 Le laboratoire d'informatique de Paris 6 (Lip6) est une unité de recherche de Sorbonne Université spécialisée dans les sciences informatiques. 5, 6

McPAT Outil de modélisation utilisée pour estimer la consommation énergétique et autres métriques d'un processeur[6]. 10, 11, 12

NUMA Non-Uniform Memory Access, architecture mémoire étudiée par EnergAt. 12, 14

SSR Axe Sécurité, sûreté et fiabilité de recherche du Lip6. 6

TDP Thermal Design Power, mesure la quantité de chaleur dégagée par un processeur au cours de son utilisation. 58

TMC Axe Théorie et outils mathématiques de recherche du Lip6. 6

[1] *Page web de l'équipe ALSOC.*
[2] *Page web de description de l'AtomMan disponible au Lip6.*
[3] *Dépôt de codecarbon.*
[4] *Dépôt de EnergAt.*
[5] *Page web de la carte d'évaluation de l'i.mx93.*
[6] Li et al., "McPAT".

Part I
Introduction

1 État final du projet

Mots clés : Micro-contrôleur, impact énergétique, analyse consommation dynamique
Ce rapport présente le travail effectué durant mon stage de recherche conjoint entre le laboratoire Lip6 de Sorbonne Université et Thales. Le projet vise à instaurer des **métriques pour évaluer les performances de l'exécution d'une application sur un microcontrôleur**, dans une démarche d'écoconception.

Le contexte de recherche est divisé en deux environnements, celui du Lip6 et celui de Thales, chacun apportant des perspectives et des ressources distinctes. Le défi réside dans la difficulté de mesurer avec précision l'impact carbone de l'exécution de programmes sur différentes architectures de processeurs. Deux cartes, la i.mx93 et la AtomMan x7ti, ont été mises à ma disposition pour réaliser des tests et expérimentations.

Un modèle théorique pour répondre à la problématique est proposé mais n'a pas encore été testé. Des tests ont été conduits pour comprendre les systèmes étudiés et l'implication des différentes optimisations implantées sur la consommation de ces cartes.

2 Résumé des réalisations

Nous nous sommes attelés en premiers lieux à réaliser un état de l'art des méthodes d'estimation de consommation existantes dans la littérature académique, dont nous résumons les grandes lignes partie IV. Cela nous a permis de balancer nos choix et avoir un premier aperçu de ce qui était possible dans ce domaine. Nous n'en parlons pas ici, mais une partie de ce travail de recherche s'est aussi porté sur les outils de modélisation de la mémoire à ces fins de prédiction. Au vu de la difficulté grandissante du problème au fur et à mesure de nos avancement, nous avons décidé de laisser cette partie de côté. Nous revenons sur la chronologie de ces choix en partie VIII.

Il a fallu ensuite prendre en main les cartes qui m'étaient proposées et leurs environnement, travail que nous abordons section 4. L'hétérogénéité de ces plateformes fut une force comme un frein et nous a permis notamment comprendre la difficulté de la construction de modèles prédictifs de consommation.

Nous sommes rapidement tombés sur de nouveaux défis techniques en essayant d'instrumenter ces cartes : quels outils utiliser ? Y en a-t-il qu'il faut privilégier ou non sur le long terme ? De quelles données avons-nous besoin ? Nous revenons sur la sélection que nous fait partie V. Ce travail a résulté en la construction de plusieurs scripts de déploiement et de traitement de données basés sur ces outils, sur lesquels nous revenons rapidement dans la section 13.

S'inspirant des méthodes de la littérature, et utilisant les données produites par nos outils, nous avons finalement mis sur pied une méthode d'estimation de consommation sur laquelle nous revenons en partie VI.

Enfin, tout au long de ce stage nous avons émis des hypothèses et les avons affirmées ou infirmées concernant les postes responsables de la consommation des programmes, ce sur quoi nous revenons en partie VII. Les points à retenir quant à ce travail de recherche et de développement sont résumés en partie X. Nous y développons aussi nos idées et les pistes futurs que nous voudrions explorer dans la suite en thèse de ce sujet.

Enfin, nous avons réalisé une analyse post stage de ce qu'il aura coûté à notre chère planète, et plus globalement à la société. L'analyse - développée en partie IX - est ici intéressante, car comme nous le montrons en partie III suivante, ce stage prend origine dans le mouvement d'écoconception qui se monte au sein de Thales. Cela permet de remettre en contexte, si ce n'est répondre, à la question : est-il "éco-responsable" de travailler sur l'éco responsabilité ?

Part II
Contexte du stage

Dans cette partie nous abordons deux aspects de mon environnement de travail. Le premier correspondant au lieux, où et avec qui ai-je travaillé, et le second correspondant aux matériels mis à disposition sur chacun de ces lieux.

3 Environnement de travail

Mon stage est un stage de recherche conjoint entre le Lip6 et Thales. De ce fait, j'ai la particularité d'avoir 2 lieux de travail, donc 2 équipes et 2 contextes distincts pour développer le projet de ce stage.

Lip6 Le laboratoire d'informatique de Paris 6 (Lip6) est une unité de recherche de Sorbonne Université composée d'environ 500 personnes, spécialisée dans les sciences informatiques. Elle est décomposée en 4 axes de recherche :

- AID - Intelligence artificielle et science des données
- ASN - Architecture, système et réseaux
- SSR - Sécurité, sûreté et fiabilité
- TMC - Théorie et outils mathématiques

L'équipe ALSOC, dont je fais partie, se concentre sur la conception des systèmes multiprocesseurs intégrés sur puce, particulièrement les manycores, afin de répondre aux besoins de performance des applications embarquées telles que le traitement de flux vidéo et multimédia, et le traitement de paquets dans les télécoms. Elle développe des méthodes de conception conjointes matériel-logiciel, abordant des aspects comme l'architecture matérielle, les protocoles de communication, les systèmes d'exploitation embarqués, la vérification formelle, le test post-fabrication, et la génération de code optimisé pour diverses architectures. Elle supporte des recherches qui appartiennent aux domaines ASN, AID et SSR.

Thales Thales est, quant à elle, une entrerprise à échelle internationale, comprenant pas moins de 77000 employés à travers le monde et proposant des solutions sur des domaines tels que : l'aéronautique, les télécommunications, l'espace, la sécurité, la défense ou encore les modes de transports. Nous ne rentrerons pas dans les détails de l'organisation de cette entreprise, car ce n'est pas l'objectif de ce rapport. Cependant, contextualisons un peu le cadre dans lequel se déroule mon stage.

Thales est une entreprise organisée en unités globales de gestion des projets qui ont chacune leur domaine d'expertise.

Mon équipe fait partie d'une de ces grandes entités et travaille sur des drivers GPP (General Purpose Processor) et DSP (Digital Signal Processor), l'industrialisation de systèmes, la sécurisation et le durcissement d'OS et la virtualisation.

4 Environnement de développement

Le Monolithe

Le Monolithe est un projet maintenu notamment par l'un de mes encadrant, Adrien Cassagne, qui met à disposition aux chercheurs et stagiaires de l'équipe un ensemble de systèmes sur puces avec architectures hétérogènes. L'objectif est de fournir à ces cartes une interface homogène, simple d'utilisation et d'expérimentation.

En réalité placé dans une armoire, le monolithe a une architecture similaire à celle de la plateforme grid5000 à échelle réduite. Une machine Dell OptiPlex d'accès aux cartes est mise à disposition à

tout utilisateur possédant un compte en ssh. Elle détient une installation de l'outil slurm permettant de réserver des ressources sur les nœuds de la topologie qui y est connectée, ainsi, tout nœud étant connecté au réseau local de la machine Dell (nommée "front") peut-être accessible par ssh après réservation.

Cette plateforme contient notamment le nœud "x7ti" que nous décrirons dans la prochaine partie. Une partie du travail réalisé pendant ce stage a donc été de créer des scripts de déploiement sur le Monolithe afin de pouvoir lancer automatiquement des tests et instrumenter les programmes testés.

l'AtomMan X7TI

L'AtomMan X7TI est un SOB (system on board) avec un processeur Intel de génération MeteorLake. Ce processeur, l'Intel Core Ultra 9 185H, contient 22 coeurs logiques de différentes catégories : P-core,E-core ou LPE-core. Les P-cores (Performance Cores) sont optimisés pour les tâches exigeantes en termes de calculs et offrent une haute performance monocœur, sur ce processeur ils sont à la version "Redwood" de la micro-architecture. Les E-cores (Efficient Cores) sont conçus pour gérer des tâches moins exigeantes et sont capables de gérer plusieurs threads simultanément. Ils sont à mi-chemin en termes de performance entre un P-Core et un LPE-Core. Sur ce processeur, l'architecture est à la version Crestmont. Enfin, les LPE-cores (Low Power Efficient Cores) sont optimisés pour les tâches à faible consommation d'énergie, ils sont moins rapide en fréquence, ont accès à moins de ressources mémoire, mais permettent une meilleure consommation.

Pour illustrer notre propos, la figure 27 est disponible en annexe et reprends l'architecture globale de ce processeur.

L'efficacité des cœurs dépends grandement de leurs fréquences de fonctionnement, que nous résumons dans la table ci-dessous :

	Fréquence maximale sans Turbo Boost	Fréquence maximale avec Turbo Boost
P-core	2,3 Ghz	5,1 Ghz
E-core	1,8 Ghz	3,8 Ghz
LPE-core	1 Ghz	2,5 Ghz

La technologie Turbo Boost d'Intel permet au processeur d'augmenter dynamiquement sa fréquence d'horloge pour répondre à une demande de performance élevée. Cela signifie que lorsque certaines conditions sont remplies, notamment en termes de puissance et de température, les cœurs peuvent fonctionner à des fréquences supérieures à leur fréquence de base pour offrir une performance accrue. Dans les faits, cette technologie est plus généralement appelée DVFS (Dynamic Voltage and Frequency Scaling) et permet, via un petit composant matériel supplémentaire, de faire varier la fréquence et la tension d'alimentation de la puce. Nous reviendrons sur son impact en section 15.

Cette carte a été choisie car elle présente une diversité dans les cœurs dont elle dispose. Cela n'a pas été abordé ici, mais elle possède aussi une carte GPU embarquée de manufacture Intel, qui permettrait aussi à l'avenir de pouvoir étendre les travaux à ces architectures particulières de puces.

Enfin le processeur est sous architecture x86_64 ce qui permet de contrebalancer avec l'architecture ARM de la carte disponible chez Thales.

l'I.MX93

L'i.MX93 est un processeur fournit par NXP qui embarque 2 cœurs ARM A55 d'une fréquence nominale de 1,7 Ghz. Elle embarque aussi un cœur M33 destiné à des usages de temps réel. La carte mise à ma disposition embarquant ce processeur comprenait un ensemble varié de composant supplémentaires : interfaces ethernet, wifi, bluetooth, lecteur de carte SD, NPU, etc.

Cette carte, l'i.MX93evk, a notamment permis l'utilisation du logiciel "BCU" que nous aborderons plus tard dans ce document. Une vue d'ensemble de ce que contient cette carte est affiché dans la figure 28 disponible en annexe. Nous ne rentrerons pas dans les détails de ce qu'elle contient, car ce n'est pas le sujet de ce rapport et que la majorité des composants exposés n'ont pas été manipulés pendant ce stage.

L'i.MX93 a été choisie car NXP, son fournisseur, est un choix privilégié dans les processus de Thales. De plus, elle est très bien instrumentée en termes de compteurs physique pour mesurer la consommation, ce que nous avons l'occasion d'aborder dans la partie 8.

Part III
Problématique du stage

Mesurer l'impact environnemental d'un logiciel peut passer par plusieurs étapes. Aujourd'hui, des normes comme l'ISO14040 et l'ISO14044[7] propose des règles pour encadrer une pratique nommée l'analyse du cycle de vie, qui a pour but de mesurer et d'agir sur les différents aspects de consommation d'un système (pas seulement numérique donc). L'ARCEP (autorité de régulation des communications électroniques, des postes et de la distribution de la presse) a mis sur pied un guide contenant un système de notation et d'analyse par critère pour les systèmes numériques : le Référentiel Général de l'écoconception des services numériques (RGESN)[8]. Bien que ces initiatives permettent de mettre sur pied des outils accessibles au plus grand nombre et de répandre de nouvelles tendances plus vertueuses dans le développement applicatif, ils restent des outils d'analyse empirique. Pour notre cas d'étude, ces propositions sont trop proches de l'utilisateur final pour être utiles.

Si nous prenons une mesure classique de l'impact environnementale d'un système quelconque, l'impact carbone, nous pouvons raisonnablement nous poser la question de la proportion, pour un outil numérique, de cet impact entre la phase de production et celle de l'utilisation de ce système. Malmodin and Lundén, "The Energy and Carbon Footprint of the Global ICT and E&M Sectors 2010–2015" propose une analyse de cette proportion partie 5.1, figure 13 et partie 4.10, figure 10. Ces figures montrent une importance certaine à l'analyse de la consommation dynamique (consommation à l'utilisation). Cependant, la part liée à la production n'est pas négligeable. Ce stage n'abordera cependant pas cette dernière : nous avons décidé d'évoluer à architectures fixées. Cela permet de réduire les inconnues à considérer lors de l'étude ainsi que d'être en meilleure adéquation avec le fonctionnement interne de Thales. De futurs travaux pourront être envisagés à cet effet afin d'avoir une analyse plus complète.

Enfin, une partie de la recherche scientifique se concentre à établir un lien entre l'impact carbone et l'exécution d'un programme (comme celui présenté en section 22). Cependant, ce lien se base étroitement sur l'impact énergétique des programmes étudiés et présente des incertitudes quand utilisé avec des charges courtes - des exécutions de programmes courtes.

Chez Thales, lors de la construction d'un nouveau projet, impliquant des dimensions de technologies embarquées, plusieurs corps de compétences s'attèlent à répondre au problème. Traditionnellement, une équipe s'occupe de la partie physique, et une autre (composée de certains membres de la mienne) se charge des problèmes de liaison entre l'application finale du projet et l'interfaçage avec le physique.

D'un point de vue physique, comparer les performances des architectures à utiliser pour un problème donné revient à comparer des indicateurs comme : la surface de la puce, sa rapidité, la puissance dégagée en fonction de sa fréquence et l'énergie consommée au global par celle-ci. Cependant, ces indicateurs sont des indicateurs purement physiques, et sont donc des choix indépendants du logiciel - ou presque, considérant que les équipes entretiennent tout de même des discussions sur ces sujets.

Une brique est manquante à la frontière entre les deux mondes : comment utiliser au mieux une puce donnée d'un point de vue énergétique, peu importe la puce ? Comment, en fonction du profil énergétique d'une application donnée, choisir la carte à prendre pour un projet donné ?

Le point de départ pour l'ébauche d'une réponse à ce problème est la mise en place d'une mesure universelle de dépense énergétique, ou de profil de dépense énergétique, pour une application sur

[7] *Page Wikipedia de l'Analyse du cylce de vie.*
[8] *Page web de la RGESN.*

une carte donnée. C'est instaurer une échelle pour pouvoir comparer plusieurs versions d'un même algorithme sur une carte, ou plusieurs cartes pour la même version d'un algorithme.

C'est pourquoi nous sommes arrivés à formuler la problématique suivante :

> "Mise en place d'une méthode pour l'évaluation des performances de l'exécution d'une application sur un micro-contrôleur"

Elle s'inscrit dans une stratégie plus globale de Thales qui vise à développer ce que la norme ISO14050 décrit comme de l'éco-design au sein du groupe: "approche visant à appliquer des méthodes et processus d'ingénierie dans le but délibéré de réduire les impacts négatifs d'une solution sur l'environnement". Mon stage est la traduction de cette stratégie dans le domaine de l'informatique embarquée.

Mon stage a pour but de trouver/développer une méthode fine répondant à ce problème, puis de comparer ces résultats avec des évaluations académiquement reconnues.

Part IV
Méthodes existantes pour évaluer la consommation

Dans cette partie nous décrirons un aperçu des méthodes existantes pour répondre à la problématique. Dans la littérature, 2 grandes catégories de méthodes d'estimation sont prédominantes : par analyse physique et par estimation statistique.

La première catégorie se base sur une représentation plus ou moins précise des postes de consommation de la carte par analyse physique (nombre de lignes de cache par cache, nombre cœurs, nombre d'ALU, etc) et propose un modèle qui infèrent ces paramètres en utilisant les équations de consommation liées à la physique des matériaux. Une fois cette représentation de carte établie, les applications cible tournent sur des simulateurs afin d'en récupérer la façon dont elles influencent les cartes via un ensemble de paramètres. Ceux-ci sont ensuite entrés dans le premier modèle pour en estimer la puissance/énergie finale.

La seconde propose des méthodes avec une approche orthogonale à la première. Les méthodes partent du système entier en boîte noire et considèrent un ensemble de paramètres influencé par l'exécution de l'application cible (nombre de cache hit en fonction du niveau de cache par exemple). Elles exécutent un ensemble de programme de taille et nature variable en fonction des papiers afin de générer une base de données contenant ces paramètres et l'énergie/la puissance totale dépensée mesurée pour achever l'exécution des programmes. Elles proposent ensuite des modèles plus ou moins complexes pour décrire le fonctionnement de ces cartes en utilisant des méthodes de régression sur les données enregistrées.

5 Modèles analytiques

Premiers modèles que nous avons explorés lors de ce stage, les modèles analytiques qui suivent présentent un avantage non négligeable comparé aux suivants : ils permettent facilement d'expliquer la consommation. De ce fait, maîtriser de tels outils garantiraient, nous pensions, d'en implanter d'autres conseillant sur les optimisations à implanter pour améliorer l'empreinte énergétique des programmes étudiés.

Nous présentons dans cette section 2 outils : McPAT et EnergAt. Là où McPAT se concentre à simuler la puce utilisée pour faire exécuter le programme cible, EnergAt s'attrait à attribuer la bonne portion de temps processeur à chaque thread/process dans le contexte d'une exécution dans un environnement partagé, comme un serveur.

McPat

Description

Li et al., “McPAT” modélise le comportement d’un processeur sur plusieurs niveaux d’abstraction afin d’en tirer des données du point de vue consommation électrique, efficacité temporelle, taille de la puce.

Il prend en charge un ensemble très large d’options architecturales et micro-architecturales. C’est un logiciel en invite de commande .L’un de ses atouts est qu’il vise à réduire le nombre d’entrées de l’utilisateur en utilisant un optimiseur qui s’améliore de façon cyclique.

Le logiciel permet de faire des projections sur des processeurs encore inexistants. Le papier présente un cas pratique, et arrive à tirer des performances de configurations inexistantes au jour où elles étaient simulées. Afin de produire des résultats en fonction du programme passé en argument, McPAT utilise une variété de simulateurs (bloc ”Runtime Power Stats” dans la figure 1). Cependant, il n’est lié à aucun d’entre eux, ce qui le rend flexible.

Considérant la taille de l’outil, nous nous restreindrons à décrire la partie concernant ses estimations de consommation énergétique.

Méthodologie

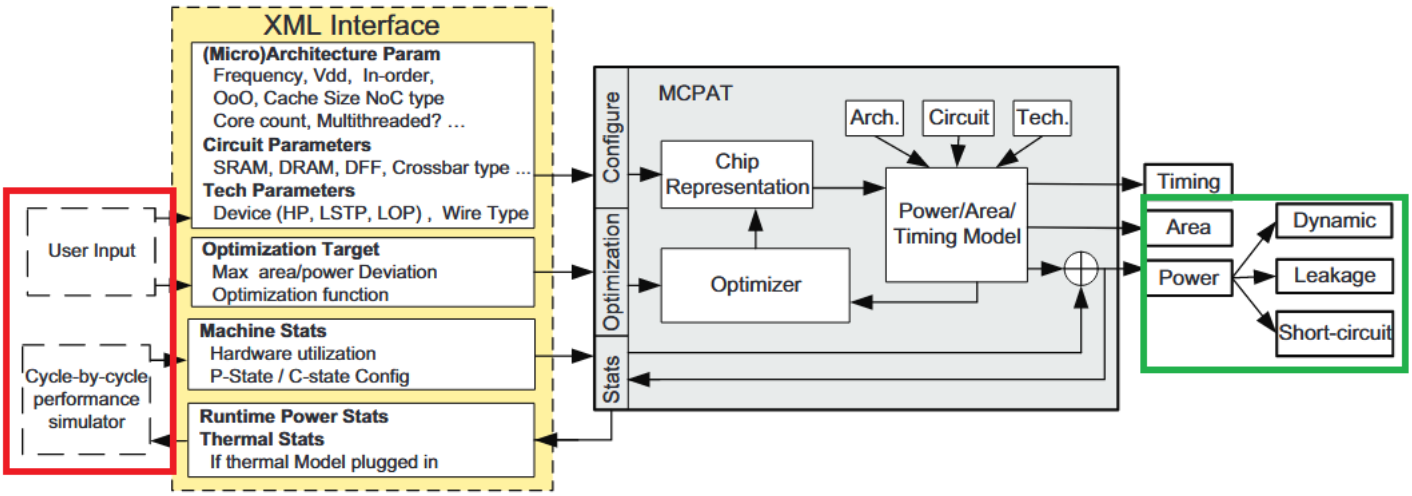


Figure 1: Diagramme Bloc du fonctionnement de McPat

La figure 1 ci-dessus représente le fonctionnement global de l’outil. En entrée, McPat attend une configuration au format XML des différentes options architecturales, micro-architecturales, de composition et de technologie du processeur étudié. Le papier ne liste pas l’ensemble de ces paramètres, mais des exemples de configurations de McPat pour certains processeurs sont disponibles sur le dépôt git du projet[9]. Nous ne rentrerons pas dans les détails du fonctionnement de cet outil ici, mais ils sont abordés dans le papier décrivant l’outil, Li et al., “McPAT”.

Une fois en possession de ces paramètres, la représentation de la puce est créé selon un procédé hiérarchique, illustré dans la figure 2. On peut distinguer trois niveaux : architectural, circuit et technologique.

Modélisation Pour le niveau architectural, McPAT modélise les composants suivants :

- **Core** : Le travail de modélisation effectué par McPAT se base sur le modèle existant de Palacharla, Jouppi, and Smith, “Complexity-Effective Superscalar Processors”. Ici, McPAT modélise les différentes unités classiques d’un cœur (Instruction Fetch Unit, Execution Unit,

[9] *Dépôt de McPat.*

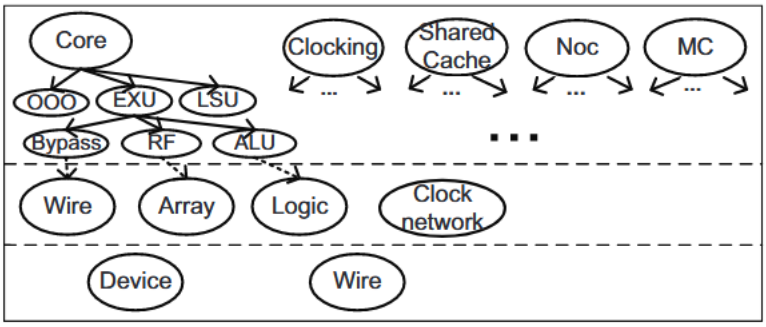


Figure 2: Diagramme de démonstration du fonctionnement hiérarchique de l'outil McPat

Load-Store Unite, Issue/Dispatch Unit, etc) de façon "grossière" en s'arrêtant à l'ALU (Arithmetic Logic Unit) en termes de finesse d'estimation. Les processeurs multi-thread sont supportés, car il les considère comme un réseau de cœurs intégré, en dupliquant donc le hardware associé et en ajoutant une structure de communication entre eux. Cette structure a été modélisée en se basant sur les architectures multi-cœurs des processeurs Niagara de l'époque, sur la technologie "Hyperthreading" d'Intel et sur certaines recherches dans ce domaine.

- **NoC (Network on Chip)** : Pour ces réseaux, McPAT modélise 2 composants : les fils et les routeurs. Ces modélisations sont propres à McPAT. Elles utilisent cependant le même procédé : les routeurs sont découpés en blocs de base qui sont ensuite modélisés par McPAT.
- **Caches intégrés** : En fonction des paramètres entrés, ils peuvent être modélisés soit comme une structure CAM (Content Adressable Memory) soit comme un cache classique. McPAT supporte les caches privés et partagés.
- **Contrôleurs Mémoire** : C'est le premier outil à proposer la modélisation de ces composants. Les auteurs se base sur un design produit par Denali, "Using Configurable Memory Controller Design IP with Encounter RTL Complier" pour la réaliser. La mise à l'échelle est faite de manière empirique, en se basant sur les données publiées par AMD et Rambus.
- **Systèmes Horloges** : Il est composé de la modélisation de 2 parties : les PLLs (phased-locked loop) et le réseau de communication des horloges. La mise à l'échelle de ce réseau se base sur des valeurs empiriques données par Intel et Sun.

Pour le niveau circuit, nous ne rentrerons pas dans les détails de modélisation, mais McPAT modélise les composants suivants :

- Les câbles
- Les tableaux
- La logique
- Le réseau de distribution d'horloge

Les modélisations de ce niveau sont grandement basées sur Wilton and Jouppi, "CACTI" - les tableaux entièrement, et la modélisation de la logique utilise la même approche algorithmique.

Enfin, pour le niveau technologique, McPAT se fonde sur Association, *Model for Assessment of CMOS Technologies and Roadmaps (MASTAR)* pour dériver les paramètres provenant des projections de l'ITRS.

Estimations de consommation Une fois ces étapes de construction du modèle du processeur renseigné, McPAT réalise des estimations de consommation de l'énergie pour un programme donné. Comme montré dans le rectangle vert de la figure 2, elles sont divisées en trois parties : consommation dynamique, consommation de fuite et consommation de court-circuit.

La consommation dynamique est proportionnelle à : la charge totale des condensateurs du circuit, la tension de l'alimentation, la fréquence d'horloge, la variation de tension pendant une transition d'état et un facteur d'activité. Ce dernier facteur est estimé en fonction de statistiques sur la simulation architecturale et les propriétés du circuit.

La consommation de fuite est estimée en fonction de la taille des transistors et de l'état de la puce. Elle provient de 2 sources : la fuite sous-seuil, et la fuite de porte. La première arrive car un petit courant passe à travers les transistors en état "off" et la seconde car un courant fuite par les terminaux des portes logiques.

Enfin, la consommation de court-circuit représente la fuite d'énergie présente lorsque les portes logiques changent d'état et créent momentanément un court-circuit. Selon le papier[10], il peut représenter jusqu'à 25% de la consommation dynamique du circuit.

Une fois ces estimations faites, McPAT utilise un optimiseur qui cherche la solution optimale en termes d'énergie dépensée par le programme donné et de place prise par la carte, comme montré dans le rectangle rouge de la figure 2. Il va aussi essayer de maximiser la fréquence d'horloge, mais s'autorisera à ne pas sélectionner les solutions qui proposent cela si elles ne parviennent pas à optimiser les 2 premiers paramètres.

Conclusion

McPAT est un outil complet. Il se base sur les standards de son époque (Wilton and Jouppi, "CACTI") et les porte à un niveau qui permet une exploitation dans des cadres de recherches industrielles. Le papier propose une partie validation dans laquelle il compare ses estimations à 4 processeurs : le Niagara, le Niagara 2, l'Alpha 21364 et le Xeon Tulsa. Ces estimations sont des consommations pics, et non moyennes. Par exemple, pour le processeur Niagara, l'erreur moyenne de prédiction par composant est de 23% (les contributeurs majeurs étant la puissance liée à l'horloge et la puissance de fuite) pour une différence absolue de 1.47W (10%). L'erreur totale augmente pour les autres processeurs, ce que les auteurs expliquent notamment par le fait que McPAT ne modélise pas la consommation des entrées/sorties du processeur (Ethernet, PCI, RAM externe) et que certains processeurs ne dévoilent pas leur conception précise.

McPAT a, depuis, profité de plusieurs améliorations et certaines versions sont plus adaptées à nos technologies actuelles.

EnergAt

Version 1.0.6[11]

Description

EnergAt est un outil sous licence MIT d'analyse de consommation énergétique. Il ne fait pas le lien direct avec l'impact carbone de l'énergie consommée, mais a une granularité fine dans ses mesures. Le logiciel tourne autour des applications multi-tenants, utilisées dans des serveurs par exemple. Le principal objectif de l'outil est de proposer une mesure fiable de la consommation des applications parallélisées. L'outil ne traque, cependant, que la consommation du CPU et des mémoires NUMA privées, omettant donc les GPUs et autres types de mémoires partagées. Il ne propose pas non plus d'analyse de consommation sur ses entrées/sorties.

Méthodologie

EnergAt se veut le moins impactant possible sur les performances de la machine étudiée, tout en restant précis dans ses prises de mesure. Pour ce faire, le logiciel utilise un démon qui s'attèle à prendre les mesures liées à l'application étudiée, et inspecte aussi sa propre consommation afin de l'écarter de son rapport final.

[10] Li et al., "McPAT".
[11] *Dépôt de EnergAt.*

Posons les variables suivantes pour la suite de l'analyse :

$$\begin{aligned} A &:= \text{"Ensemble d'actions représentant l'application"};\\ a \in A &:= \text{"une action effectuée dans le cadre d'une application"};\\ S &:= \text{"Ensemble des sockets matériels de la machine"};\\ s_i \in S &:= \text{"Socket matérielle i de la machine"};\\ Th_i &:= \text{"Ensemble des threads ou processus d'un socket i"};\\ t_j \in Th_i &:= \text{"Thread ou processus j du socket i de la machine"}; \end{aligned}$$

Dans le présent rapport, et le papier étudié, un "socket matériel" est un un emplacement sur une carte qui peut accueillir une puce de calcul principal. Cette puce peut elle-même être composée de coeurs plus ou moins autonomes dans leur gestion.

Afin de simplifier le propos, nous allons nous appuyer sur la figure 3 suivante, représentant une machine à 3 sockets matérielles où 2 threads tournent en parallèle sur chaque socket.

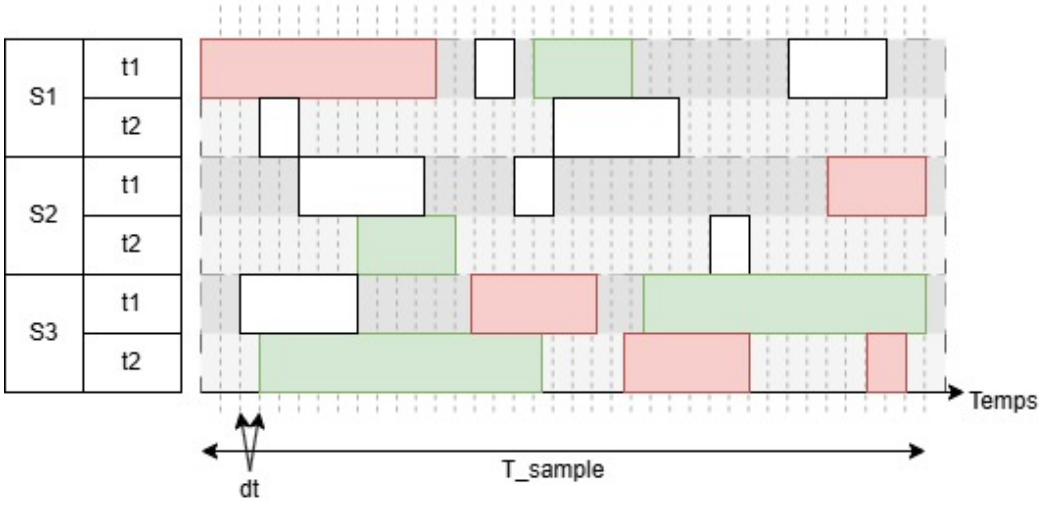


Figure 3: Diagramme temporel de démonstration de la méthode de calcul de EnergAt, sur une machine à 3 sockets matériel où 2 threads tournent en parallèle sur chaque socket

Chaque rectangle, blanc ou coloré représente l'exécution d'une action a par un thread t_j - on simplifiera le discours à thread, mais cela comprend aussi les processus - sur un socket s_i. Dans le modèle de EnergAt, il est supposé que les actions sont toutes, au final, exécutées sur un CPU. Dans le modèle de EnergAt, le temps est **discrétisé** par les prises de décision de l'ordonnanceur du kernel. Ce temps d'échantillonnage est représenté par **dt**. T_{sample} représente quant à lui la période de prise de mesure : par défaut elle est de **10ms** (valeur évaluée empiriquement comme suffisante pour les résultats attendus par le papier).

Supposons que les rectangles verts représentent l'exécution de a_0 et les rouges celle de a_1. L'application étudiée sera $A = \{a_0, a_1\}$. À toutes les décisions de l'ordonnanceur, EnergAt incrémente des compteurs internes qui lui permettent ensuite de calculer facilement les proportions des temps d'exécutions d'actions $a_k, k \in \{0, 1\}$ de la vie d'un thread t_j sur un socket s_i, pour chaque socket - ce qui revient à calculer l'aire des rectangles rouges et verts pour chaque socket de la figure 3, si l'on considère que chaque rectangle est de hauteur 1. Une fois ces proportions calculées, il est facile de pouvoir calculer le rapport entre le temps d'utilisation de l'application A et le temps total de vie de chaque socket pour chaque socket, car T_{sample} est fixe. Le papier présente cette dernière proportion comme suit :

$$(T_A^{CPU})^s / (T_{TOTAL}^{CPU})^s$$

où $(T_A^{CPU})^s$ est le temps d'utilisation de l'application A sur le socket s, et $(T_{TOTAL}^{CPU})^s$ le temps d'utilisation totale du socket s par le CPU. De cette façon, EnergAt sépare les temps "consommés"

par l'application A entre les sockets, ce qui lui permet d'affiner son modèle en définissant une nouvelle mesure appelée "crédit énergétique" par socket:

$$(C_A^{CPU})^s = [(T_A^{CPU})^s/(T_{TOTAL}^{CPU})^s]^\gamma, \gamma \in [0,1]$$

Où γ est un facteur de mise à l'échelle déduit des caractéristiques de la machine étudiée qui représente le fait que la consommation électrique croît non-linéairement par rapport à la proportion d'utilisation des CPUs.

Une fois ce crédit calculé pour chaque socket, la consommation énergétique de l'application est déduite à l'aide de l'outil RAPL d'Intel qui est utilisé pour récupérer la consommation de chaque socket sur la période T_{sample}.

Le même mécanisme est utilisé pour calculer la consommation énergétique de la DRAM, mais les threads ne sont plus pris en compte car leur PG ("programm group") est égal dans les statistiques NUMA, les rendant indissociables sur ce point. Ces statistiques pour les nœuds NUMA sont tirées du package "numactl" selon le papier. Les auteurs ont utilisé la commande "numastat" avec l'argument "p" pour récupérer les informations liées à l'utilisation mémoire du processus.

Les résultats prennent aussi la forme d'une série temporelle, stockée dans une base de donnée.

Conclusion

L'outil EnergAt se concentre principalement sur la validité de l'association entre les données de consommation et l'impact des applications étudiées. Cela n'a pas été abordé, mais son modèle se base sur une analyse probabiliste du problème, et utilise donc les outils associés à ce pan des mathématiques. L'utilisation de cet outil convient cependant mieux à des serveurs, ayant une architecture pluri-socket, et exécutant de ce fait des applications "fortement" parallélisées - le code source mentionnait même un manque de tests sur une architecture avec un seul socket.

De plus, il est important de rappeler que ces données n'incluent pas la consommation des mémoires NUMA partagées et d'autres composants plus atypiques comme GPUs, TPUs ou autre. Enfin, EnergAt ne fait pas le lien entre cette consommation et l'impact carbone de l'énergie générée.

6 Modèles empiriques

Accurate and Stable Run-Time Power Modeling

Description

Walker et al., "Accurate and Stable Run-Time Power Modeling for Mobile and Embedded CPUs" proposent ici une méthode de création de régression permettant, sous certaines conditions, de créer des régressions linéaires d'estimation de la consommation de cartes qui ne dépendent que de certains PMCs et avec un de bons résultats de corrélation. Ce papier fournit un exemple de l'application de leur méthode pour créer une abstraction statistique des CPU A7 et A15, en mettant l'accent sur la méthode de sélection des *Performance Monitoring Counters* (PMC) et le raffinement des données finales. L'étude ne couvre pas la partie logicielle, mais utilise une charge de tests substantielle composée de 60 *workloads* différentes.

Méthode

Avant d'aborder leur objet d'étude, ils évaluent l'impact de l'instrumentation de leurs programmes de test en examinant l'empreinte moyenne en puissance instantanée de la carte entre une situation où le programme est exécuté en même temps que le programme évalué et une situation "à vide". Dans le premier cas, les auteurs font varier la fréquence d'échantillonnage de leur programme sonde.

L'effet de mesure étant quantifié, les auteurs commencent par le constat que le nombre maximal de PMC pouvant être sélectionnés en même temps pour l'étude de la consommation d'un logiciel sur plateforme ARM est de six. Cette limite est due à une contrainte physique (voir section 9). De ce

constat ils développent un algorithme rigoureux de sélection des événements à analyser pour le sujet de la consommation des programmes, de telle sorte à maximiser l'impact des PMCs considérés. Cette sélection d'événements se fait en deux étapes :

1. **Regroupement et corrélation** : Les événements sont regroupés en *clusters* en utilisant la méthode de la *Hierarchical Cluster Analysis* (HCA), qui regroupe les événements en fonction de leur inter-corrélation. Ensuite, le *Moment Produit de Pearson* est utilisé pour corréler ces événements à la puissance instantanée globale. Les meilleurs événements, ceux qui influencent le plus le coefficient de corrélation entre le modèle de régression final et la consommation, sont sélectionnés un à un en suivant l'algorithme suivant :

Algorithm 1 Algorithme de sélection des meilleurs PMCs

```
1: procedure SELECT_EVENTS(allEvents, no.Events)
2:     selectedList ← cycleCountEvent
3:     while length(selectedlist) ¡ no.Events do
4:         for all pmcEvent in allEvents do
5:             build_model(selectedList + pmcEvent)
6:             if newR^2 > bestR^2 then
7:                 bestEvent ← pmcEvent
8:                 bestR^2 ← newR^2
9:         append bestEvent to selectedList
10:    return to_ret
```

la fonction "build_model" ici présente prends un ensemble de d'événements et construit une régression linéaire à partir de leurs valeurs observées pendant l'entraînement.

2. **Réduction de la multi colinéarité** : La multi colinéarité est réduite en utilisant le *Variance Inflation Factor* (VIF) et en appliquant des transformations entre les événements. Par exemple, dans le cas du Cortex-A15, l'événement PMC 0x1B compte toutes les instructions exécutées de manière spéculative, tandis que l'événement PMC 0x73 compte les instructions entières exécutées de manière spéculative. Comme l'événement 0x73 est inclus dans l'événement 0x1B, les deux événements sont nécessaires mais entraînent une multi colinéarité. Pour résoudre ce problème, les auteurs transforment l'événement 0x1B en 0x1B-0x73.

Une fois la présélection réalisée, les auteurs entraînent une régression linéaire multiple en utilisant un estimateur des *Ordinary Least Squares* (OLS). Cette régression se décrit comme suit :

$$P_{cluster} = \left(\sum_{n=0}^{N-1} \beta_n E_n V_{DD}^2 f_{clk} \right) + f(V_{DD}, f_{clk})$$

Les coefficients β_n sont estimés par OLS, E_n est la fréquence des événements (en événements/secondes), f_{clk} est l'horloge de l'ensemble des cœurs, V_{DD} est la tension d'alimentation du processeur. La partie $f(V_{DD}, f_{clk})$ représente la dépense statique liée aux phénomènes physiques en jeu dans la consommation du circuit électrique.

Cette modélisation est intéressante car elle reprend 2 principes tirés de la physique par rapport à la consommation : la tension augmente quadratiquement avec la fréquence (section 1.4[12]), la puissance totale peut-être divisée en une catégorie statique et dynamique.

Pour entraîner le modèle, les auteurs utilisent un ensemble de *benchmarks* (environ 60 différents) avec une méthode de validation croisée *k-fold* à 10. Ils valident les hypothèses de l'utilisation d'un modèle de régression linéaire en vérifiant l'*homoscédasticité* du modèle (le fait que la variance soit constante pour tous les paramètres). Cette hypothèse ne se vérifie pas dans leur cas (ni dans le cas général), mais leur modèle OLS propose de bons résultats. L'estimation de son erreur prend en compte cette propriété en utilisant le *Heteroskedasticity-Consistent Standard Errors* (HCSE).

[12]Keating, *Low Power Methodology Manual*.

Conclusion

Le tableau 1 suivant reprend les résultats affichés par le papier :

Parameter	A7 Value	A15 Value
R^2	0.993	0.997
Adjusted R^2	0.993	0.997
No. Observations	1680	2160
Std Err. of Regression (SER) [W]	0.0133	0.0517
Avg. VIF (PMC events only)	2.13	2.25
Avg. VIF (V and f inc.)	4.94	3.04

Table 1: Résultats du modèle de Walker et al., "Accurate and Stable Run-Time Power Modeling for Mobile and Embedded CPUs"

Ainsi que le résultat global, table 2, sur le jeu de donnée de validation issu de la méthode k-fold :

Parameter	A7 Value	A15 Value
No. Folds	10	10
Fold Group Size	168	216
Avg. Err. (MAPE) [%]	3.79	2.81
Mean Sq. err. (MSE) [W^2]	$1.86 * 10^{-4}$	$2.76 * 10^{-3}$
Root Mean Sq. err. (MSE)	9.75 mW	61.3 mW

Table 2: Résultats sur les données de validation de Walker et al., "Accurate and Stable Run-Time Power Modeling for Mobile and Embedded CPUs"

La méthode que proposent les auteurs de cet article est une des meilleures que nous ayons trouvé jusqu'à présent du point de vue des résultats. Pour des méthodes de cet acabit, la norme se trouve autour d'une MAPE de 10%, ici les auteurs en proposent une en-dessous des 4% ce qui est très prometteur. Cette méthode nous a servi d'inspiration pour la création de la solution de ce stage.

Estimating Applications Performance and Energy Consumption Through Static Analysis

Description

Marantos et al., "A Flexible Tool for Estimating Applications Performance and Energy Consumption Through Static Analysis" présentent une méthode pour créer un modèle statistique paramétrable permettant d'estimer la consommation en temps et en énergie d'un programme. La méthode utilise l'outil `llvm-mca` et la représentation en arbre syntaxique abstrait (AST) des codes compilés. A la différence de 6, les auteurs ne se penchent pas sur le côté physique ni sur la carte à analyser, mais plutôt les principales caractéristiques des programmes analysés.

Méthodologie

La méthode d'estimation de ce papier, représentée par la figure 4 se concentre uniquement sur les *Basic Blocks*, suite d'instructions contigües sans branchement, et non sur la modélisation de branchements ou de boucles. Une fois ces basic blocks étudiés, les auteurs calculent la consommation totale du programme en combinant ces consommations avec le nombre de tour de boucle et la destination des branchements (dernière étape avant l'estimation finale dans la figure 4).

Une autre caractéristique intéressante de cette méthode est qu'elle analyse aussi bien le code source que le code compilé. En faisant cela, les auteurs sont capables de récupérer des informations seulement disponibles par l'une des 2 versions, comme le débit d'instructions estimé pour le code compilé, ou l'identification des basics blocks via le code source. De cette façon les auteurs ont dressé la liste de paramètres suivant pour leur modèle :

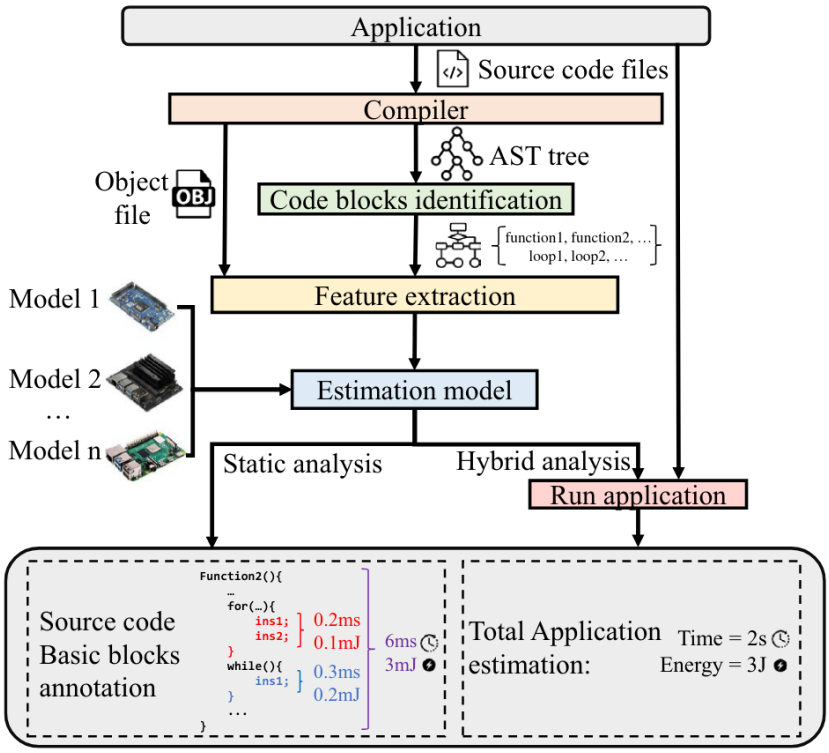


Fig. 1 Overview of the proposed methodology

Figure 4: Méthode d'estimation de Marantos et al., "A Flexible Tool for Estimating Applications Performance and Energy Consumption Through Static Analysis"

- Nombre d'instructions (INS)
- Débit estimé (par LLVM-mca)
- Nombre d'instructions LOAD
- Nombre d'instructions STORE
- Nombre d'instructions OP (opérations)
- Nombre d'instructions dans la classe INS 1 (add, sub, shift, et mul)
- Nombre d'instructions dans la classe INS 2 (conv, arrays, div)
- Ordre des instructions OP, LOAD, et STORE

On voit ici que l'analyse statique du programme amène à catégoriser les instructions en différents groupes. Ces groupes ont été créés pour la plateforme de test en particulier, ils sont dépendants de leur temps d'exécution.

Ces paramètres sont extraits depuis dans la partie "Feature extraction". Ils sont ensuite envoyés au modèle de régression, préalablement entraîné. Cette partie "Estimation Model" de la figure 4 est détaillé dans le graphique 5 suivant :

De la même façon que Walker et al., "Accurate and Stable Run-Time Power Modeling for Mobile and Embedded CPUs", l'estimation est entraînée sur un ensemble d'exécutions diverses. Les données de consommation et de puissance instantanée sont récupérées via des compteurs physiques sur les cartes, et non par simulation comme Bazzaz, Salehi, and Ejlali, "An Accurate Instruction-Level Energy Estimation Model and Tool for Embedded Systems" ou Li et al., "McPAT". Les exécutions en question sont des exécutions de boucles aléatoires contenant des opérations sur matrices, vecteurs et scalaires

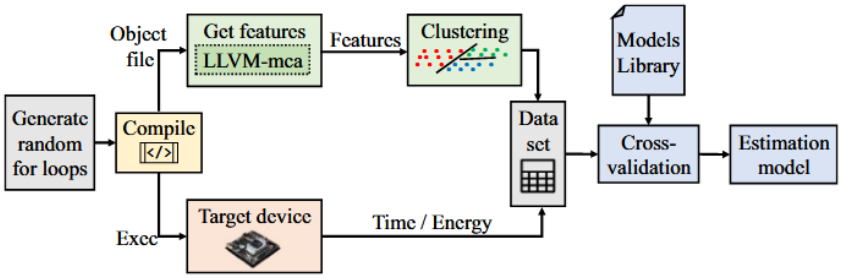


Figure 5: Fonctionnement de la création du modèle de régression

dont le nombre de matrices, type de données et opérations sont aléatoires. Les données sont ensuite divisées par le nombre de boucle pour avoir une estimation des basic blocs interne des boucles. Ces benchmarks permettent d'entraîner des modèles de régression choisis parmi un ensemble utilisable avec la bibliothèque d'analyse statistique choisie. Les auteurs retiendront à la fin le modèle correspond le mieux au cas d'usage étudié : l'énergie ou le temps.

Conclusion

Une des conclusions de ce papier est qu'un modèle universel ne convient pas à l'estimation du temps d'exécution optimale et de la consommation énergétique optimale : les auteurs suggèrent donc d'utiliser le modèle le plus efficace pour le cas concerné par l'utilisateur.

Pour comparer avec les modèles proposés par Walker et al., "Accurate and Stable Run-Time Power Modeling for Mobile and Embedded CPUs", les auteurs proposent les graphiques suivants 6.

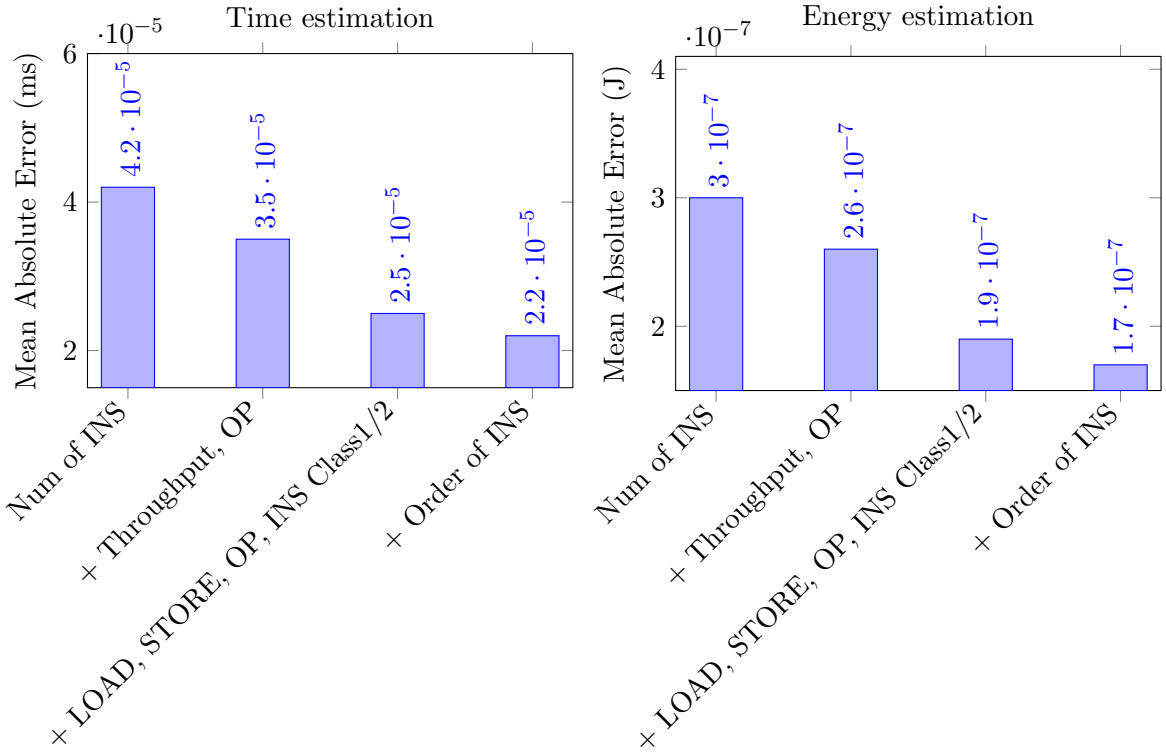


Figure 6: MAPE en fonction des paramètres inclus dans le modèle

De la même façon que dans le papier de Walker, les auteurs montrent ici une diminution de l'erreur moyenne plus de paramètres sont ajoutés dans le modèle. Les auteurs ne vont pas jusqu'à montrer le VIF des modèles comme les précédents auteurs, mais cela montre tout de même une démarche similaire : partir de peu de paramètres et en ajouter au fur et à mesure sans perdre en fiabilité.

Au final, ce papier apporte une vision complémentaire à la méthode précédemment analysée. Là où celle de Walker se concentrait sur l'abstraction de la puce, celle-ci est tourné vers le logiciel. Les benchmarks d'entraînement sont contrôlés mais les auteurs portent moins d'attention à la définition du modèle en lui-même : ils prennent celui qui marche le mieux dans un ensemble déjà existant.

Cette méthode nous a inspiré dans le sens où elle se concentre sur des aspects du programme, ce que nous asseyons de reprendre en comptant les différents types d'instructions.

Part V
Outils de mesure de la consommation

Cette partie présente une liste non exhaustive des outils qui ont été retenus pour la production de résultats pendant ce stage. Nous rentrons plus en détails dans le fonctionnement de 4 d'entre eux : RAPL, BCU, Perf et eBPF. Les deux premiers sont des outils nous permettant de récupérer la consommation énergétique des cartes étudiées et les deux suivants nous permettent de rentrer en détails dans le fonctionnement des systèmes étudiés pour expliquer la provenance de ces consommations.

7 RAPL

Intel propose depuis son architecture SandyBridge un outil de mesure de la consommation nommé RAPL (Running Average Power Limit). Nous n'avons pas trouvé de source fiable quant à la nature exacte de ce support physique, mais nous supposons qu'il embarque à minima une mesure de courant et de tension doté d'un algorithme d'approximation de la consommation.

RAPL est un outil qui permet de contrôler et analyser la consommation de différentes parties de la puce en fonction des versions de la microarchitecture sur laquelle il est implanté. Ces localités sont la RAM, les Coeurs CPU, le package, et une dernière appelée "non-coeur"(GPU principalement). RAPL permet aussi d'avoir la main sur la température de la puce, en lui imposant certaines règles. La documentation développeur (volume 3b, chapitre 16, page 55) indique que le temps d'échantillonnage pour récupérer les informations de consommation détenues dans les MSRs (Model Specific Registers) de la puce est **d'une milliseconde environ**. Le fait que cette période en soit pas précise, et puisse fluctuer rend l'analyse de consommation à "grain fin" complexe. Hähnel et al., "Measuring Energy Consumption for Short Code Paths Using RAPL" étudie ce problème de façon détaillé en montrant une méthode permettant de réduire l'erreur de prise de mesure. Cependant, cette méthode ne s'étend pas à des architectures où des programmes tournent sur plusieurs cœurs.

L'avantage principal de cet outil est qu'il est présent sur la plupart des architectures Intel récentes, permettant ainsi de les comparer entre elles sans avoir à les instrumenter. Ainsi, bon nombre d'outils d'analyse de consommation et de mesure d'impact carbone logiciel se base cette technologie[13][14][15].

Le défaut majeur de cet outil est son côté propriétaire. Toujours d'après la documentation du processeur, il implante un algorithme de prévision de consommation, et non une mesure directe : ce qui rend opaque la méthode de prise de mesure. David et al., "RAPL" décrit le modèle que RAPL utilise pour sa localité DRAM. Il montre comment, en définissant différents états de RAM, correspondant à différents niveaux de débit mémoire et de puissance utilisé, RAPL fait pour arbitrer lequel prendre. Cependant cela ne concerne que la localité DRAM.

Fiabilité Certains auteurs se sont penchés sur la fiabilité des résultats de RAPL. Desrochers, Paradis, and Weaver, "A Validation of DRAM RAPL Power Measurements" montre que sur le serveur Haswell-EP avec 80 Gb de DDR4, RAPL peut avoir une erreur de mesure de la RAM allant jusqu'à 74% lorsque les valeurs mesurées étaient faibles. Khan et al., "RAPL in Action" montre au contraire

[13] *Dépôt de codecarbon.*
[14] *Dépôt de Carbontracker.*
[15] *Dépôt de Scaphandre.*

une très bonne corrélation (coefficient de corrélation de 0,99) de l'outil sur des zones différentes en comparant la zone "package", censée représenter la consommation totale de la puce avec la consommation mesurée à la prise. Les auteurs expliquent cette différence par l'amélioration de la technologie en fonction du temps et mais montrent également, une incertitude plus grande sur la zone DRAM de RAPL. Dans le cas de la x7ti, le compteur RAPL concernant la RAM n'est pas disponible. En effet, depuis l'architecture RaptorLake, Intel a abandonné cette fonctionnalité. Ce changement reflète bien le souci de dépendance à l'outil : une méthode, aussi bonne soit-elle, ne saurait perdurer dans le temps en ne se basant seulement que sur un seul outil de mesure.

Récupération de l'information Un autre avantage de RAPL est qu'il existe depuis longtemps et bénéficie d'une multitude d'outils plus ou moins facile à prendre en main pour l'utiliser en tant développeur. En voici une liste :

- MSRs (Model Specific Registers) : C'est la façon le plus bas niveau de récupérer et piloter RAPL. Décrits dans la documentation développeur d'Intel, volume 3[16] , ces compteurs agissent directement sur le matériel et permettent de récupérer les informations à la source.
- Powercap : Depuis la version 3.13 du noyau[17], ce système de fichiers est accessible sur Linux, facilitant la récupération de l'information à la simple lecture d'un fichier dans une arborescence.
- Perf_events : perf_events est une interface système qui permet la lecture d'un grand nombre de compteurs de performances (PMCs, performance monitoring counters) tels que le nombre de cycles, le nombre de cache hit/miss, etc. Cette interface propose la lecture de MSR, et donc ceux de RAPL, ainsi que la lecture de l'interface Powercap du système.
- BPFe (Berckelet Packet Filter extended) : est une fonctionnalité du kernel, qui a le droit a sa propre partie juste après. Il permet d'aller lire les registres spécifiques à RAPL directement depuis le noyau.
- Utiliser un outil plus haut niveau : Nombre d'outils utilisent RAPL comme nous l'avons évoqué. Certains d'entre eux proposent une interface unifiée avec d'autres types de compteurs de telle sorte qu'il est très facile de les analyser ensemble. Weaver et al., "Measuring Energy and Power with PAPI" en est un répandu complet.

Raffin and Trystram, *Dissecting the Software-Based Measurement of CPU Energy Consumption* aborde la façon la plus optimale en terme d'empreinte sur la mesure pour la lecture des compteurs de RAPL. La conclusion des auteurs est que perf_events semble être la meilleure option. La figure 7 suivante, tirée du même papier montre un résumé des méthodes comparées et de leurs avantages/inconvénients en fonction de plusieurs critères. Celui qui nous intéresse principalement pour notre cas d'étude est "technical difficulty" qui représente - selon ses auteurs - la difficulté d'utilisation de l'outil.

Nous avons finalement choisi d'utiliser PAPI, un outil plus complet qui permet l'intégration d'autres mesures plus facilement. PAPI nous propose d'utiliser l'interface powercap, qui dans le tableau ci-dessus 7 est dans les plus faciles d'utilisation. Cependant, le choix de l'utilisationd de PAPI est advenu avant la découverte du papier, ce qui pourrait remettre en cause son issue.

8 BCU

BCU (Board Control Utilities) est un programme développé par NXP ayant pour but de contrôler et renvoyer des informations de la majorité de leurs cartes de série i.MX. L'i.MX93evk, carte que nous utilisons dans le cadre de ce stage, est instrumentée de capteurs de diverse sortes.

Elle embarque notamment des capteurs PAC1934[18], des multimètres qui permettent la mesure de l'énergie et de la puissance sur un ensemble large de composants séparément.

[16] *Documentation développeur d'Intel.*
[17] *Site de noireaudes page sur l'analyse de consommation.*
[18] **bibid**.

mechanism	technical difficulty	required knowledge	safeguards	privileges	resiliency
MSR	medium	CPU knowledge	none	SYS_RAWIO cap. + msr module	poor
perf-events + eBPF	high (long, complicated code)	limited	overflows unlikely, many other possible mistakes	PERFMON and BPF capabilities	manual tweaks necessary for adaptation
perf-events	low	limited	good, overflows unlikely	PERFMON capability	good
powercap	low	limited	beware of overflows	read access to one dir	good, very flexible

Table II
QUALITATIVE COMPARISON OF THE MEASUREMENT MECHANISMS

Figure 7: Comparatif des différentes méthode d'utilisation de RAPL tirées de Raffin and Trystram, *Dissecting the Software-Based Measurement of CPU Energy Consumption*

PAC1934 Ces capteurs possèdent plusieurs modes et méthodes de calcul du courant (pour coller à différentes échelles de valeurs) dans lesquels nous ne rentrerons pas en détail ici. Il est cependant important de noter les choses suivantes concernant ces capteurs :

- La fréquence d'échantillonnage peut-être définie, mais est par défaut de **1024 samples/secondes** (équivalent à celle de RAPL donc)
- Le mode d'acquisition est "bipolaire" par défaut. Dans ce mode, le capteur peut interpréter des valeurs négatives de tension mathématiquement. La raison de ce choix pour notre cas est que l'incertitude détaillée dans la documentation du PAC1934 relative à ce mode pour les valeurs de puissance que nous observons est environ dix fois plus petite que dans le mode inverse, "unipolaire".
- Les valeurs enregistrées dans la carte sont des valeurs moyennées sur 8 échantillons (comportement par défaut). Il est possible de passer outre ce comportement, mais cela induirait une incertitude supplémentaire, décrite page 22 de la documentation.

Ces capteurs sont un des avantages majeurs de cette carte : il nous est facilement possible d'isoler les postes de consommation de la carte en ne sélectionnant qu'un sous-ensemble des zones observées par ces composants, nous permettant donc de supprimer de l'aléa et de cibler la mesure. Par exemple, dans nos mesures pour la construction de notre modèle, nous avons seulement considéré la ligne "vdd_soc" de la présente figure, car elle ne représente que la consommation des 2 coeurs A55 de la carte ainsi que leurs caches les plus proches (L1 et L2). En faisant cela, nous écartons la consommation de la RAM, ce qui nous évite de considérer les accès qui sont causés par la politique de gestion de cache (et de mémoire plus globalement) que nous ne maîtrisons pas dans ce cas d'étude. Ainsi, même si nos programmes accèdent à de la mémoire de amnière imprévu, cela n'apparaît pas dans la consommation finale.

Récupération de l'information Ces zones sont exprimées sous la forme de voies dans le logiciel BCU, et traduisent directement ce que les capteurs reçoivent comme informations. Il est d'ailleurs possible (ce que nous n'avons pas exploré) de modifier les paramètres des capteurs à travers cette interface. La figure 8 suivante montre un aperçu de retour direct du lancement de l'outil avec la

sous commande "monitor". Ces données sont directement disponibles en version csv, avec une option d'export au lancement du logiciel. C'est ce que nous utilisons pour récupérer les données de consommation et pouvoir les traiter après coup.

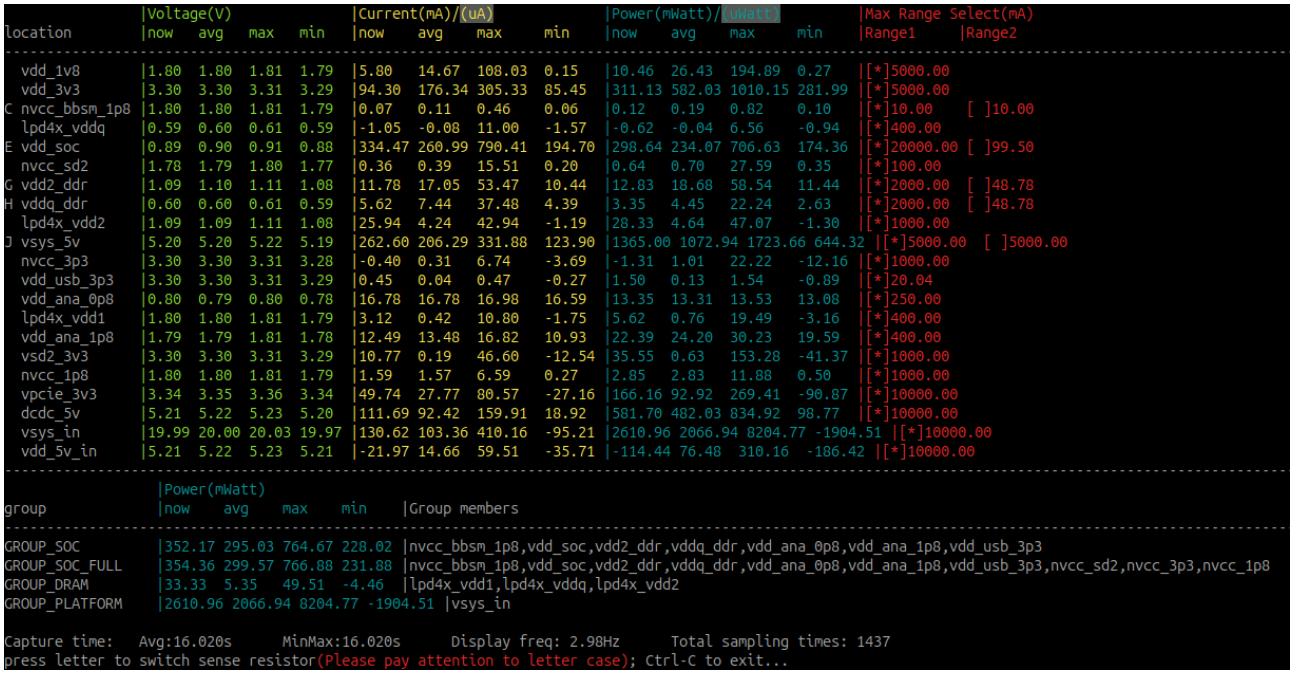


Figure 8: Capture d'écran du prorgamme BCU en fonctionnement sur l'i.mx93

9 PMCs et état du système

La requête d'informations sur le système s'est avérée nécessaire à la résolution de notre problème. En effet un ensemble de techniques, partie 6, sont directement liées à l'étude de compteurs systèmes pour caractériser un profil de consommation.

Dans cette optique, l'usage d'outils comme perf ou eBPF et Intel Pin, dont nous dressons un rapide portrait ici, s'est avéré essentiel. Avant cela, il est important de revenir sur le fonctionnement d'un composant principal dans l'analyse des systèmes : la PMU.

PMU et PMCs

Les PMU (Performance Monitoring Unit) sont des unités matérielles dédiées à la collecte de données sur les événements micro architecturaux et architecturaux du processeur. Elles permettent de mesurer des métriques comme le nombre de cycles d'horloge, d'instructions exécutées, d'accès mémoire, de prédictions de branchement, et d'autres événements spécifiques à la microarchitecture. Elles fonctionnent via un ensemble de registres de comptage et de contrôle. Les registres de comptage (PMCs) accumulent le nombre d'occurrences d'événements spécifiques, tandis que les registres de contrôle permettent d'activer, désactiver, et configurer les compteurs.

En général, les PMUs des architectures ARMv8.1 et x86_64 présentent des limites significatives. ARMv8.1, malgré ses 31 compteurs programmables ce voit réduire le nombre de compteurs concurrents maximum à 6 sur le A55. Pour x86_64, les PMUs sont généralement limitées à 8 compteurs et souffrent de variabilité entre les implémentations d'Intel et AMD, rendant le code moins portable.

Enfin, il existe aussi des limites sur les types des compteurs, de telle façon que même pour un nombre inférieur à la limite maximale de compteur, la PMU pourrait ne pas les supporter. Par exemple, pour les P-core de la x7ti, seulement 3 compteurs de nom "MEM_*_RETIRED_*" peuvent être instrumentés en même temps. Les informations quant aux limites d'utilisation de ces unités sont

disponibles dans la documentation développeur d'Intel pour x86_64, *Répertoire de PMCs d'Intel*, et la note sur les PMU pour ARMv8.1 *Notice d'utilisation de la PMU d'ARMv8.1*.

L'instrumentation et l'utilisation de ces compteurs ont déjà été implantées par des outils répandus comme perf ou eBPF que nous introduisons juste après, il ne nous a donc pas été nécessaire de réimplanter ces fonctionnements. Cependant, par soucis d'homogénéité et de contrôle sur l'erreur pour une application finale, il pourrait être envisagé de reprendre ces lectures et de les intégrer à notre solution.

Logiciels

Perf : perf, introduit en 2009 avec le noyau Linux 2.6.31, est un outil de profilage et d'analyse des performances. Il permet de mesurer divers événements liés à la performance, tels que les cycles CPU, les instructions exécutées et les défauts de cache.

perf peut s'interfacer avec la Performance Monitoring Unit (PMU) des processeurs (si implantée) via l'infrastructure perf_events du noyau Linux. Il configure les événements de performance à surveiller dans la PMU puis collecte et interprète les données qu'elle retourne. Nous utilisons la commande perf stat pour récupérer les données du système. L'option -I permet de spécifier un intervalle de temps pour l'affichage des statistiques. Par exemple, -I 1000 affichera les statistiques toutes les 1000 millisecondes. Cela nous permet d'analyser les données des programmes exécutées en fonction du temps.

eBPF : eBPF a été introduit dans le noyau Linux 3.18 en 2014. eBPF a connu de nombreuses améliorations et est devenu une technologie importante pour la surveillance, la sécurité et la mise en réseau dans le noyau Linux. La version 4.16 du noyau Linux a marqué une étape majeure pour eBPF avec l'introduction de la compilation juste-à-temps (JIT) et d'autres améliorations significatives. Une illustration de son fonctionnement, réalisée par Brendan Gregg sur son site *Page eBPF du site de Brendan Gregg*, est affichée ci-dessous 9.

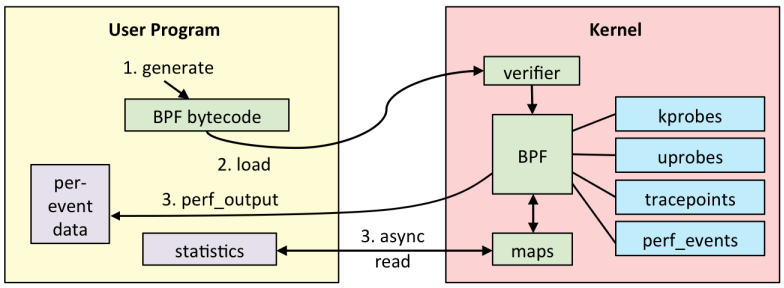


Figure 9: Fonctionnement global de eBPF

L'utilisateur peut créer des programmes BPF, de la façon dont il le souhaite, qui seront ensuite transmis au kernel pour être exécuté. De cette façon, un utilisateur peut récupérer des informations depuis le runtime du kernel, sans avoir à compiler de module spécifique. Plusieurs outils instrumentent la technologie BPF : les deux plus connus sont BCC et Bpftrace. Le premier est un compilateur qui permet de traduire un programme C en bytecode BPF. Le fait qu'il utilise directement l'API de la libbpf du kernel lui permet de proposer un contrôle complet sur la technologie. Bpftrace est un outil proposant un langage particulier, plus expressif, pour créer des programmes BPF. À la façon d'un shell, il permet de lancer ses programmes en une commande sans difficulté supplémentaire quant à la compilation. Cependant, ce gain en expressivité apporte son lot de limitations, qui peuvent être un frein à la récupération de certains états du système.

La communauté[19] conseille de commencer par l'usage de bfptrace, ou d'outils déjà existant construits sur bcc pour parvenir à ses besoins plutôt que d'essayer d'utiliser directement BCC. C'est d'ailleurs une des remarques que souligne Raffin and Trystram, *Dissecting the Software-Based Measurement of CPU Energy Consumption* : bcc est complexe à utiliser. C'est ce que nous avons donc fait.

[19] *Page de tutoriel de bcc.*

eBPF nous a été utile pendant nos phases d'expérimentation. Il nous a notamment permis de mettre en valeur un décalage entre la mesure de consommation et le lancement du programme sur i.mx93 ainsi que de montrer les outils de mesures peuvent avoir un effet non négligeables sur la consommation.

Intel PIN Intel PIN est un framework d'instrumentation binaire dynamique développé pour les architectures IA-32 et x86-64 par Intel. Initialement créé avant 2007, la version la plus récente est la version 3.31 publiée en juin 2024, celle que nous utilisons. Ce framework permet l'insertion de code d'instrumentation dans des programmes binaires compilés sans nécessiter de recompilation du code source. Il offre une API qui permet une instrumentation à différents niveaux de granularité, notamment au niveau de l'image binaire, des routines, des traces d'exécution et des instructions individuelles. PIN gère automatiquement la sauvegarde et la restauration des registres, assurant ainsi l'intégrité de l'exécution du programme instrumenté.

Parmi les fonctionnalités principales de PIN, on trouve l'instrumentation prédiquée, qui permet une exécution conditionnelle de code injecté, ainsi que l'interception des appels système via des fonctions de rappel. Ces caractéristiques nous ont permis de créer un Pintool, plugin spécifique à l'utilisation de PIN et compilé avec makefile donné par le projet.

Dans notre cas, nous avons utilisé la sonde permettant d'introduire des compteurs après chaque instruction. Les résultats sur ces mesures et leurs corrélation avec la consommation sont présentés en partie 17.

Part VI
Travail réalisé

Avant d'aborder les choix que nous avons fait ainsi que les directions prises pour arriver à la solution présentée dans cette section, il est important de noter que nous réduisons le problème à l'étude seule des cœurs de calcul, des CPUs. Notre but est de pouvoir estimer la consommation du CPU seule, et non d'autres parties comme la mémoire, pour simplifier le problème.

10 Méthode générale

La méthode que nous utilisons finalement lors de ce stage s'inspire de la catégorie de méthodes présentée dans la partie 6. Dans cette section nous passons en revue les raisons de ce choix ainsi qu'une description précisée de la cible de ce stage et de notre méthode générale pour y parvenir.

Choix

Nous avons fait le choix d'utiliser une méthode de régression pour plusieurs raisons :

Difficulté d'utilisation des méthodes analytiques Nous avons abordé le problème lors de la description de McPAT : ces outils sont difficiles d'utilisation. En effet, pour le faire fonctionner, il faut renseigner un fichier contenant plusieurs dizaines de lignes de paramètres dont les valeurs nous sont parfois obscures. C'est d'ailleurs une caractéristique inhérente au domaine d'étude dans lequel nous évoluons avec ce genre d'outils : la conception de puces. Les architectures open-sources implantées dans de vraies projets sont rares et encore à l'état de découverte dans les projets industriels. De ce fait, il n'est pas possible d'atténuer le problème en s'orientant vers ces alternatives plus libres, car cela ne profiterait pas à Thales. D'autant plus que le contexte du stage fait que le modèle doit pouvoir estimer sur 2 cartes d'architectures différentes : donc 2 fois plus de travail.

Précision McPAT était précis à 10% en moyenne sur le Niagara, sortie en 2005. Nous nous proposons de créer un modèle qui puisse à la fois tourner sur x86_64 MeteorLake (Intel Core 9 Ultra 185H) sortie en décembre 2023 et ARM A55 annoncé en 2017. Les différences micro-architecturales

sont telles que nous doutons raisonnablement de l'efficacité d'un tel outil sur des architectures récentes comme celles que nous visons. Butko et al., "Full-System Simulation of Big.LITTLE Multicore Architecture for Performance and Energy Exploration" étudie l'efficacité de McPat sur des architectures ARM plus récentes que le Niagara - le A7 et A15 respectivement sortis en 2011 et 2010. L'erreur moyenne de mesure de l'énergie oscille entre 22% et 28% en fonction des cœurs considérés dans cette microarchitecture hétérogène. Les meilleures méthodes de la seconde catégorie étudiées sur les mêmes cœurs arrivent à des approximations moyenne de 4%.

Implantation

Considérant les choix décrits, nous nous sommes orienté vers des modèles de régression pour représenter la consommation. L'architecture que nous avons adopté est classique, nous la décrivons dans la figure 10 suivante :

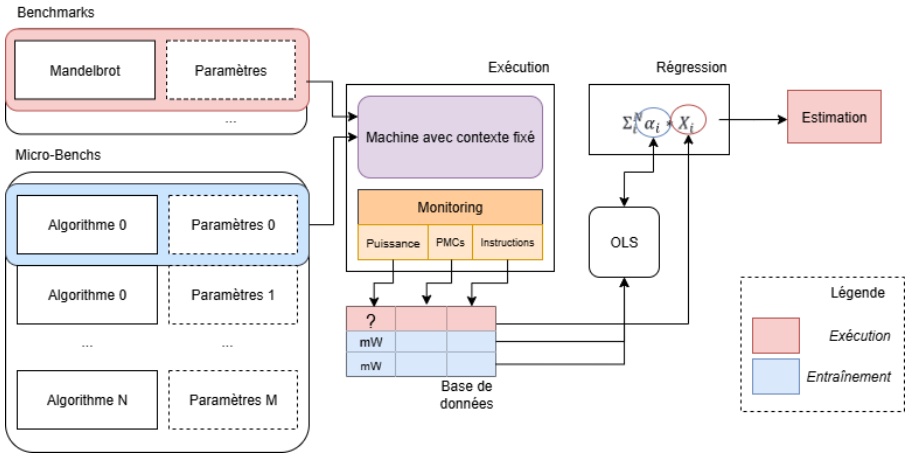


Figure 10: Méthode d'estimation finale

Le principe est le suivant. Lors de l'entraînement (éléments en bleus dans le graphique), des micros-benchmarks sont exécutés pour récupérer un jeu de données couplant paramètres et énergie dépensée par exécution. Ce jeu de données servira d'entraînement à notre modèle de régression, que nous avons choisis être un modèle OLS (Ordinary Least Squares - Méthode des Moindres Carrés). Dans une seconde phase (éléments en rouge), le modèle de régression linéaire ainsi créé pourra être utilisé sur les exécutions et l'algorithme cible. Un des gros défauts de notre méthode actuelle est qu'elle oblige l'exécution de l'application cible pour pouvoir anticiper sa consommation. Ce problème pourra, à l'avenir, être évité en imaginant un modèle prédictif de PMCs. Le choix de l'OLS comme modèle n'est motivé par aucun argument rationnel mesuré si ce n'est qu'il est l'un des plus simple : il peut être voué à changer à l'avenir si des tests montrent qu'il manque d'efficacité. Certains auteurs[20] l'ont utilisé et ont réussi à trouver de bons résultats, d'autres[21] préfèrent des méthodes plus automatique dans la sélection du modèle. Enfin, certains[22] prennent des modèles basés sur des arguments physiques - les PMCs ne peuvent pas faire varier la consommation da façon négative par exemple. Le choix du modèle précis à utiliser reste donc un vecteur d'amélioration potentiel à explorer.

11 Cible et Micro-benchmarks

Cible du stage Comme nous l'avons mentionné en début de section, le problème est réduit à l'étude des CPUs. Nous avons concrétisé cet objectif par l'étude d'un algorithme cible : la fractale

[20] Walker et al., "Accurate and Stable Run-Time Power Modeling for Mobile and Embedded CPUs".
[21] Marantos et al., "A Flexible Tool for Estimating Applications Performance and Energy Consumption Through Static Analysis".
[22] Mazzola et al., *Data-Driven Power Modeling and Monitoring via Hardware Performance Counters Tracking*.

de Mandelbrot[23]. Cet algorithme se résume au calcul de l'ensemble des points $c \in \mathbb{C}$ pour lesquels la suite suivante est bornée :

$$\begin{cases} z_0 = 0 \\ z_{n+1} = z_n^2 + c \end{cases}$$

Traditionnellement, cette fractale est représentée sous la forme d'une image où les composantes réelles et imaginaires de c sont les coordonnées du pixel et la couleur est déterminée par le nombre d'itérations réalisées par l'algorithme, normalisés sur une échelle allant de 0 à un nombre maximal d'itérations. Les conditions d'arrêt de calcul de la suite sont le nombre d'itérations et le fait que le module de z_i ne dépasse pas une certaine borne .

Choix des micro-benchmarks Une fois la cible définie, une question légitime se pose quant à la nature des micro-benchmarks utilisés pour l'entraînement du modèle. En effet, entraîner des benchmarks comme STREAM[24] n'aurait pas de sens car orientés mémoire : mais quels benchmarks choisir donc ?

La littérature travaillant sur la problématique ne mentionne pas de solution automatisée/approfondie quant à la sélection des benchmarks d'entraînement du modèle, en tout cas pas à notre connaissance. Nous pouvons supposer que c'est dû au fait qu'à la différence de notre cas, il n'est pas nécessaire aux auteurs de sélectionner ces micro-benchmarks car leur modèle a pour but de s'appliquer à n'importe quel programme. De ce fait, l'usage de benchmarks généralistes dans le domaine embarqué comme MiBench[25], ou spécialisé pour des architectures hétérogènes, comme Rodinia[26], est suffisant en fonction de la cible.

Choisir des programmes "similaires" au programme cible comme benchmarks d'entraînement est une hypothèse plausible. C'est celle que nous avons choisis de suivre sans plus de recherche par manque de temps et car ce problème est annexe au sujet principal du stage. Certains auteurs ont creusé cette piste sur la similarité entre programmes, notamment pour la constitution d'ensembles de benchmarks avec thématique comme Che et al., "Rodinia". Hoste and Eeckhout, "Microarchitecture-Independent Workload Characterization" propose une caractérisation des benchmarks de manière statistique et se basant sur des critères - étrangement - similaires à ceux utilisés pour l'évaluation de consommation. Williams, Waterman, and Patterson, "Roofline" propose une caractérisation par limite du programme (mémoire ou calculatoire), et peut même proposer des paliers en fonction des optimisations faites ou non sur le programme.

Pour notre problème nous avons choisis d'entraîner le modèle sur 4 algorithmes similaires à Mandelbrot, des algorithmes de fractales. En voici un récapitulatif :

L'attracteur de Henon Ensemble des points (x_i, y_i) du plan tels que

$$\begin{cases} x_0 = y_0 = 0 \\ x_{n+1} = y_n + 1 - a * x_n^2 \\ y_{n+1} = b * x_n \end{cases}$$

où $a = 1,4$, $b = 0,3$, $n + 1 < N$ et $i \in [0, N]$.

Dépendances :

- N :="Nombre maximal d'itérations"

[23] *Page de tutoriel de bcc.*
[24] McCalpin, *STREAM: Sustainable Memory Bandwidth in High Performance Computers.*
[25] Guthaus et al., "MiBench".
[26] Che et al., "Rodinia".

Triangle de Sierpinski *(Version jeu du chaos)*

Algorithm 2 Sierpinski version chaos

```
x ← 0.5, y ← 0.5
v ← [(0 0), (1 0), (0.5 1)]
to_ret ← {}
for i ← 0 à N do
    j ← random(0, 3)
    (x y) ← ((x y) + v.get(j))/2
    push (x y) in to_ret
return to_ret
```

Dépendances :

- N :="Nombre maximal d'itérations"

Fougère de Barnsley Soient les fonctions suivantes :

$$f_1 = \begin{bmatrix} 0 & 0 \\ 0 & 0.16 \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix}, \quad f_2 = \begin{bmatrix} 0.85 & 0.04 \\ -0.04 & 0.85 \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 0 \\ 1.6 \end{pmatrix},$$
$$f_3 = \begin{bmatrix} 0.2 & -0.26 \\ 0.23 & 0.22 \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 0 \\ 1.6 \end{pmatrix}, \quad f_4 = \begin{bmatrix} -0.15 & 0.28 \\ 0.26 & 0.24 \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 0 \\ 0.44 \end{pmatrix}.$$

L'algorithme de Barnsley est le suivant :

Algorithm 3 Barnsley

```
x ← 0, y ← 0
to_ret ← {}
for i ← 0 à N do
    w ← random()
    if w < 0.01 then j ← 1
    else if w < 0.86 then j ← 2
    else if w < 0.93 then j ← 3
    else j ← 4
    (x y) ← f_j((x y))
    push (x y) in to_ret
return to_ret
```

Dépendances :

- N :="Nombre maximal d'itérations"

Fractale du Burning Ship Ensemble des points c du plan complexe pour lesquelles la suite suivante est bornée :

$$\begin{cases} z_0 = 0 \\ z_{n+1} = |\Re(z_n) + i * \Im(z_n)|^2 + c \end{cases}$$

avec $n+1 < N$. Nous restreignons le plan à l'étude d'une image, $\Re(c) < width, \quad \Im(c) < height$.

Dépendances :

- N :="Nombre maximal d'itérations"
- $width$:="largeur de l'image"
- $height$:="hauteur de l'image"

12 Dimensions de la régression linéaire

Tout comme Walker et al., “Accurate and Stable Run-Time Power Modeling for Mobile and Embedded CPUs”, Mazzola et al., *Data-Driven Power Modeling and Monitoring via Hardware Performance Counters Tracking*, Noudohouenou and Jalby, “Using Static Analysis Data for Performance Modeling and Prediction” ou encore Bazzaz, Salehi, and Ejlali, “An Accurate Instruction-Level Energy Estimation Model and Tool for Embedded Systems” notre modèle se base sur des PMCs, dont le mécanisme a été décrit en partie 9. Nous listons ici ceux que nous avons choisis pour ce stage. Ces choix sont le résultat des différentes remarques et résultats des papiers jusqu'à maintenant cités sur le sujet. Dans la majorité des cas, ces travaux travaillaient sur des architectures ARM, les rendant donc caduque pour la x7ti, sous architecture x86_64. Pour pallier à ce problème, nous avons tenté de trouver des compteurs similaires à ceux choisis pour ARM basé sur leurs descriptions dans leurs pages dédiées, *Répertoire de PMCs d'Intel*. La définition de ceux d'ARM sont quant à elles disponibles dans *Notice d'utilisation de la PMU d'ARMv8.1*.

En plus de ces informations, plusieurs travaux précédemment cités proposent d'analyser le type des instructions exécutés. Marantos et al., “A Flexible Tool for Estimating Applications Performance and Energy Consumption Through Static Analysis” propose une découpe en quelques groupes d'instruction en fonction de leur temps d'exécution.

Enfin, nous revenons plus en détails sur les parties des carte choisies et mesurées par RAPL et BCU.

Métrique	Valeurs pour l'imx93	Valeurs pour la x7ti
Consommation	La consommation des **2 cœurs A55** et de leurs **cache L1 et L2**. Les **contrôleurs** liés à l'Ethernet, la mémoire, les composants analogiques et autres sont aussi pris en compte dans la zone que nous avons contrôlé (”vdd_soc”).	La consommation de **l'ensemble des cœurs**, de leurs **cache L1 et L2** (quand il y en avait). Comme expliqué dans la partie 7, la validité de l'ensemble des composants englobés par la zone RAPL (”ENERGY_UJ:ZONE0_SUBZONE0” dans powercap) que nous avons choisi d'analyser n'est pas décrit explicitement par la documentation Intel.
PMCs	INST_RETIRED	INST_RETIRED.ANY
	DP_SPEC	UOPS_EXECUTED.CORE
	BUS_ACCESS	LONGEST_LAT_CACHE.REFERENCE
	L1I_CACHE_ACCESS	MEM_LOAD_RETIRED.L1_HIT + MEM_LOAD_RETIRED.L1_MISS
	STALL_BACKEND + STALL_FRONTEND	CYCLE_ACTIVITY.STALLS
	VFP_SPEC	FP_ARITH_INST_RETIRED.SCALAR
Instructions	ADD,SUB,MUL,DIV,SHIFT,MOV,CMP,NOP	

Table 3: Ensemble des dimensions de notre régression linéaire

13 Prise de mesure

Pour créer la base de données d'entraînement du modèle, nous avons développé un ensemble de scripts (Shell et Python principalement) permettant de compiler, déployer, exécuter et traiter les données des

micro-benchmarks dont nous avons parlé dans la partie précédente. La façon dont nous avons pris les mesures jusqu'à aujourd'hui est décrite en fonction des cartes et de la multiplicité des benchmarks dans les graphiques 11, 12, 13 et 14 suivants.

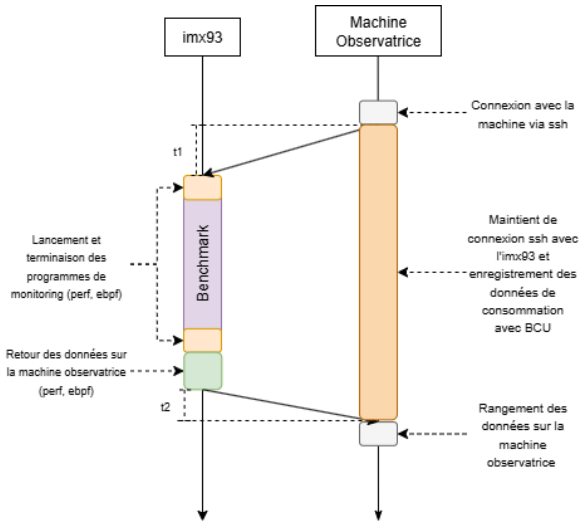


Figure 11: Diagramme temporel de l'exécution et la mesure d'un benchmarks sur imx93

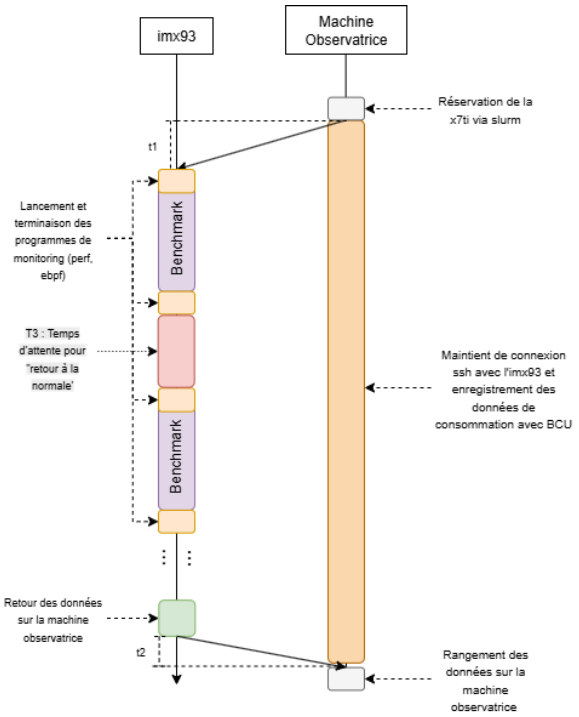


Figure 12: Diagramme temporel de l'exécution de plusieurs benchmarks sur l'imx93

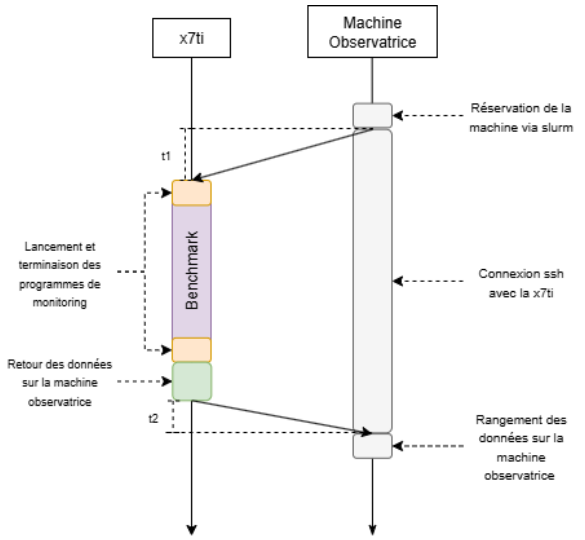


Figure 13: Diagramme temporel de l'exécution d'un benchmark sur la x7ti

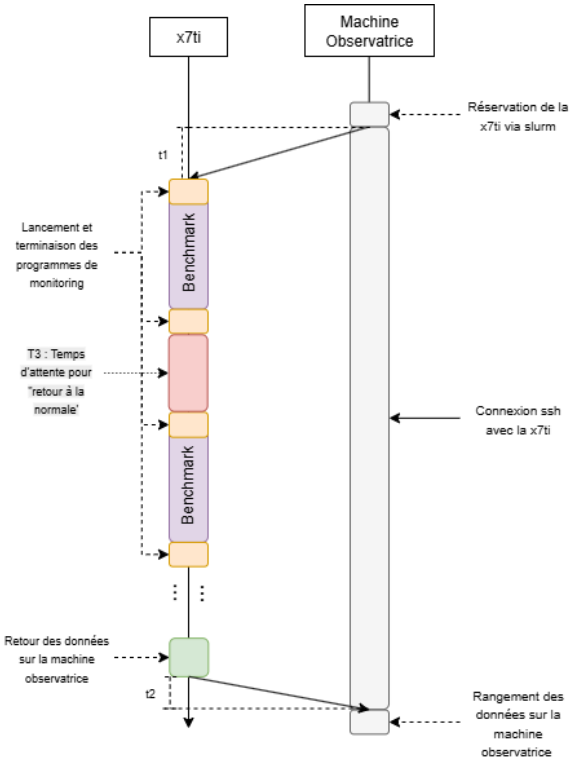


Figure 14: Diagramme temporel de l'exécution de plusieurs benchmarks sur la x7ti

La différence majeure entre la méthode sur imx93 et celle sur x7ti est que l'outil de mesure de

l'énergie pour l'une est accessible hors de la carte alors que pour l'autre non. Dans notre implémentation, les horloges entre l'i.mx93 et la machine observatrice ne sont pas synchronisées ce qui peut amener à des erreurs de mesure. Selon Raffin and Trystram, *Dissecting the Software-Based Measurement of CPU Energy Consumption* requêter RAPL via l'interface powercap prends entre 1,9 μs et 2,1 μs ce qui est négligeable dans notre cas d'étude (grain de mesure à 1ms).

De façon systématique, le système de mesure attend un temps donné (cases rouges) entre l'exécution de 2 benchmarks. La raison à cela est que nous avons observé des effets de bords liés au chargement de programmes eBPF et de monitoring de façon générale qui ne s'estompaient pas à la fin de l'expérience. Ce temps d'attente permet au système de retrouver une consommation égale à celle avant expérience.

Enfin, notre système maintient une connexion ssh avec la machine observée, ce qui augmente la consommation de l'i.mx93 (nous n'avons pas encore pu le tester sur la x7ti). Ce problème pourra être mitigé à l'avenir en étudiant les cartes avec des systèmes externes. Nous reviendrons sur ces systèmes dans l'ouverture de la conclusion de ce rapport.

Part VII
Résultats

Dans cette partie, nous présentons les différents résultats quant à la variation de la consommation des cartes, que nous avons pu présenter partie 4, en fonction de différents paramètres. L'organisation des sections de cette partie correspond grossièrement à l'ordre chronologique de nos découvertes lors de ce stage.

Tout d'abord, nous avons cherché à étudier l'impact du choix des types de cœurs pour l'exécution d'un même programme. Nous avons donc exploité le côté hétérogène de l'architecture de la x7ti et l'avons comparé à la consommation de l'imx93.

Suite à cette étude nous avons essayé de fixer une mesure d'étalonnage à laquelle il nous aurait été possible de nous référer pour pouvoir comparer les optimisations des différents systèmes. Mais un système qui exécute des programmes se voit appliqué par le kernel différentes optimisations en fonction de sa charge. DVFS en est une et impacte grandement la consommation des cartes étudiées, mettant en péril cette hypothèse. D'autres techniques d'optimisations - "power-gating" et "clock-gating" - peuvent être appliquées quand le système "ne fait rien" sur une période de temps donnée. Ces optimisations prennent la forme d'une pile logicielle nommé CPUIdle au sein du kernel. Nous reviendrons sur la définition d'un système qui "ne fait rien" ainsi que l'impact de ces états sur l'énergie dépensée.

Enfin, suite à l'étude de la catégorie de méthodes d'analyse empirique (voir section 6) nous avons entreprit d'étudier la consommation par instruction sur chacune des cartes en fonction de différentes opérande. Le but étant ensuite de pouvoir fournir ces résultats à notre méthode de régression.

14 Effet de l'architecture des cœurs sur la consommation

Coremark

Le benchmark CoreMark a été développé en 2009 par Shay Gal-On à l'EEMBC (Embedded Microprocessor Benchmark Consortium). Il vise à mesurer les performances des microcontrôleurs (MCU) et des unités centrales de traitement (CPU) utilisées dans les systèmes embarqués. Conçu pour remplacer le benchmark Dhrystone[27], CoreMark est devenu une norme industrielle pour évaluer les performances des microcontrôleurs.

CoreMark effectue plusieurs types de calculs, notamment :

[27] Weicker, "Dhrystone: a synthetic systems programming benchmark".

- **Traitement de listes** : opérations de recherche et de tri sur des listes (mergesort et find)
- **Manipulation de matrices** : opérations matricielles courantes. Multiplication de matrices à matrices, matrices à vecteurs et matrices à scalaires.
- **Machines à états** : vérification de la validité des nombres dans un flux d'entrée. Coremark implante un algorithme de traitement d'opérations algébriques sous forme d'une chaine de caractères.
- **Contrôle de redondance cyclique (CRC)** : Cet algorithme sert à la fois de charge de travail courante dans les applications embarquées et assure l'auto-vérification du bon fonctionnement du benchmark.

CoreMark est conçu pour fonctionner sur une large gamme de dispositifs, des microcontrôleurs 8 bits aux microprocesseurs 64 bits. Par défaut, ce benchmark réserve une place mémoire contigüe de 2 kilo-octets pour les données qu'il traite, c'est un choix optimal pour notre problème. Nous l'avons utilisé lors de nos premiers essais pour s'assurer de la nature de la charge appliquée sur les cartes.

Étude empirique du choix des cœurs sur la consommation

Afin de comparer l'impact du choix du cœur sur la consommation finale à attribuer à l'exécution d'un programme, nous avons exécuté le benchmark Coremark sur 2 des cœurs de chaque type : A55, LPE, E, P. Le premier est présent sur l'i.mx93 au contraire des 3 derniers qui ont été évalués sur la x7ti. La technologie DVFS était active sur lors de ce test. La figure 15 présente les résultats de cette expérience.

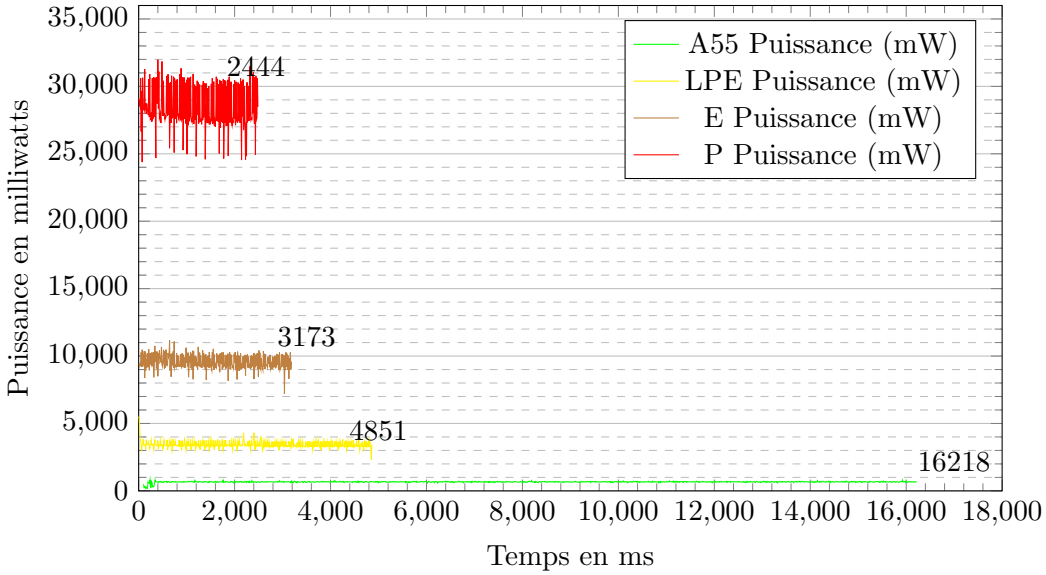


Figure 15: Puissance en fonction du temps pour une exécution de Coremark attachée sur 2 coeurs de type A55, LPE, E and P

Les étiquettes numérotées en face de chaque courbe représentent l'abscisse du dernier point de donnée, correspondant au temps total d'exécution du benchmark pour le type de coeur en millisecondes.

En complément de ce diagramme, les données suivantes sont jointes :

Le test montre plusieurs choses intéressantes. La première, illustrée par la figure 17, est que l'écart-type de la mesure de puissance croît avec la valeur mesurée. C'est un point que Walker et al., "Accurate and Stable Run-Time Power Modeling for Mobile and Embedded CPUs" abordent à la fin de la présentation de leur modèle : la variance de la mesure de la puissance n'est pas constante.

Type de Cœur	Puissance instantanée moyenne (mW)	énergie totale dépensée (J)
A55	664	10,7
LPE	3431	16,1
E	9560	29,4
P	28419	68,3

Table 4: Puissance instantanée moyenne et énergie en fonction des cœurs pour l'exécution de Coremark

La seconde confirme qu'un temps d'exécution plus long, pour un même programme, ne correspond pas à une consommation en énergie plus importante. C'est même l'inverse. Le graphique 16 montre que plus le cœur est performant d'un point de vue temporel, plus il consomme d'énergie. C'est la raison de cette différence que nous asseyons d'expliquer. Cependant, nous pouvons donner quelques pistes :

- Le cœur A55 a une microarchitecture "in-order" comparé à l'ensemble des cœurs x86 mentionnés.
- La finesse de gravure est plus petite pour la x7ti (16nm versus 7nm). Augmenter la finesse de gravure résulte en une augmentation drastique de la consommation statique des cœurs.[28].
- La taille des caches varie grandement entre les cœurs. Il n'est pas impossible que la consommation statique soit d'autant plus grande que le cache possède de place.

Malgré cela, les graphiques montrés dans cette section font un état global du profil de consommation des différents cœurs à notre disposition. Les considérant, il est possible d'établir une échelle sur la propension à consommer des cœurs que nous avons à notre disposition. Pour une même charge C qui prend un nombre d'instruction fini, nous pouvons supposer avoir la relation d'ordre

$$Ej_{A55} < Ej_{LPE} < Ej_E < Ej_P$$

où les applications Ej représentent la consommation réelle en joules des coeurs en fonction des charges. Nous se basons sur cette hypothèse pour la suite des résultats en faisant fluctuer certains paramètres pour voir si elle est remise en jeu.

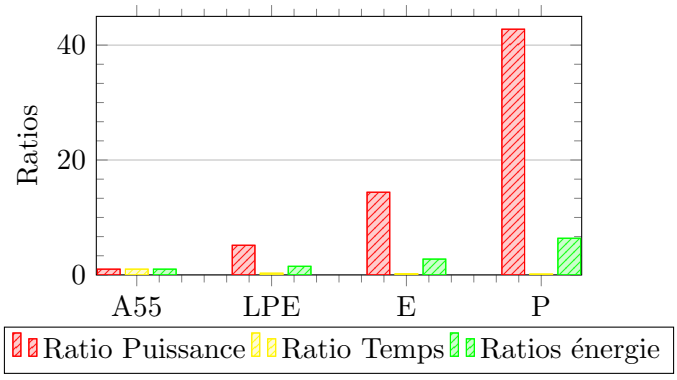


Figure 16: Ratio d'énergie, de temps et de puissance entre les différents types de coeurs et le coeur A55

15 DVFS - CPUfreq

CPUFreq est une interface logicielle implantée dans le noyau Linux pour contrôler la fonctionnalité de Dynamic Voltage and Frequency Scaling (DVFS) implantée physiquement sur les puces nouvelles générations. C'est une des techniques classique utilisée pour réduire la consommation de puces (section 9[29]), à l'instar du "clock-gating" ou du "power-gating". Cette fonctionnalité a grandement

[28]Keating, *Low Power Methodology Manual*.
[29]Keating, *Low Power Methodology Manual*.

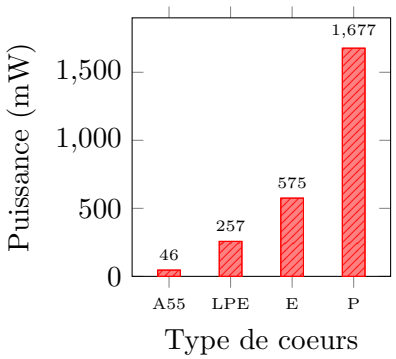


Figure 17: Ecart-type de la mesure de puissance

impacté les résultats que nous avons eu lors de ce stage. Certaines méthodes prennent d'ailleurs en compte le DVFS comme un paramètre du modèle d'estimation de consommation[30] (ou du modèle à sélectionner). Étrangement, cette fonctionnalité n'est pas disponible sur l'i.mx93 (voir échange dans le forum spécialisé de NXP[31]) : les données exposées dans cette section ne concerneront donc que la x7ti.

Les graphiques de la section précédente ne prenaient pas en compte le DVFS, que nous décrivons dans la suite de cette section. Hors, cette technologie est un acteur important dans la consommation des puces que nous étudions comme nous allons le voir. Sachant cela, les tests suivants avaient pour but de vérifier la relation d'ordre précédemment évoquée en précisant que $\forall c \in \{A55, LPE, E, P\} Ej_c(p) = f_c(freq, p)$ où f_c est une application dépendant de la fréquence moyenne de fonctionnement du cœur $freq$ pendant la charge du programme p.

CPUFreq

La pile logicielle de CPUFreq comprend plusieurs composants : le core, qui enregistre les cœurs en fonction de leur disponibilité et associe des politiques à chacun d'entre eux ; le governor, qui manipule les données pour arbitrer la fréquence à choisir ; la politique, qui répartit les décisions du governor à tous les CPUs associés ; et le driver, qui discute avec le hardware pour récupérer et appliquer les fréquences demandées. Ce fonctionnement est illustré dans la figure suivante.

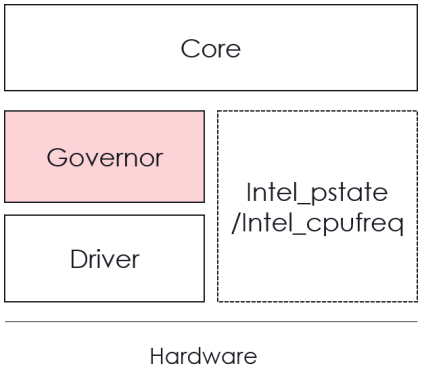


Figure 18: Diagramme bloc du fonctionnement de CPUFreq

Théoriquement, il est possible d'utiliser n'importe quel governor sur n'importe quel hardware et d'utiliser différents governors pour différents sous-ensembles de CPU logiques.

Intel a développé son propre driver et governor, nommé "intel_pstate", pour optimiser la gestion de la fréquence des processeurs à partir de l'architecture Sandy Bridge. Ce driver supporte la technologie "Hardware-managed P-states", permettant une gestion automatique des états de fréquence

[30] Mazzola et al., *Data-Driven Power Modeling and Monitoring via Hardware Performance Counters Tracking*.
[31] *Discussion sur le forum de NXP parlant de DVFS*.

du processeur. Le driver "intel_pstate" fonctionne en deux modes : actif et passif. En mode actif, la fréquence ne peut pas être sélectionnée par l'espace utilisateur, et les governors disponibles sont "powersave" et "performance". En mode passif, le driver fonctionne comme un driver classique de CPUFreq, permettant la sélection des governors disponibles dans le noyau Linux.

Intel propose également une technologie appelée EPB (Energy and Performance Bias), permettant de donner des orientations sur une échelle de 16 valeurs entre "performance" et "power". Ces valeurs sont utilisées par le governor "intel_pstate" pour arbitrer les fréquences.

Pour des questions de compatibilité entre différentes plateformes, nous désactivons ce driver spécifique à Intel. Son fonctionnement n'est pas connu, ses politiques de gestion non plus et les plateformes ne sont pas garanties d'avoir la même version de ce driver. Nous préférons donc utiliser le driver et fonctionnement classique de CPUfreq, bien qu'il soit sous-optimal dans certains cas.

Étude empirique de l'impact de DVFS sur la consommation

La figure 19 suivante reprend l'exécution de Coremark sur 2 cœurs P. Tout comme la première exécution, le temps total d'exécution du logiciel est présenté sur le graphique par les étiquettes au-dessus de chacune des séries.

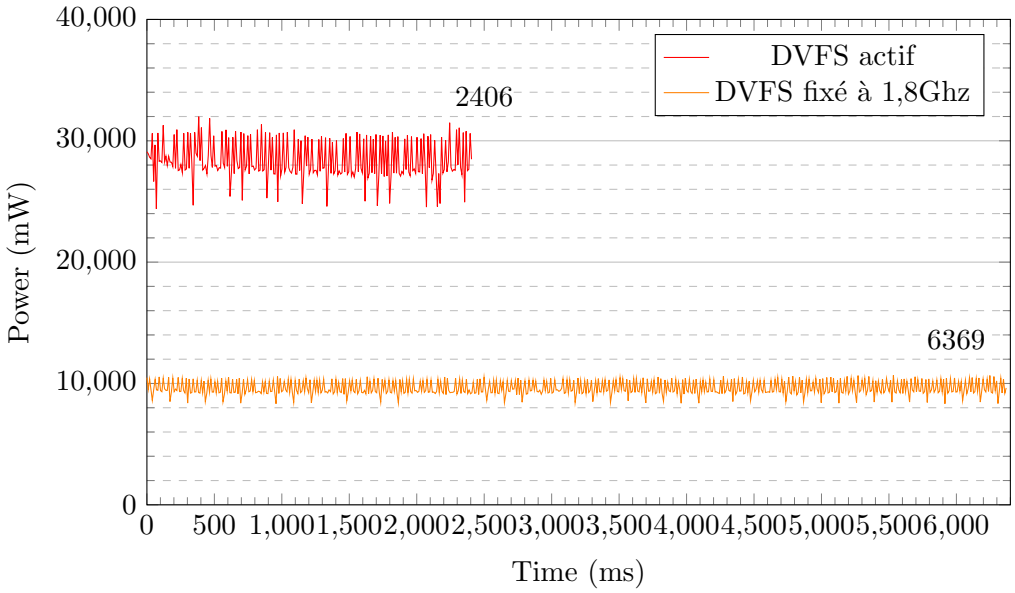


Figure 19: Exécution de cormark sur 2 coeurs P à fréquence variable et fixée

Pour compléter la figure ci-dessus, le tableau suivant est donné :

Etat de DVFS	Puissance instantanée moyenne (mW)	énergie totale dépensée (J)
DVFS actif	28418	68,3
DVFS fixé à 1,8Ghz	9626	61,3

Table 5: Puissance instantanée moyenne et énergie en fonction des coeurs pour l'exécution de Coremark

Considérant cette dernière table, l'hypothèse proposée en préambule peut être infirmée. Pour une même charge sur un même cœur, la consommation est différente en fonction de la fréquence de celui-ci.

Cela veut aussi dire qu'il n'est pas possible de comparer, si le souhait est de comparer l'efficacité énergétique intrinsèque des programmes (donc indépendamment de la fréquence de fonctionnement du cœur), l'exécution de deux programmes sans à minima fixer ou prendre en compte le DVFS dans les mesures. Hors, DVFS est particulier à chaque plateforme : établir des comparaisons entre un état X du côté ARM et x86 - quand c'est faisable - serait malvenu car chacun a sa définition de l'état. De ce fait, si il est impossible de les comparer, il est possible d'étudier les systèmes lorsque les DVFS sont

inactifs , c'est à dire regarder la consommation des coeurs lorsqu'il ne font "rien". Cette étude nous a amené à chercher du côté de CPUIdle.

16 CPUIdle - Etats de sommeil

À la recherche d'un étalon de mesure entre les 2 cartes nous avons essayé de comparer les deux cartes à charge nulle, c'est à dire lorsqu'elle ne faisait rien. Nous pensions implanter un simple programme d'attente passive comme "sleep", et mesurer la consommation pendant le temps d'exécution de la commande de sortes à avoir une borne inférieure pour nos mesures de consommation. Ainsi nous aurions pu quantifier le potentiel de gain en énergie des programmes étudiés, même grossièrement, en normalisant les données de consommation entre cette valeur et une valeur maximale, donnée lorsque le processeur a une charge supportable maximale. Une fois cette mesure effectuée, nous aurions aussi pu la faire varier selon différents états de DVFS pour catégoriser le gain potentiel.

Cependant, nous allons le voir dans les graphiques suivants, le comportement du processeur lorsqu'il n'a rien à faire - c'est à dire sans tâches à ordonnancer - dépend des plateformes étudiées et de certains paramètres.

Là où CPUFreq intègre le fonctionnement du DVFS dans le kernel Linux, CPUIdle se charge du fonctionnement de mise en veille des unités de calcul principales.

CPUIdle

CPUIdle est une pile logicielle du noyau Linux qui gère les états de veille du processeur. CPUIdle permet au processeur lorsqu'il n'exécute pas d'instructions de désactiver certaines parties de son circuit pour économiser de l'énergie. La pile logicielle de CPUIdle est similaire à celle de CPUFreq, avec des concepts de gouverneurs et de pilotes. Cependant, les états de sommeil dans CPUIdle sont clairement définis et peuvent correspondre aux C-states de l'ACPI[32].

Un état de sommeil est défini par un couple de valeurs : la résidence cible (le temps minimal de résidence dans l'état de sommeil pour économiser plus d'énergie) et la latence de sortie (le temps maximal entre la demande d'entrée en sommeil et la demande d'exécution d'une autre instruction).

Le gouverneur CPUIdle est activé lorsqu'une tâche spécifique, nommée "idle" ou "swapper", est mise dans l'état "exécutable" lorsqu'aucune autre tâche ne peut être sélectionnée. Cette tâche va progressivement entrer dans des états de sommeil de plus en plus profond (c'est à dire dont le temps d'entrée et de sortie sont de plus en plus longs). Cependant, le CPU est constamment réveillé par les ticks du scheduler (1-10 ms), ce qui empêche d'entrer dans des états de sommeil trop profonds. Certains noyaux peuvent être "tickless", c'est-à-dire qu'ils arrêtent l'horloge de l'ordonnanceur et ne se réveillent que lors d'une interruption autre.

Il est possible de désactiver tous les états de sommeil, ce que nous avons fait pendant les expériences qui suivent. Même dans ce cas, cependant, un état "C0" reste actif et contient une boucle de traitement ("poll loop") qui diffère en fonction des plateformes mais qui permet de réaliser une attente très rapide. Historiquement, cette boucle contenait une suite de "nop" (le cas sur ARMv6 par exemple). C'est encore le cas sur la x7ti.

Dans notre cas, le coeur A55 est sous ARMv8.1 et possède une boucle d'attente passive grâce à un mécanisme physique. Les opérations instructions pouvant activer cette attente passive sont "WFI" (Wait For Interrupt) et "WFE" (Wait For Event). La boucle d'attente principale de C0 sur A55 ne contient qu'un "WFI".

Étude empirique de l'impact de CPUIdle sur la consommation

Sachant cela, nous avons essayé de comparer la consommation de l'exécution d'une commande "sleep" sur 5 secondes à celle de Coremark avec les états de mise en veille actifs et désactivés. Le résultat est disponible dans les figures suivantes :

[32] *Site de l'ACPI sur les états de processeur.*

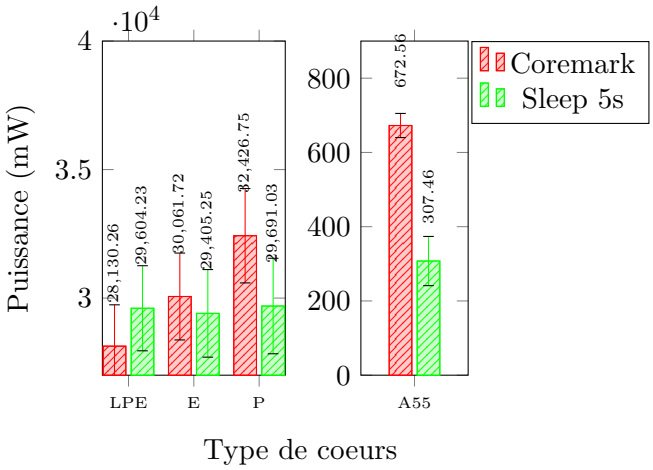


Figure 20: Puissance instantanée moyenne de l'exécution de Coremark et d'un Sleep de 5s sur différents types de coeurs, avec état CPUIdle inactifs. Les barres d'erreur représentent les écarts-types des échantillons

Coeur	Ecart-type (mW)	
	Coremark	Sleep
A55	33	67
LPE	1608	1655
E	1690	1705
P	1836	1858

Figure 21: Ecart-type de la figure 20

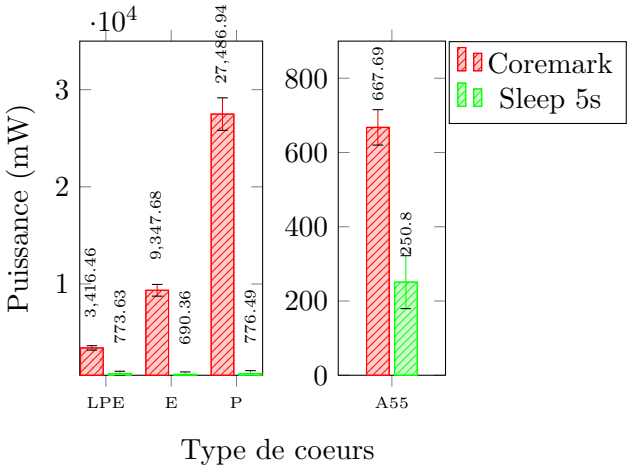


Figure 22: Puissance instantanée moyenne de l'exécution de Coremark et d'un Sleep de 5s sur différents types de coeurs, avec état CPUIdle actifs. Les barres d'erreur représentent les écarts-types des échantillons

Coeur	Ecart-type (mW)	
	Coremark	Sleep
A55	48	72
LPE	227	242
E	604	655
P	1674	298

Figure 23: Ecart-type de la figure 22

Il est important de rappeler que ces valeurs de puissance sont prises sur l'ensemble des cœurs pour chaque puce et que les états sont désactivés pour chacun des cœurs sur chaque puce. De plus, l'exécution d'une commande "sleep", même attachée au cœur concerné par le test, amène la tâche en exécution à être considérée comme endormie. Ce que l'on mesure donc réellement en exécutant la commande "sleep" sur un cœur en particulier c'est la consommation des tâches de fond qui s'y exécutent (des démons, des tâches ordonnancées depuis d'autres cœurs, etc). De futurs travaux se concentrerons à évaluer la charge de telles tâches au sein du cœur afin d'avoir une attribution plus fine de la consommation.

De ces tests nous pouvons tirer les analyses suivantes :

- L'attente passive d'ARM est efficace énergétiquement comparée à l'attente active d'Intel. Là où la puissance instantanée lors d'un sleep est équivalente (voire supérieure !) à celle lors d'une charge comme Coremark sur x86_64, sur ARM la puissance instantanée est divisée par 2.

- Les écarts-types lors des sleep sont légèrement plus élevés (une exception pour l'exécution sur cœur P avec états de sommeil actifs). Cela peut s'expliquer par le fait que d'autres tâches peuvent être amenées à être exécutées sur les cœurs étudiés, et donc faire varier leurs utilisation.

- On peut remarquer une certaine tendance à l'augmentation de l'écart-type des données de puissance instantanée, comme précédemment mentionné, en fonction de la valeur mesurée. Des tests supplémentaires seraient à conduire pour confirmer cette hypothèse.

En conclusion, les états de mise en veille ont un impact non négligeable sur la puissance instantanée. On voit que pour l'exécution d'une charge définie non nulle, la puissance instantanée augmente significativement sur x86 - au moins 5 watts - mais beaucoup moins sur l'imx93 - 5 milliwatts si l'on considère qu'une telle différence n'est pas comprise dans l'erreur. Sur x86_64, l'exécution d'une charge nulle (ou presque) résulte en 2 comportements très différents en fonction de l'usage ou non des états de mise en veille : dans un cas la puissance instantanée est comparable à celle de l'imx93 lorsqu'elle exécute coremark, dans l'autre elle est comparable à l'exécution de coremark sur x7ti, soit un facteur 38 de différence. Cette différence est réduite sur ARMv8.1 avec un facteur de 1,2 entre sans et avec CPUIdle.

Ces résultats sont à pondérer par le fait que les 2 cartes n'ont pas la même architecture ni le même but. Le nombre de cœurs influe certainement ici : ne pas éteindre les 22 cœurs de la x7ti est difficilement comparable à ne pas éteindre les 2 cœurs de l'imx. Cependant, ils marquent la difficulté à trouver un étalon de mesure entre les 2 cartes : désactiver CPUIdle reviendrait à faire des mesures avec une incertitude de plus d'1.5watts sur x7ti et ne pas le faire pourrait revenir à comparer 2 systèmes ayant des propriétés fondamentalement différentes (notamment en réactivité, puisque c'est ce que ces états garantissent). Une hypothèse maximaliste ne fonctionnera donc probablement pas dans notre cas. De prochains travaux porterons sur l'intégration de ces paramètres dans le modèle final.

17 Différences entre instructions

Un programme est composé d'instruction. Une façon d'étudier sa consommation énergétique est donc d'étudier les instructions qui le composent. La difficulté de cette technique est dans la combinaison d'options possibles : Plus de 3000 instructions dans l'ISA (Instruction set Architecture) d'Intel, plusieurs centaines côté ARMv8.1 en fonction des options.

En s'inspirant des techniques décrites dans la partie 6, nous avons sélectionné un sous-ensemble d'instructions restreint et avons tenté d'évaluer leur impact en fonction des opérandes sélectionnées. Nous décrivons dans cette section les résultats de cette analyse.

Cadre d'expérience

Les résultats que nous montrons dans la section suivante ont été mesurés en suivant la méthode décrite dans les figures 14 et 12. Les données ont été moyennées sur 10 tentatives identiques espacées de 3 secondes de tampon. Ces tentatives étaient composées de 10 milliards d'instructions pour la plateforme x86, et 1 milliard pour l'i.mx93 (pour limiter le temps d'exécution principalement).

Dû au fait qu'un exécutable contenant 10 milliards d'instructions soit exceptionnellement gros, les tests incluent des boucles. Nous avons cependant essayé de limiter au maximum leur usage, pour limiter l'utilisation d'instructions de branchement, qui pollueraient le résultat final. Pour la x7ti, le nombre de branchement s'élève à 5000 pour un total de 10^9+5000 instructions, soit un ratio (théorique) de $5 * 10^{-6} = 0,005\%$ instructions de branchement sur le nombre d'instruction total. Pour l'imx93, ce ratio est plus grand - car le compilateur ARM impose qu'un Basic Block soit inférieur à une certaine taille en nombre d'instructions - car le nombre de branchements s'élève à 10000, soit un ratio final d'environ $10^{-4} = 0,1\%$.

Enfin, plusieurs tests ont été effectués par instruction. Un descriptif de chacun des tests est affiché dans la table suivante :

Nom	Description
Max	Faire une opération OP avec la valeur maximale acceptée par l'opération. Les Shifts sont limités à 32, les autres opérations retombent sur la valeur maximale d'un entier signé acceptable par plateforme.
Zero	Homologue au test précédent, mais avec la valeur "minimale"
Alternate	Alternance d'opération utilisant l'opérande maximale et 0
Random	Opération avec un immédiat aléatoire constant compris dans les valeurs acceptables de l'opération.

Table 6: Description des différents tests de la section 17

Ces tests sont grandement inspirés de Bazzaz, Salehi, and Ejlali, "An Accurate Instruction-Level Energy Estimation Model and Tool for Embedded Systems". Les auteurs justifiaient une différence entre la consommation des instructions atomiques en fonction du nombre de bits changés (décrits comme une distance de Hamming entre les encodages binaires des instructions) à chaque opération, et donc des opérandes. Ainsi en effectuant ces tests nous nous attendions à plusieurs choses : les tests "alternate" devaient mener à une consommation plus grande que les autres, les tests "zero" devaient minorer les ensemble (optimisation de certaines instructions par rapport à cette valeur), les tests "random" devaient éviter la plupart des optimisations matérielles de base et donc mener à une consommation plus grande que celle des "zero" et les "max" ont été ajoutés pour voir si une tendance particulière apparaissait avec cette valeur - overflow à répétition par exemple, entraînant une surconsommation.

Étude empirique de consommation par instruction

Le résultat des expériences décrites dans la figure 24 ci-dessous. Pour ces expériences, l'ensemble des cœurs de la x7ti ont étés fixés à une fréquence de 1.8 Ghz et les états de mise en veille ont été désactivés. Comme dit précédemment, le cœur A55 ne possède pas de DVFS, il tourne à une fréquence nominale de 1,7Ghz. Enfin, chaque test a été attaché à un cœur de chaque type. De ces graphiques nous pouvons tirer les observations suivantes :

- L'architecture ARM est plus économe que x86 à fréquence quasi égale.
- A fréquence fixe, plus le cœur est performant en termes de puissance de calcul sur x86 (LPE puis E puis P), plus l'énergie par instruction est faible. Cela peut s'expliquer par un débit d'instruction plus important sur les cœurs plus puissants. Certaines instructions échappent cependant à la règle (comme CMP). Pour confirmer cette hypothèse, il serait possible de faire varier la fréquence sur chacun des cœurs et d'observer les résultats.
- Les hypothèses que nous avions avant le lancement de ces tests ne sont pas consistantes en fonction des cœurs. Le cœur P n'observe que très peu de variabilité sur l'ensemble des tests de ses instructions, alors que le cœur LPE voit une nette variation entre les différents tests sur les instructions ADD et SUB.
- Une différence est en effet constatée entre le test "alternate" et "zero" sur le cœur LPE pour les instructions ADD et SUB. Au vu des données sur les autres cœurs cependant, il n'est pas possible d'en supposer une relation quelconque. La différence pourrait s'expliquer par la répartition d'instruction avec opérande 0 et d'instructions avec opérandes maximale - la différence entre la moyenne du test alternate et celle du test zero ainsi que celle du test alternate et du test max sont quasiment égale.
- L'opération MUL est généralement plus consommatrice que les autres (au moins 83%) pour x86. Cette différence peut s'expliquer par le temps d'exécution de cette instruction par rapport aux autres. Cela conforte la classification de Mazzola et al., *Data-Driven Power Modeling and Monitoring via Hardware Performance Counters Tracking*.

- La multiplication sur ARM est beaucoup moins consommatrice que celle sur x86 relativement aux autres instructions étudiées.
- En fonction des architectures des instructions pourraient être considérées comme équivalente en profil de consommation. Les shifts sont un bon exemple : la consommation par instruction reste constante en fonction du test et du sens de shift. Sur le coeur LPE, une instruction CMP pourrait être considérée équivalente à une instruction ADD ou SUB avec une certaine tolérance.

En général, les effets observés par Bazzaz, Salehi, and Ejlali, "An Accurate Instruction-Level Energy Estimation Model and Tool for Embedded Systems" ne sont pas confirmés via ces observations. La différence peut venir de plusieurs endroits : finesse de gravure, évolution de la technologie (plus de 10 ans d'écart entre aujourd'hui et l'émission du papier). Cependant, catégoriser les instructions en fonction de leur type, et appliquer une analyse statique en fonction de ces groupes pourrait amener à de bons résultats. C'est ce que préconisaient Mazzola et al., *Data-Driven Power Modeling and Monitoring via Hardware Performance Counters Tracking*.

Architecture	Instruction	Mnemonics	Min	Max	Opérandes	Valeurs des opérandes			
						Max	Zero	Alternate	Random
x86	ADD	add	0	$2^{31}-1$	imm, rd	imm=Max	imm=0	alternance entre imm=max et imm=0	imm=random
x86	SUB	sub	0	$2^{31}-1$	imm, rd	imm=Max	imm=0	alternance entre imm=max et imm=0	imm=random
x86	MUL	mul	0	$2^{31}-1$	rs	rs=Max	rs=0	rs=Max, 0	rs=random
x86	NOP	nop	0	$2^{31}-1$					
x86	SHIFT	shl, shr	0	32	imm, rd	imm=32	imm=0	imm=32, 0	imm=random(0-32)
x86	CMP	cmp	0	$2^{31}-1$	imm, rd	imm=Max	imm=0	imm=Max, 0	imm=random
x86	MOV	mov	0	$2^{31}-1$	imm, rd	imm=Max	imm=0	imm=Max, 0	imm=random
ARM	ADD	add	0	4095	rd, rs, imm	imm=Max, rs=random	imm=0, rs=0	imm=Max, 0, rs=Max, 0	imm=random, rs=random
ARM	SUB	sub	0	4095	rd, rs, imm	imm=Max, rs=random	imm=0, rs=random	imm=Max, 0, rs=Max, 0	imm=random, rs=random
ARM	MUL	mul	0	65535	rd, rs, rx	rs=Max	rs=0	alternance entre rs=max et rs=0	rs=random
ARM	NOP	nop	0	65535					
ARM	SHIFT	lsl, lsr	0	32	rd, rs, imm	imm=32, rs=Max	imm=0, rs=random	imm=32, 0, rs=Max, 0	imm=random(0-32), rs=random
ARM	CMP	cmp	0	65535	rd, rs	rs=Max	rs=0	alternance entre rs=max et rs=0	rs=random
ARM	MOV	mov	0	65535	rd, imm	imm=Max	imm=0	alternance entre imm=max et imm=0	imm=random

Table 7: Description des opérandes d'instructions pour chacun des cas mentionnés table 6

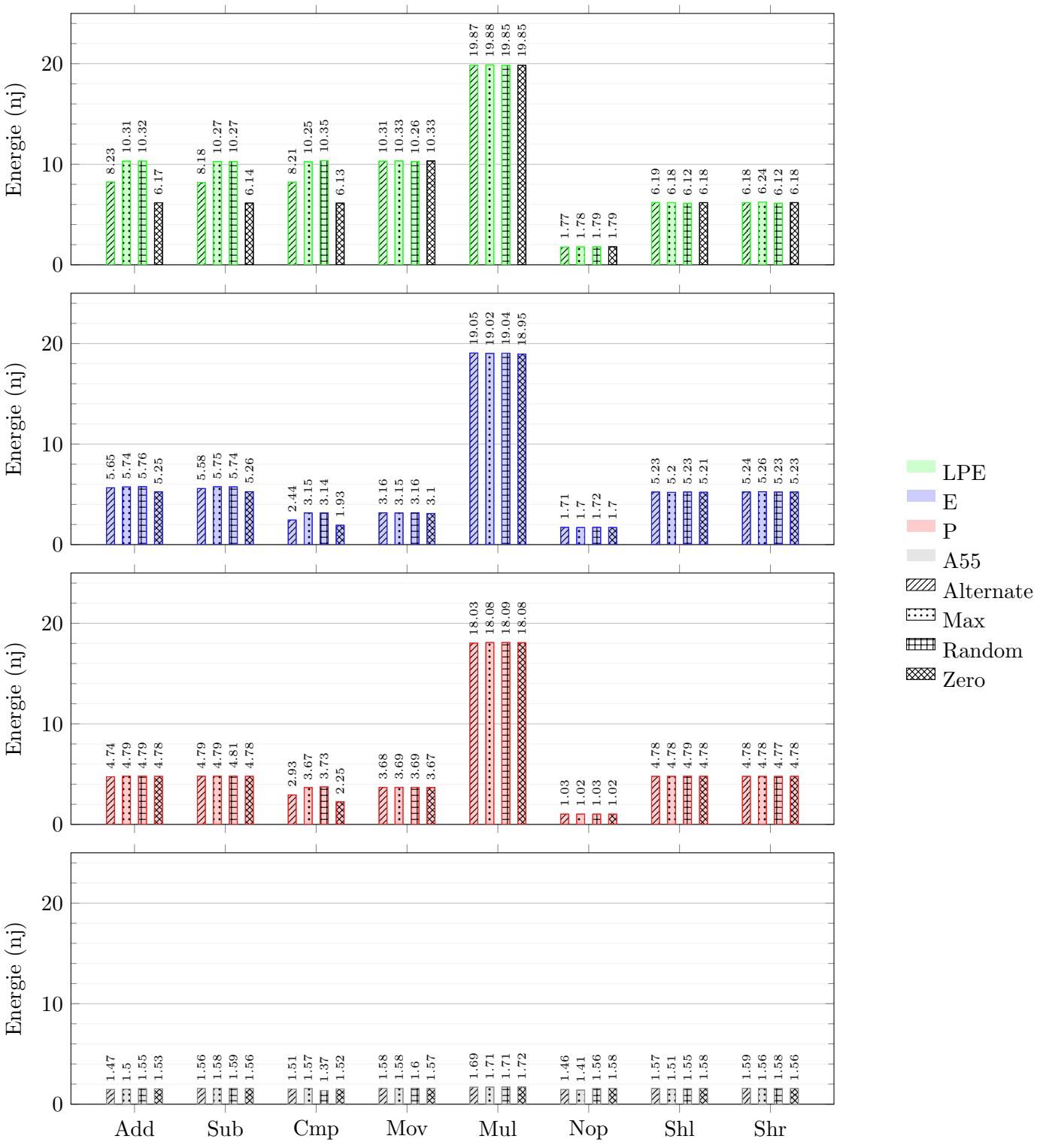


Figure 24: Comparaison de la consommation par instructions en fonction du test effectué et du coeur sur lequel il est exécuté

18 Modèle de régression

Le temps ne nous aura pas permis d'avoir un modèle de consommation fonctionnel lors de ce stage. Cependant, le reste à faire s'est grandement amenuisé et nous espérons en avoir un efficace sous peu. Des premiers résultats ont pu être produits via le modèle que nous avons décrit mais ils ne sont, pour l'instant, pas convaincants.

Nous pensons que c'est principalement dû à 2 choses : le raffinement pro régression des données et le type de modèle que nous utilisons. Pour l'instant, nous n'avons pas du tout mis en place de système de raffinement comme on pu le faire d'autres auteurs. Certes, notre démarche de sélection des PMCs pour la prédiction a été de récupérer les optimaux des papiers de la littérature, mais des méthodes plus précises pourraient être mises en place. De plus, il est fort peu probable qu'un simple modèle linéaire conviendrait dans notre cas ($\sum_{i=0}^{N} \alpha_i * x_i$) puisque la définition de la puissance dynamique n'est elle-même pas linéaire en fonction de ses paramètres. Enfin, la régression OLS est certes la plus simple, mais il est fort à parier que ce ne soit pas la meilleure dans notre cas.

Part VIII
Avancement

Cette partie reprends la chronologie du stage et permet au lecteur de situer nos avancés et défis, précédemment mentionnés, dans le temps.

La figure 25 suivante reprends la chronologie du stage. Les traits bleus représentent des périodes de thématiques du stage, correspondants en partie à l'avancement par rapport aux hypothèses et pistes de recherches exposées dans la partie VII. Les tâches vertes représentent les périodes de tests et les diamants noirs des jalons importants. Les diamants rouges représentent des pivots dans le stage qui ont impliqué un changement de direction drastique. Enfin, ce diagramme ne tient pas compte du travail de lecture bibliographique qui a été effectué tout au long du stage, car comme la première tâche du graphique, c'était une tâche de fond qui a commencé au début du stage et ne s'est jamais vraiment arrêtée.

Le diagramme commence au moment où mes connaissances sur le sujet d'étude ont commencées à me permettre de récupérer des données et tracer quelques graphiques (que vous avez pu observer partie 14). Nous voulions stabiliser et surtout expliquer la mesure que nous recevions. De ce fait j'ai installé un OS minimal (construit avec YOCTO) sur l'i.mx93 pour élaguer tout ce qui n'était pas nécessaire au fonctionnement de nos benchmarks et des outils de prise de mesure. En parallèle je commençais à me renseigner sur le DVFS et CPUIdle que nous avons pu aborder section précédente. Le premier pivot affiché avant ces deux premières périodes représente le choix fort que nous avons fait de laisser de côté, pour ce stage, tout ce qui ne touchait pas à l'utilisation des CPUs.

La période qui s'en suivit était dans la continuité de cette volonté d'explication. En chargeant nos programmes d'analyse du système nous avions un impact significatif sur la mesure d'énergie. Nous avons donc entrepris d'estimer le coût moyen de nos outils de mesure (eBPF à ce moment-là) sur les 2 plateformes pour pouvoir l'écarter de nos mesures futures.

Une fois cette tâche accomplie, et dû à la lecture de plusieurs papiers portant sur des méthode empiriques d'estimation, nous avons estimé le coût énergétique pour quelques instructions sur les deux plateformes. Cela nous a mené aux résultats constatés partie 17.

Rapidement après cette étape, nous avons bifurqué à nouveau de cible, qui reste finalement à ce jour inchangé. La description de cette cible a fait l'objet d'une partie de ce rapport, la partie 11. L'objectif était, encore une fois, de réduire le problème à quelque chose d'atteignable lors du stage.

S'en est finalement suivi des suites de tests et la rédaction du présent rapport.

La planification précédemment présentée dans le rapport de mi stage est bien différente de celle finalement achevée. Cela s'explique par le fait que le sujet traité est vaste, impacte beaucoup de domaines du développement logiciel comme matériel (système, microarchitecture, statistiques, etc) sur lesquels il a fallu monter en compétences et apprendre. Une autre difficulté de ce sujet a été l'hétérogénéité des plateformes étudiées, augmentant drastiquement la charge de travail. Enfin, aucun consensus scientifique quant à une méthode de mesure généralisée indépendante de la plateforme et du programme de l'énergie n'existe encore à ce jour. De ce fait, un gros travail bibliographique a été effectué pour récupérer et analyser les méthodes de différents bords afin de trouver celle qui con-

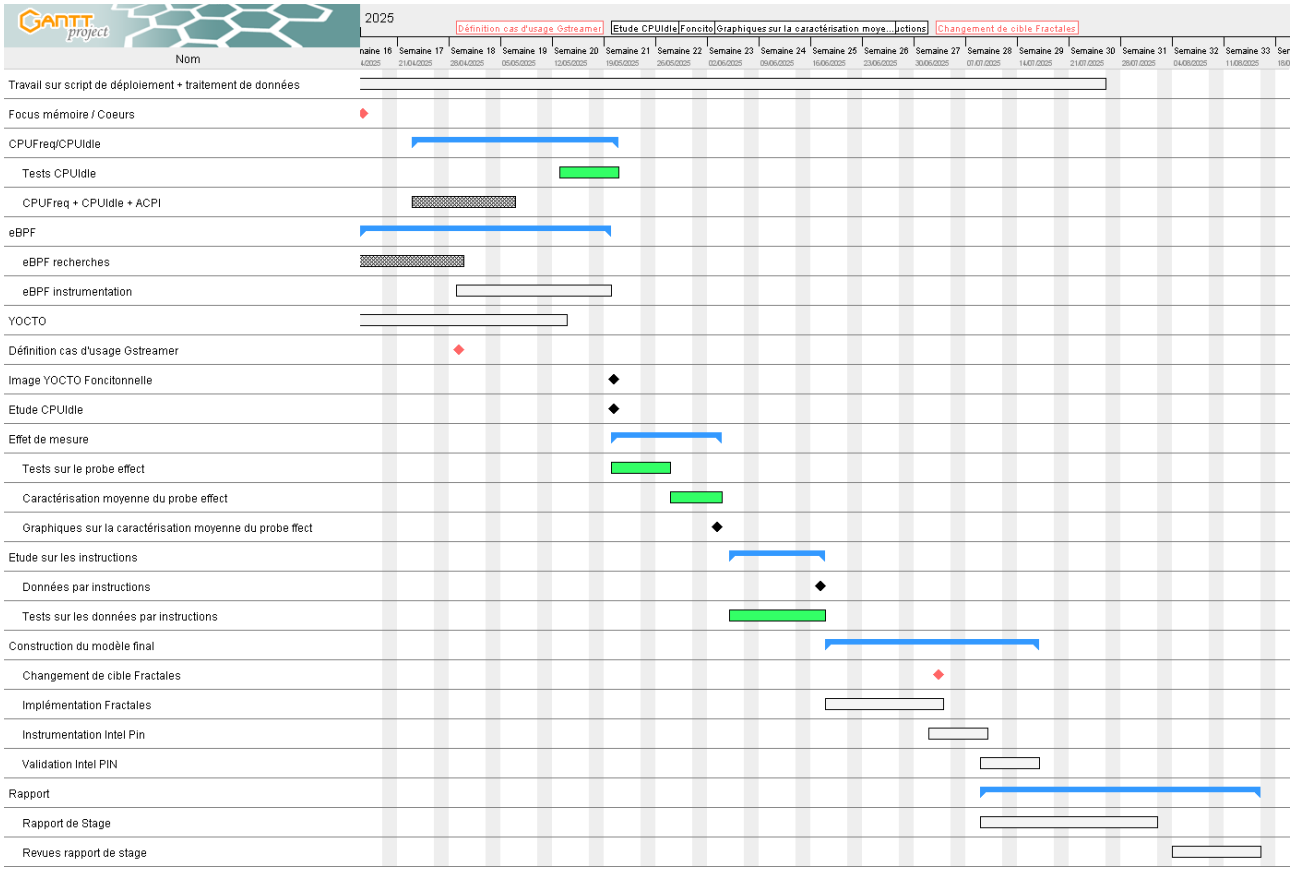


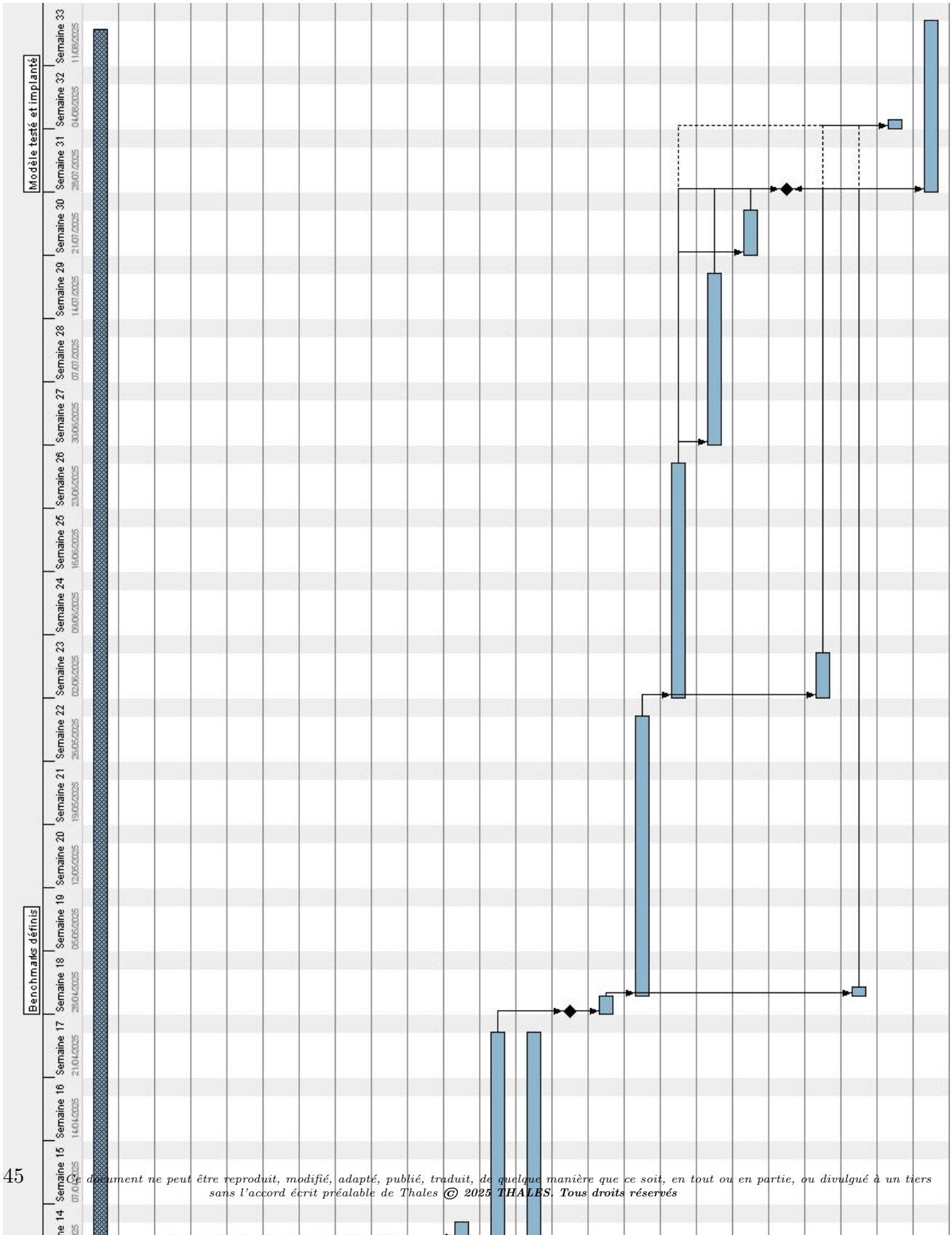
Figure 25: Planning final effectif du stage

viendrait le mieux au cas d'étude.

La table 8 suivante reprends les tâches de la précédente planification, affichée figure 26, pour laisser au lecteur la possibilité de comparer au stage effectué.

Titre	Description
Exploration bibliographique	Tâche de fond qui consiste à se renseigner sur l'état de l'art de l'estimation d'empreinte carbone des logiciels
Production de résumés d'outils	Sélection de certains outils pour les résumer dans des documents afin de les rendre plus accessibles aux membres concernés par le stage
Étude des outils de mesures de consommation	Recherches sur les différents outils de traçage de consommation dynamique disponibles depuis Linux et autre
Prise en main de l'écosystème de développement Thales	Installer les outils de développement, faire les demandes de matériel nécessaires
Prise en main de l'i.mx93	Comprendre comment interagir avec la carte, la façon dont elle démarre, etc
Pris en main du Monolithe	Système disposant de l'AtomMan x7ti. Plus d'information sur le site web du Monolithe[33]
Prise en main de l'AtomMan x7ti	
Exécution du benchmark Dhrystone sur les cartes	Compilation et développement de petits programmes pour le lancement du benchmark sur les cartes
Compréhension de l'alimentation de l'i.mx93	Analyse des datasheets du processeur ainsi que des différents rails de puissance pour savoir à quoi correspondent les valeurs renvoyées.
Rencontre avec certains architectes côté Thales pour choisir les cas d'usage	Discussion avec certains Architectes Software côté Thales pour avoir une idée de l'utilisation des cartes étudiées dans un cadre de production
Traitement du résultat de Dhrsytone sur les 2 cartes	
Définition des benchmarks à utiliser ou développement de ceux nécessaires	Recherche de benchmark qui correspondent aux cas d'utilisation des cartes chez Thales. Développement de certains dans le cas où ils n'existent pas
Benchmarks définis	Point d'étape
Traiter les données sorties des benchmarks	
Recherche d'un modèle empirique pour l'explication de la provenance des parties de consommation	Étude de la littérature et des données des tests exécutés pour les benchmarks afin d'en tirer un modèle mathématique/logique de la consommation des différentes portions de la carte
Implantation du modèle empirique	Implanter le logiciel qui prendra en entrée un logiciel et qui produira une consommation par analyse dynamique de celui-ci.
Développement d'une plateforme de déploiement du modèle	Développement d'un logiciel permettant de déployer facilement le code du modèle
Test du modèle sur des codes arbitraires	Prouver que le modèle fonctionne bien en le comparant à d'autres modèles sur des tests arbitraires.
Modèle testé et implanté	Point d'étape
Documentation du modèle	
Documentation des données reçues des expérimentations	
Test du modèle sur d'autres architectures	Tâche optionnelle dans le cas où le temps le permettrait
Rédaction du rapport final	

Table 8: Description des tâches de la figure 26



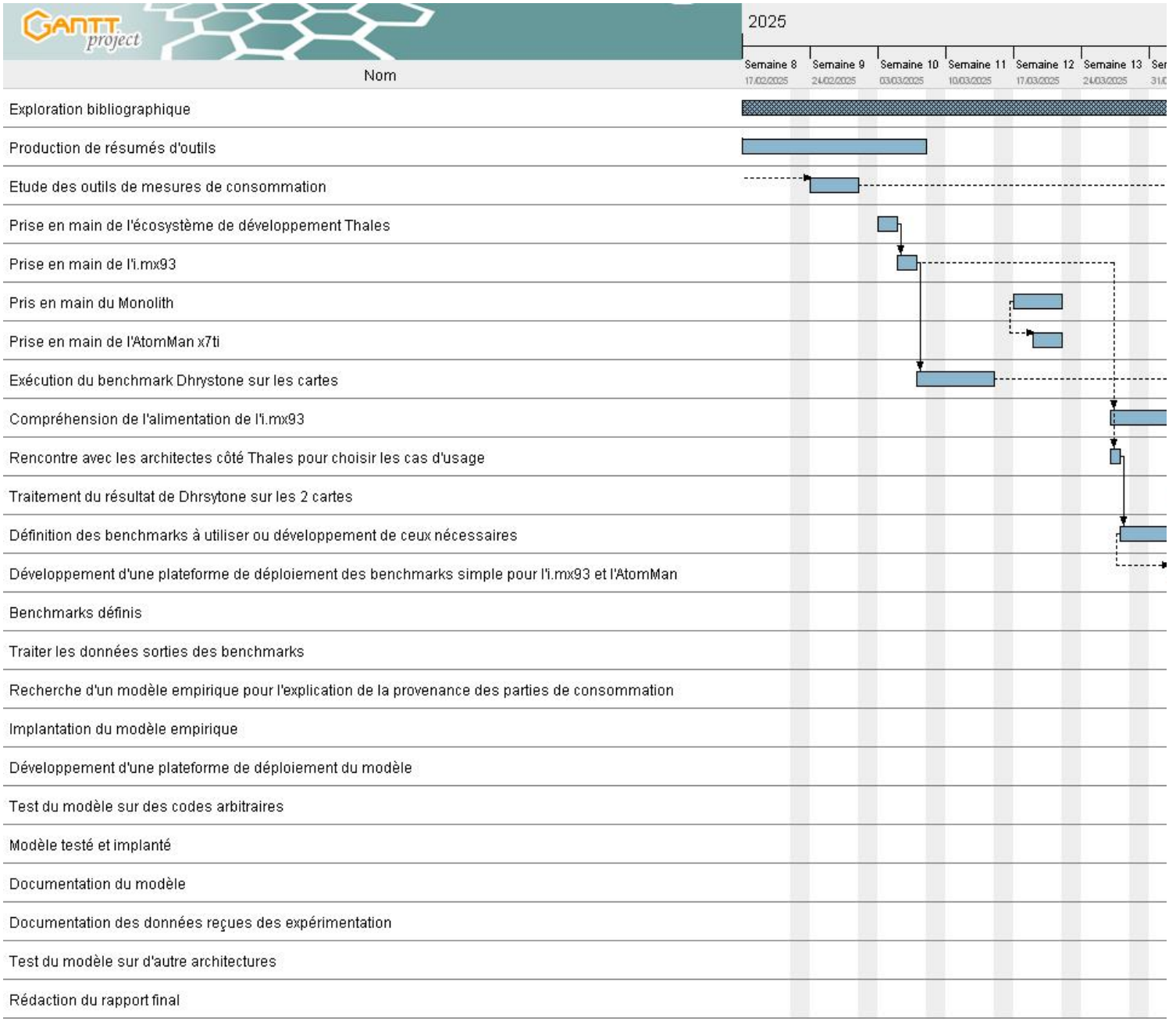


Figure 26: Diagramme de Gantt de la progression du stage

Part IX
L'écologie au sein du développement logiciel

Suite à ces développements et recherches que nous avons menés, nous avons entreprit de réaliser une analyse de l'impact environnemental de ce projet. Cette partie en fait un résumé.

L'impact du numérique sur notre environnement n'est pas négligeable. l'ADEME[34] statue que la part de l'impact carbone total du numérique en France est légèrement inférieure à celle du secteur des poids lourds (soit 4,4%). Selon l'IAE[35] (International Energy Agency), le numérique pourrait passer d'une demande mondiale de 470 TWh à l'heure actuelle jusqu'à une demande de 945 TWh horizon 2030, soit une part doublée en 5 ans.

Il est de fait intéressant de se poser la question de son propre impact sur ces chiffres monstrueux. Cette partie aborde un état de l'impact sur l'environnement, plus précisément l'impact carbone, de ce stage à plusieurs niveaux de lecture. Nous aborderons en d'abord les politiques en matière d'environnement des 2 structures d'accueil qui ont hébergées ce stage. Nous apporterons ensuite une réflexion sur l'impact de ce stage à court et long terme en abordant le problème de l'effet rebond. Enfin, nous nous prêterons à un exercice de modélisation de l'impact personnel induit par ce stage.

19 Politiques des structures d'accueil

Heureusement, les deux structures dans lesquelles j'ai été ont toutes deux une conscience de leur impact et propose une politique de mitigation de celui-ci. Un point intéressant à considérer, cependant, est leurs tailles : d'un côté Thales possède 77000 employés partout dans le monde, d'un autre le Lip6 est localisé à Paris et ne possède "que" 500 employés. Considérant cela, on pourrait s'attendre à des propositions plus fortes et des réussites plus importantes sur ces sujets de la part de Thales que du Lip6 : plus la structure est grosse, plus gros est son impact, plus elle doit s'évertuer à le réduire. C'est donc avec ce spectre de lecture que nous allons aborder les 2 sous-sections suivantes.

Thales - SIX - HTE

Thales propose une politique et des objectifs forts en termes d'écologie. Ces objectifs ont été analysés et répertoriés sur le site de la SBTi (Science Based Target initiatives), concrétisant un peu plus leurs validités scientifiques et leurs côté véridique. Parmi eux, nous pouvons lister les suivants :

- Réduction de 50,4 % des émissions des scopes 1 et 2 (en référence à l'année 2018 et en valeur absolue) – alignée avec une trajectoire 1,5 degrés
- Réduction de 15 % des émissions du scope 3 (en référence à l'année 2018 et en valeur absolue) – alignée avec une trajectoire 2 degrés.

Selon Wikipedia[36], les scopes sont définis de la façon suivante :

- Scope 1 : recouvre les émissions de l'activité elle-même.
- Scope 2 : correspond aux émissions indirectes liées à l'énergie
- Scope 3 : tous les autres processus induits, en amont ou en aval

Les scopes 1 et 2 ne représentent que 1,6% de l'empreinte carbone de Thales. Le reste provenant du scope 3. Actuellement, Thales a réduit de 56,8% ses émissions pour les scope 1 et 2, de 24,7% pour le scope 3, comparé aux valeurs d'impact de 2018. Thales a été capable d'arriver à ces résultats en entreprenant la réalisation de projets dont nous listons un échantillon ici:

[34] Thomas BRILLAND, "EVALUATION DE L'IMPACT ENVIRONNEMENTAL DU NUMERIQUE EN FRANCE".
[35] *Rapport de l'IEA sur la consommation énergétique et l'IA globale.*
[36] *Page wikipedia sur la définition des "scope".*

Le projet DECOR En 2024, Thales a lancé et pilote le projet de recherche collaboratif DECOR, dont l'objectif est de déployer à grande échelle le concept Green Flag pour l'optimisation de la navigation aérienne. Ce projet, réalisé en partenariat avec la Direction des services de la navigation aérienne (DNSA) et Air France, vise à réduire de 10% l'empreinte environnementale du transport aérien d'ici 2030, principalement via une réduction des émissions de CO2.

Le concept Green Flag cible des problématiques opérationnelles comme l'optimisation des temps de descente et la réduction des retards en vol, qui ont un impact direct sur les émissions du secteur. Grâce à des outils numériques facilitant la coordination entre contrôleurs aériens, pilotes, compagnies et gestionnaires de trafic, il s'agit d'identifier des fenêtres optimales pour effectuer des procédures de vol économes en ressources, selon la trajectoire, l'altitude et la vitesse des avions.

Après des tests en laboratoire, deux vols d'essai Air France entre Paris-Orly et Toulouse-Blagnac ont été réalisés en mars 2022 (projet OCTAVIE), montrant une diminution possible des émissions de CO_2 de l'ordre de 10% via une gestion améliorée des trajectoires et des flux.

CLOE et PETER CLOE est définie comme une "liste de contrôle pour orienter l'écoconception depuis la description de la valeur utilisateur jusqu'aux exigences d'ingénierie" et PETER comme un "outil d'évaluation des produits pour l'écoconception et le reporting avec une base de données unique de modèles de plateformes de mobilité". L'idée sous-jacente derrière ces deux outils est de fournir aux équipes de développement de projet des moyens pour décider efficacement et dégrossir l'impact final de leurs produits. Elles prennent la forme de calculatrice ou liste de cases à cocher en ligne qui embarquent des heuristiques et des données d'impact carbone comparables à celles de d'ecodiag[37].

Réseaux d'entreprise Plusieurs réseaux de personnes à différents niveaux de la hiérarchie Thales se réunissent et entretiennent des discussions autour des sujets d'écoconception. J'ai pu assister à des présentations/journées de retours au sein de l'un d'entre eux. Ces réseaux permettent plusieurs choses. La première est d'être un vecteur de communication entre la hiérarchie plus haute et les acteurs de terrain de ces problématiques. Le second est de pouvoir servir de retour d'expérience envers ces acteurs de terrain : montrer ce qui a pu fonctionner pour certains, ne pas fonctionner pour d'autres ou bien ce qui a été accompli. Enfin, ces réseaux permettent de mettre en relation des personnes de domaines diverses pour faire émerger des idées et concepts, l'idée sous-jacente étant que l'éco conception passe par toutes les étapes de la conception d'un produit et qu'il vaut mieux y réfléchir de façon horizontale à toute les disciplines plutôt que sillonnée à une seule. Bien que ces réseaux ne soient pas une forme concrète d'action et impacte directement la réduction de l'impact carbone, ils permettent de faire émerger les initiatives qui le feront.

D'autres actions pourraient encore être citées. Par exemple, Thales favorise le recyclage de projets client et les accompagne dans la gestion de fin de vie de leurs produits : ils ont racheté et démantelé des nacelles d'avion pour réutiliser les matières premières. Une partie de la stratégie du groupe repose aussi sur la revalorisation des produits en fin de vie.

Lip6 - ALSOC

Le Lip6 avance plus lentement et surtout sous la tutelle de l'université de la Sorbonne. En effet, une grande partie de la gestion du laboratoire est laissée à l'université et impute donc ses actions possibles. Par exemple, le bilan carbone[38] émis en 2021 par l'université indique que 35% des émissions de l'université provient des bâtiments. Deux des cinq recommandations fournies par l'organisme d'évaluation responsable de ce bilan portent sur l'amélioration de l'efficacité énergétique globale de l'université. Ce sont des axes sur lesquels le Lip6 ne peut pas vraiment agir et dont il est dépendant de l'université. On peut cependant noter les quelques initiatives suivantes :

[37] *Site web de l'outil ecodiag.*
[38] *Bilan carbone de la Srobonne, 2021.*

Colloquiums Des colloquiums sont organisés sur ces sujets d'éco conception : par exemple, Agnès Crepet, directrice technique de Fairphone, est venue présenter la vision de sa marque quant aux méthodes pour déjouer l'obsolescence programmée des smartphones (conférence à laquelle j'ai pu assister). Ces présentations permettent une vulgarisation des sujets traités et promeuvent des initiatives aux problèmes abordés (j'y ai appris l'existence d'OS alternatifs pour smartphone comme le postmarketos). Ces événements sont ouverts à tous (étudiants comme personnel du laboratoire et de l'université) et servent de moyens de diffusion de l'information.

Modification de leur règlement intérieur Suite à des discussions avec le groupe "impact-eco" du Lip6, celui-ci a changé son règlement intérieur pour ajouter un alinéa invitant son personnel à se questionner sur l'impact de sa recherche sur l'environnement ainsi que des façons de le réduire. L'intégrer de cette façon dans le règlement intérieur est une première étape vers une institutionnalisation de méthodes plus globales et contraignantes au sein de laboratoire pour réduire son impact.

PEPR Numérique éco responsable Un projet de recherche sur le numérique éco responsable est en train de se monter au sein du laboratoire. Je n'ai pas plus d'informations que cela, mais s'il arrive à bout, il pourra notamment financer des recherches sur ces sujets.

Critiques quant aux politiques

Pour être honnête, j'ai été surpris de voir que ce sujet de stage et Thèse ait été proposé par Thales. Une fois à l'intérieur de l'entreprise cependant, je peux dire qu'il y a une vraie dynamique autour de l'éco conception, avec des responsables qui ont un temps alloué à ces réflexions. Tout n'est pas parfait - certains se plaignent d'un manque de moyens au niveau des projets pour mettre en place les idées évoquées - mais le mouvement en est plus qu'à sa genèse. Une critique peut être formulée quant à l'étude des effets rebond de l'amélioration de certains systèmes. Le projet DECOR, par exemple, promet une amélioration de 10% de la consommation énergétique d'un aéroport en optimisant les trajectoires de ses avions, mais rien n'est dit quant à la quantité totale de demande en avion que cela pourrait engendrer. Les outils comme CLOE ou PETER restent très haut niveau dans leur utilisation et se base sur des heuristiques simples : hors, le problème n'est pas simple. C'est l'analyse que nous avons proposé en introduction à la problématique partie III. Bien qu'ils permettent une première réflexion et de premières améliorations, ces outils ne seront probablement pas suffisant dans l'étude plus approfondie de conceptions complexes. Enfin, derrière le terme "revalorisation" des produits conçus peut se cacher bien des sens. Ainsi, dans l'exemple que nous avons évoqué, rien n'est dit quant à la nature de la revalorisation de ces nacelles : un esprit mal tourné pourrait même dire que les racheter constituait la revalorisation.

Le Lip6 a encore une marge de progression importante sur ces sujets. La problématique est moins mûre que chez Thales, mais les choses avancent. Au niveau de l'université, un plan d'action a été mis en place pour répondre aux enjeux de développement durable. Une direction et des groupes de réflexions ont été mis en place. Cependant, il n'y a pas encore de plan détaillé, avec des objectifs chiffrés et des avancements clairement mesurés.

20 Impact global du projet

Dans cette section nous allons aborder l'impact global du produit de ce stage sur la société et l'environnement. Nous supposerons les choses suivantes : le modèle décrit partie VI est complet et fonctionnel sur tout type de programme pour tout type de cartes et les cartes et programmes évalués sont déjà existants.

Impact environnemental

Le cycle de vie que nous considérons ici pour le produit final est composé de 4 étapes : la fabrication, l'acheminement, l'utilisation et la fin de vie.

Petite échelle

Production En ce qui concerne la fabrication du logiciel final, quelques cartes auront été nécessaires à son élaboration. Nous avons cité l'i.mx93 et l'AtomMan x7ti, mais d'autres seront probablement nécessaires à l'élaboration d'une méthode plus générale qu'actuellement. Le Lip6 possède un second système comme le monolithe nommé "Dalek", composé uniquement de cartes graphiques - GPU. Ces cartes sont très largement utilisées dans certains cadres d'application, et sont responsables d'une grosse partie de la consommation : elles ne sont donc pas négligeables.

Transports En supposant que ces cartes ait été acheminées sur un container Panamax remplit provenant de Chine et ne transportant que des cartes de même dimensions, nous aurions une approximation du coût carbone de cette partie du trajet d'environ $35 \pm 5 * 20000/(15 \pm 5)^6 = 5,5 \pm 2,5 gCO^2$. En supposant que la carte une fois arrivée au port du Havre (car livré à Paris) soit livrée en semi-remorque classique - d'une capacité de stockage de 33 palettes pour un volume total de 90 m^3, avec à la louche 50 ± 20 i.mx93 par m^3- jusqu'à un entrepôt de stockage, elle coûterait $0,873 * 200/(90 * (50 \pm 20)) = 0,0388 \pm 0,026 gCO2$ pour cette partie du trajet. Au vu de cette dernière "mesure" nous n'ajouterons pas de coût lié au transport entre l'entrepôt et Thales ou le Lip6, car négligeable dans ce cas. En supposant que nous commandions 10 cartes de cette façon, du même volume et provenant de Chine, nous pourrions avoir un coût carbone de transport total de $10 * (5,5 + 0,038 \pm 2,526) = 55,38 \pm 25,26 gCO2$.

Utilisation L'utilisation du logiciel en soit se compose de 2 parties : une première d'entraînement et une seconde d'inférence. L'inférence a pour but d'être plus efficace que des méthodes de simulations ou analytique où le temps de calcul se compte en heure. Ici, en supposant que cette méthode d'inférence ne demande pas l'exécution du programme étudié, elle ne sera que de quelques minutes tout au plus, en fonction de la taille du programme. La partie entraînement cependant impliquera de plus grosse plages de temps d'utilisation des cartes, pouvant s'étaler sur un ou deux jours complets - nous avons fait des tests jusqu'à 400 exécutions différentes durant quelques heures, et certains papiers arrivent à avoir une bonne performance avec un jeu de données de 2600 éléments. En supposant que le temps d'entraînement soit de 48h sur une carte type AtomMan x7ti à régime maximal, on obtiendrait une dépense de $5,2MJ$ pour l'entraînement - noter que ces suppositions sont fortes, car le principe de l'entraînement est de tester différentes combinaisons de programmes pour différentes consommations finales. Selon le site de RTE France[39] et le site de l'ADEME, le coût carbone moyen du mix énergétique français sur l'année 2024 était d'environ 51,9 gC02/kWh. Cela reviendrait donc à un coût de $(5200/(3,6 * 1000)) * 51,9 = 74,9 gCO2$ par entraînement.

En supposant maintenant que la durée d'utilisation du logiciel est de 5 minutes sur une AtomMan x7ti en régime maximal, la consommation totale par d'utilisation par programme serait de $5 * 60 * 30 = 9kJ$. Cela revient à un coût carbone de $0,12 gCO2$.

Dans le cadre du développement du logiciel et de l'utilisation du logiciel dans un cercle restreint (équipe de développement et test) nous pourrions considérer que le nombre d'exécutions testées pourra avoisiner raisonnablement la dizaine de millier par cartes (vérification et création de jeux de données pour validation du modèle, etc). Partant de ce constat, le développement de ce logiciel aura finalement coûté $0,12 * 100000 + 55,38 + 74,9 * 10 = 12804 gCO2$. C'est équivalent à 4096km en TGV selon le comparateur public ImpactC02[40], soit l'équivalent d'un peu plus de 4 aller-retour Paris-Grenoble.

Grande échelle

Comme tout programme permettant de créer des échelles de comparaison, il permet aussi de voir ce qui est gagné par telle ou telle méthode. Directement, notre outil n'apporte pas d'amélioration aux systèmes existants, mais aura pour but à l'avenir de pointer les sources de dépenses et d'améliorations possibles. Ainsi, des programmes pourront devenir plus efficaces au vu d'un budget énergétique alloué et il sera possible d'imaginer en exécuter plus pour le même budget.

[39] *Site RTE de l'indicateur éCO2mix.*
[40] *Comparator public d'impact carbone ImpactCo2.*

Dans le cas de l'informatique embarquée cela pourrait se traduire par un surdimensionnement volontaire des cartes de calcul dans les systèmes finaux afin d'exécuter des fonctionnalités jusque-là inatteignables par manque d'énergie et d'optimisation. Nous aurions donc des systèmes plus obèses en puissance de calcul, ce qui n'est pas forcément souhaité dans le cadre de l'informatique embarqué.

Un autre point à noter est que la méthode, bien que générique, implique un entraînement par modèle de carte. De plus, supposer que le modèle fonctionne et fonctionnera sur toutes les technologies existantes sans besoin de réadaptation est quelque peu utopique : une nouvelle puce spécialisée, une nouvelle hiérarchie mémoire ou autre amélioration pourrait remettre en cause les fondements du modèle et le rendre caduque. Donc supposer que chaque carte n'aura besoin que d'un seul entraînement avant d'arriver à un modèle utilisable est peu raisonnable.

Enfin, c'est un problème global à l'informatique, si logiciel se démocratise alors son nombre d'exécutions pourrait exploser et amener à un impact beaucoup plus grand que celui présenté section précédente. Cependant, le grand public n'est pas celui visé par ce logiciel : il s'adresse à des développeurs/chercheurs comprenant les enjeux et les spécificités du domaine embarqué. Sa capacité d'explosion n'est donc pas comparable à celle d'un jeu en ligne ou d'un réseau social disponible sur smartphone par exemple, les adeptes de ces logiciels sont bien plus nombreux.

Impact sociétal

Notre logiciel n'adresse pas l'utilisateur final dans sa conception actuelle. Sa forme finale n'a pas encore été conçue ni imaginée. Il n'aborde donc pas les aspects RGPD qui y sont liés. Cependant, son utilisation requiert le super-utilisateur, ce qui amène à l'ouverture de la surface d'attaques potentielles, et donc par effet de bords, menacer l'utilisateur final.

Au sein de l'entreprise ce logiciel aura pour but d'aiguiller des personnes dont ce n'est pas le travail direct sur l'évaluation de la consommation de leurs applications. Ils leur reviendra de les améliorer et les rendre plus efficientes, mais ils n'auront pas à répondre du pourquoi de la méthode et pourront s'y référer comme échelle. De ce fait, nous espérons qu'au sein de l'entreprise, il apporte un soutien au développeur.

Nous espérons aussi que cette méthode touchera le monde de la recherche, permettant à d'autres de se poser des questions directement en lien avec la consommation et l'empreinte de nos applications et développer des méthodes génériques de conception de programme moins énergivores.

21 Impact personnel durant le PFE

Dans cette partie nous brossons un tableau de l'impact environnemental personnel lié à ce stage. Les données unitaires de consommation (gCO2) sont tirées du site de l'Ademe[41]. L'amortissement des produits est tiré des méthodes internes de d'évaluation de l'amortissement de Thales. Les dates d'année d'achat correspondent aux dates de manufactures marquées sur les produits utilisés. La puissance instantanée est soit donnée en fiche technique des produits, pour les écrans notamment, soit estimée par ma propre utilisation des produits ce qui est le cas pour les ordinateurs. Cette dernière méthode se base sur la capacité de la batterie et le temps d'utilisation moyen des ordinateurs lorsque sur batterie, en usage normal. Les temps d'utilisation tiennent compte de ma localisation (Lip6 ou Thales) et des jours fériés.

La méthode de calcul pour le coût carbone à l'achat se base sur la proportion d'utilisation du produit qui m'est attribuable. La méthode pour calculer cette proportion se base sur la durée de vie totale du produit, qui est estimée comme suit : si le produit est encore en train d'être amortit alors c'est la période moyenne d'amortissement qui est considérée, sinon c'est l'âge actuel du produit. Par exemple, la durée de vie du Samsung 2043BW considérée ici est 13 années (ainsi que quelques mois et jours) et non 5. C'est une façon pour mitiger la méthode du "pire cas" qui aurait toujours considéré

[41] *Page de Base Empreinte par l'Ademe.*

l'âge actuel du produit.

Le coût des transports a été divisé par le nombre de passagers par trajet. C'est pour cela que le coût des trajets en voiture n'est pas aussi élevé que l'exemple fournit en consignes. La consommation des claviers et souris sont pris en compte par la consommation des portable et ordinateurs.

Plusieurs choses sont intéressantes à noter quant à ces données. La première est que l'impact carbone lié à la consommation des produits qui m'ont été attribués représente seulement 12% de l'impact carbone total de ces produits. Mais il est trop réducteur de simplement dire qu'il faut conserver ses objets le plus longtemps possible pour réduire son impact total : cela dépend du produit. En l'occurrence, pour l'écran Samsung, produit en 2008, la proportion est inversée avec plus de 60% de l'impact lié à l'utilisation. Il double cette part comparée à l'écran ThinkVision qui est 4 ans plus jeune et au total, cet écran présente moins d'impact que le Samsung.

Nous voyons ici que l'impact des transports est faible comparé au reste. C'est dû au fait que je n'ai effectué qu'un seul trajet professionnel lors de ce stage, et qu'il était à moitié composé d'une partie en train. Un autre point intéressant est que la proportion des trajets quotidiens est très faible comparée à celle de ce trajet. Les transports en commun (surtout les métros) ont un impact négligeable comparé au reste.

Type	Qt	Marque	Modèle	Puissance instantanée moyenne (W)	Année d'achat	Amortissement moyen (Années)	Utilisation Quotidienne (H)	Nombre jours d'utilisation	Utilisation (gCO2)	Achat (gCO2)
écran	1	ThinkVision	LT2252pwD	20	2012-02-22	5	7	64	465.02	850.90
écran	1	Samsung	2043BW	40	2008-02-03	5	7	64	930.05	654.10
écran	2	Dell	P2414HD	19.35	2016-05-01	5	7	58	815.46	2238.54
portable	1	Lenovo	V510-151IKB 80WQ	12	2017-06-04	5	7	64	279.01	3339.13
ordinateur	1	Dell	Optiplex 9020	15	2025-08-11	5	7	58	121.80	5370.96
portable	1	HP	EliteBook 845 G10	9	2023-03-01	5	7	122	153.72	10428.49
souris	3	Dell	NaN	NaN	NaN	5	7	122	0.00	NaN
claviers	1	HP	KUS1206	NaN	NaN	5	7	64	0.00	NaN
claviers	1	Logitech	Y-UT76	NaN	NaN	5	7	58	0.00	NaN

Table 9: Coût carbone du matériel utilisé pendant le stage

Catégorie	Empreinte carbonne (gCO2)
Transports	6149
Utilisation du matériel	3204
Achat du matériel	22882

Table 10: Répartition de l'empreinte carbone en fonction du poste de dépense

Moyen de transport	Trajet	Modèle	Nombre	Distance (km)	Emission (gCO2)
Métro	Lip6-Domicile	M14+M7	116.00	10.00	44.00
Métro	Thales-Domicile	RER C + M14	128.00	20.00	109.52
Train	Paris-Angers	NaN	2.00	NaN	1700.00
Voiture	Angers-Cholet	Dacia Sandero	2.00	62.00	4340.00

Table 11: Coût carbone des trajets professionnels liés à ce stage

Emission de la production d'un ordinateur portable (kgCO2/u)	Emission de la production d'une tour (kgCO2/u)	Emission de la production d'un écran (kgCO2/unité)	Facteur d'émission de la Dacia Sandero (gCO2/km)	Emission du train sur l'allé à Cholet (gCO2/km)	Impact Carbone Moyen du mix énergétique Français (gCO2/kWh)
156	169	65.4	105	1700	51.9

Table 12: Constantes utilisées dans les formules de calcul des tableaux 11 et 9

Part X
Conclusion et perspectives

La partie VII montre à quel point le domaine étudié peut-être complexe. Des paramètres qui n'avaient pas été anticipés au départ se sont finalement révélés vitaux au problème. Comme dit, l'hétérogénéité des systèmes étudiés, que ce soit au niveau de l'OS ou de la plateforme, impliquent une étude approfondie dans leurs fonctionnements comme dans leurs conceptions. Nous avons cependant réussi à réaliser une étude préliminaire et en tirer des conclusions intéressantes :

- Les outils de mesure de consommation peuvent être un poste d'erreur non négligeable. Voir section V.
- Les méthodes analytiques pour l'estimation de consommation étaient peut-être viables il y a quelques années, mais le sont de moins en moins. Les méthodes empiriques prennent le pas et proposent de meilleurs résultats et plus transposables.
- Les architectures hétérogènes impliquent une attention particulière à la localité de l'exécution des tâches pour la consommation finale. Des compromis sont à trouver entre performance temporelle et performance énergétique.
- En plus des différents cœurs, chaque cœur peut (ou non) changer de fréquence pour une même charge. Ces changements impliquent une croissance non linéaire de la consommation, et qui dépend des cœurs utilisés.
- Lorsque les cœurs n'ont pas de charge à exécuter, ils "s'endorment" et baissent en consommation. Hors, les cœurs peuvent entrer dans différents niveaux de mise en veille qui influent la consommation de manière différente. De ce fait, poser un étalon entre plusieurs architectures et cartes différentes devient compliqué : ces états sont définis de façon différente et impactent différemment la consommation.
- Essayer de caractériser la consommation d'un programme en profilant l'énergie des instructions qui le composent peut-être une piste de réflexion pour la consommation sur x86_64. Cependant, elle dépendra des cœurs utilisés et il sera plus judicieux de considérer des groupes d'instructions plutôt que des instructions seules.

D'autres futurs peuvent être imaginées. Notre solution se base sur une contrainte forte : le fait d'impacter le moins possible la mémoire. Hors aujourd'hui, il n'existe probablement pas de systèmes industrialisés ne possédant pas de mémoire. Lever cette contrainte nous permettrait d'atteindre un modèle plus polyvalent et surtout comparable à ceux présentés section 6. Bien que nous n'en ayons pas parlé ici, l'évaluation de la consommation de la mémoire seule possède aussi son lot de recherches. Plusieurs méthodes existent, et sont en tous points similaires à celle évoquées dans la partie 6. Cependant, les modèles analytiques arrivent à avoir des estimations plus précises que ceux évoqués pour les unités de calcul principales. Une piste à explorer pourrait donc être de mêler un modèle analytique pour la RAM et un modèle empirique pour les calculs. Pour évaluer ces consommations, nous avons vu en partie 7 que RAPL n'était pas le meilleur outil pour y arriver. De plus, la mesure depuis la carte évaluée de la consommation d'énergie apporte une erreur certaine. Il sera donc possible d'utiliser des sondes type Lauterbach pour analyser l'état du système depuis l'extérieur et récupérer une donnée plus propre et moins soumis aux aléas de la mesure.

Ces modèles de régression se basent sur le domaine des probabilités et des statistiques. Fort est à parier que le raffinement statistique des données en amont de leur utilisation brute est une condition nécessaire à l'obtention de bons résultats. C'est d'ailleurs ce que font bon nombres de chercheurs dans des domaines annexes aux nôtre, mais étudiant des systèmes plus complexes encore : l'économie et la sociologie. Nous pourrions donc à l'avenir chercher à caractériser l'efficacité de notre modèle en fonction des raffinements effectués en amont : maximiser un coefficient de corrélation, garder un VIF (Variance Inflation Factor) suffisamment bas, utiliser des lois de densités connues pour des facteurs

connus (loi de Poisson pour les PMCs), etc. Il sera aussi possible de tester d'autres types d'approches comme des "surrogate model" sur lesquels il faudra monter en compétences, mais qui ont déjà fait leurs preuves dans d'autres cas d'application.

Tout modèle de ce type se base sur les données d'entrée qu'il ingère. Une autre piste pourra être de catégoriser la charge d'entraînement en fonction de la cible : trouver une méthode générique à celle que nous avons utilisé pour nos algorithmes de fractales, en section 11. Trouver une telle méthode pourrait nous permettre aussi de réfléchir dans l'autre sens. Il serait alors possible, via une mesure de distance entre un programme et un sous-ensemble de programmes dont on aurait caractérisé la consommation, de prédire une consommation. Réfléchir en termes de sous-ensembles de programmes pourrait aussi nous permettre de cibler les caractéristiques principales influant sur la consommation pour ces sous-ensembles, et donc indiquer au développeur final où chercher l'optimisation.

Remerciements Toutes ces expérimentations, recherches et développements n'auraient pas été possibles sans l'appui d'un bon nombre de personnes. Je pense en premier lieu à ces personnes qui m'ont fait confiance et m'ont suivi pendant toute la période du stage, mes encadrants : Xavier, Lounes, Alix Munier et Adrien Cassagne. J'ai pu bénéficier de leurs conseils et support au quotidien et je leur en remercie. Je remercie plus globalement l'équipe OSS et ALSOC respectivement au sein de Thales et du Lip6 qui m'ont accueilli, donné des idées et des pistes et plus globalement épaulé sur mon problème via leur connaissance sur les différents sujets qu'elles traitent. Je remercie également le Lip6 et Thales pour avoir proposé et m'avoir pris sur ce sujet.

Ce stage marque la fin de mes études supérieures de second cycle et il est, je pense, important de rappeler à quel point le cercle proche joue un rôle vital dans cet accomplissement. Je remercie mes parents, qui ont supporté mes choix et m'ont toujours assuré les meilleures conditions. Je remercie plus largement ma famille et mes proches pour l'écoute dont j'ai bénéficié : elle fut précieuse et bienvenue dans bien des cas !

Enfin, je souhaitais aussi remercier ce corps de travail, aujourd'hui sous représenté à mon goût : les professeurs. Polyech Grenoble, l'UGA (Université Grenoble Alpes) et l'Ensimag m'ont apporté beaucoup en compétences sociales et techniques et ce grâce à leurs corps de professeurs impliqués et appliqués dans leur mission. À ceux qui liront ce rapport et qui en font partie : merci !

Part XI
Annexes

22 CodeCarbon

Version 2.8.3[42]

Avant de s'orienter vers la consommation énergétique, une partie du travail de recherche s'était concentré à faire un état de l'art des méthode d'analyse d'empreinte carbone des programmes. CodeCarbon est l'un des outils les plus connus de ce domaine, nous proposons donc d'en faire une vue d'ensemble ici.

22.0.1 Description

CodeCarbon est un outil sous licence MIT[43] qui propose de s'intégrer à un code d'application donné par l'interface d'un plugin qui va calculer l'émission de CO2eq (CO2 équivalent) de celle-ci. Une fois l'application compilée, il utilise le pays dans lequel se trouve la machine l'exécutant ainsi que la consommation énergétique de l'application étudiée afin d'en tirer, via l'impact carbone du mix énergétique du pays, cet équivalent CO2. L'impact carbone du mix, nommée *efficacité carbone*, que nous noterons C est soit directement récupéré sur la *Page sur l'intensité carbone du site OurWorldInData*; soit calculé à partir du mix énergétique et de l'impact carbone de chaque source d'énergie.

Pour ce dernier cas, les proportions des mix énergétiques sont tirées d'un fichier dans leur dépôt github[44](dont la source n'est pas renseignée); et l'impact carbone des sources renseignées dans ce fichier est tiré des publications et sites suivants : Comparison of Lifecycle Greenhouse Gas Emissions of Various Electricity Generation Sources et Page Github du package python "energyusage". Pour résumer, C est donc calculé de la façon suivante pour ce dernier cas:

$$C = \sum_i p_i * m_i$$

où p_i est la proportion de l'énergie i dans le mix du pays considéré ($\sum p_i = 1$) et m_i l'impact carbone cette énergie, mesuré en kgCO2/Mwh.

Dans le cas où CodeCarbon n'aurait pas accès au mix énergétique du pays considéré (aux p_i), il prendra la valeur moyenne mondiale par pays, c'est-à-dire 475 gCO2.eq/KWh (voir Rapport de IEA sur l'état de l'énergie et du CO2 global de 2019).

CodeCarbon fournit une API et une interface graphique faciles d'utilisation, ce qui peut expliquer sa popularité sur Github (1.3k étoiles).

22.0.2 Méthodologie

Son analyse de consommation porte sur le CPU, le GPU et la RAM de la machine étudiée. L'analyse de consommation du CPU est réalisée via les outils suivants en fonction des plateformes mentionnées :

- Intel pour Windows ou Mac : utilisation du "Intel Power Gadget".
- Intel pour Linux : Utilisation de l'"Intel-RAPL"
- Apple Silicons sur Mac (M1, M2) : Utilisation de "powermetrics" (droits root nécessaires).

L'analyse de consommation GPU est quant à elle réalisée avec la bibliothèque "Pynvml". La consommation de la RAM est estimée empiriquement à 3 Watts par 8 Giga-octets.

La méthode d'étude de consommation est la suivante : toutes les 15 secondes des moyennes sur la proportion d'utilisation du CPU, de la GPU et de la RAM sont faites et un point de donnée est

[42] *Dépôt de codecarbon.*
[43] *Licence de codecarbon.*
[44] *Mix énergétiques utilisés par CodeCarbon.*

ajouté en multipliant ces proportions avec les estimations de consommation d'énergie tirées des outils mentionnés, puis des estimations d'émissions précédemment calculées/requêtées. Le résultat est une série temporelle qui représente l'émission de la machine au cours du temps. La formule suivante décrit ce que nous venons d'aborder :

$$I = C * E$$

où I est l'impact carbone en kgCO2, C l'efficacité carbone et E l'énergie totale dépensée par le système étudié.

Quelques nuances sont à noter sur ces prises de mesures. Pour la consommation du CPU, si le système ne parvient pas à estimer sa consommation en temps réel, il détectera le matériel utilisé et le comparera à une base de données de 2000 CPU Intel et AMD et leurs TDP (Thermal Design Power) pour récupérer la consommation électrique moyenne. Une fois obtenue, elle sera multipliée par l'utilisation du CPU liée à l'application. Si le CPU n'est pas répertorié, une constante globale est appliquée, et CodeCarbon suppose que 50 % du TDP représente la consommation électrique moyenne. N'ayant pas trouvé de ressources fiables sur la relation entre TDP et consommation moyenne, ils ont empiriquement déterminé que 50 % est une bonne approximation.

Les auteurs n'indiquent pas d'autres méthodes de calcul particulières dans le cas de la RAM ni de la GPU.

22.0.3 Conclusion

L'avantage principal de cet outil est sa facilité d'utilisation, ainsi que le lien direct qu'il propose entre la consommation électrique et l'impact carbone de celle-ci. On peut cependant noter que l'outil ne se base pas sur des chiffres en temps réel pour le mix énergétique. L'intervalle de mise à jour des données de consommation étant de 15 secondes, le plugin convient à des applications s'exécutant sur de longues périodes de temps.

Le plugin est aussi restreint par les outils sur lesquels il se repose. Il ne prend pas en compte la parallélisation des applications (multi-threading et décomposition en processus par exemple), ni une répartition précise de la consommation énergétique entre application car leur utilisation des outils comme RAPL ou NVML indiquent seulement la consommation de la puce entière - d'autres logiciels arrivent, avec les outils à faire plus *Page de documentation de Scaphandre sur RAPL*. Le modèle est simple, c'est ce qui le rend accessible.

Enfin, le logiciel se dit avoir un faible impact sur les performances du logiciel, mais ne propose pas de contre-mesures particulière. Ce dernier point est à tempérer sachant que la période d'échantillonnage par défaut est de 15 secondes.

23 Architectures

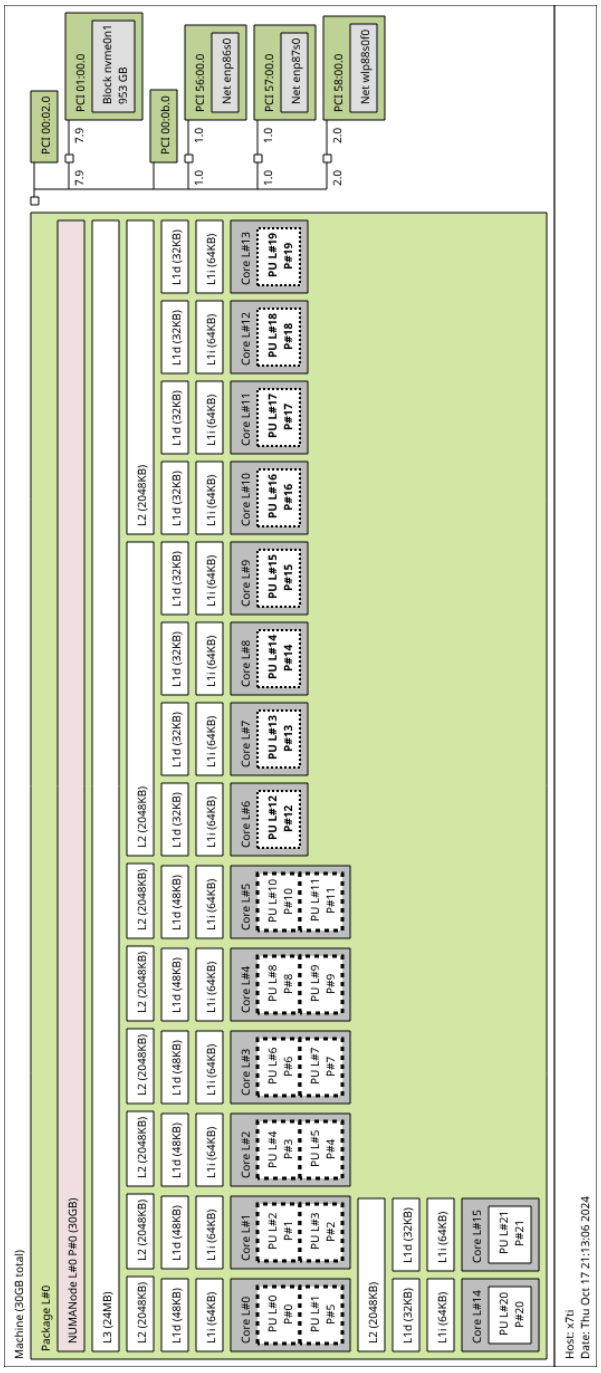


Figure 27: Architecture de l'AtomMan x7ti au sein du Monolithe

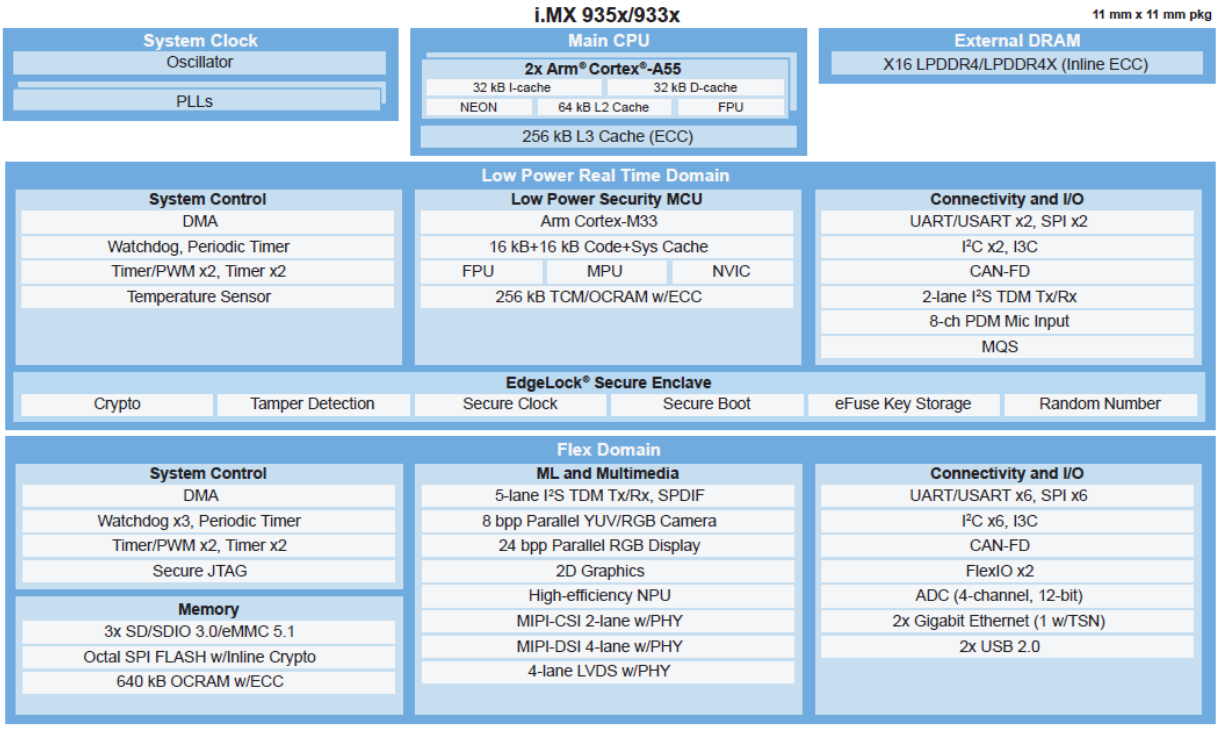


Figure 28: Composants de l'i.MX93

References

Association, Semiconductor Industries. *Model for Assessment of CMOS Technologies and Roadmaps (MASTAR)*. 2007. URL: http://www.itrs.net/models.html.

Bazzaz, Mostafa, Mohammad Salehi, and Alireza Ejlali. "An Accurate Instruction-Level Energy Estimation Model and Tool for Embedded Systems". In: *IEEE Transactions on Instrumentation and Measurement* 62.7 (July 2013), pp. 1927–1934. ISSN: 0018-9456, 1557-9662. DOI: 10.1109/TIM.2013.2248288. URL: http://ieeexplore.ieee.org/document/6478810/ (visited on 05/07/2025).

Bilan carbone de la Srobonne, 2021. URL: https://www.sorbonne-universite.fr/sites/default/files/media/2024-06/DD-RS-bilan-carbone2021.pdf (visited on 08/05/2025).

Butko, Anastasiia et al. "Full-System Simulation of Big.LITTLE Multicore Architecture for Performance and Energy Exploration". In: *2016 IEEE 10th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSOC)*. 2016 IEEE 10th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC). Lyon, France: IEEE, Sept. 2016, pp. 201–208. ISBN: 978-1-5090-3531-1. DOI: 10.1109/MCSoC.2016.20. URL: http://ieeexplore.ieee.org/document/7774439/ (visited on 03/19/2025).

Che, Shuai et al. "Rodinia: A Benchmark Suite for Heterogeneous Computing". In: *2009 IEEE International Symposium on Workload Characterization (IISWC)*. 2009 IEEE International Symposium on Workload Characterization (IISWC). Austin, TX, USA: IEEE, Oct. 2009, pp. 44–54. ISBN: 978-1-4244-5156-2. DOI: 10.1109/IISWC.2009.5306797. URL: http://ieeexplore.ieee.org/document/5306797/ (visited on 06/19/2025).

Comparator public d'impact carbone ImpactCo2. URL: https://impactco2.fr/outils/comparateur#simulateur (visited on 08/05/2025).

David, Howard et al. "RAPL: Memory Power Estimation and Capping". In: *Proceedings of the 16th ACM/IEEE International Symposium on Low Power Electronics and Design*. Austin Texas USA: ACM, Aug. 18, 2010, pp. 189–194. ISBN: 978-1-4503-0146-6. DOI: 10.1145/1840845.1840883. URL: https://dl.acm.org/doi/10.1145/1840845.1840883 (visited on 04/24/2025).

Denali. "Using Configurable Memory Controller Design IP with Encounter RTL Complier". In: *Cadence CDNLive!* 31 (2007).

Desrochers, Spencer, Chad Paradis, and Vincent M. Weaver. "A Validation of DRAM RAPL Power Measurements". In: *Proceedings of the Second International Symposium on Memory Systems*. Alexandria VA USA: ACM, Oct. 3, 2016. ISBN: 978-1-4503-4305-3. DOI: 10.1145/2989081.2989088. URL: https://dl.acm.org/doi/10.1145/2989081.2989088 (visited on 02/21/2025).

Discussion sur le forum de NXP parlant de DVFS. URL: https://community.nxp.com/t5/i-MX-Processors/IMX93-It-have-no-cpufreq-directory-under-sys-fs/m-p/1706196 (visited on 07/22/2025).

Documentation développeur d'Intel. URL: https://cdrdv2.intel.com/v1/dl/getContent/671447 (visited on 07/16/2025).

Dépôt de Carbontracker. URL: https://github.com/lfwa/carbontracker (visited on 02/25/2025).

Dépôt de codecarbon. URL: https://github.com/mlco2/codecarbon?tab=readme-ov-file#how-to-cite- (visited on 02/25/2025).

Dépôt de EnergAt. URL: https://github.com/HongyuHe/energat (visited on 02/25/2025).

Dépôt de McPat. URL: https://github.com/HewlettPackard/mcpat (visited on 04/17/2025).

Dépôt de Scaphandre. URL: https://github.com/hubblo-org/scaphandre?tab=readme-ov-file (visited on 07/16/2025).

Guthaus, M.R. et al. "MiBench: A Free, Commercially Representative Embedded Benchmark Suite". In: *Proceedings of the Fourth Annual IEEE International Workshop on Workload Characterization. WWC-4 (Cat. No.01EX538)*. Fourth Annual IEEE International Workshop on Workload Characterization. Austin, TX, USA: IEEE, 2001, pp. 3–14. ISBN: 978-0-7803-7315-0. DOI: 10.1109/WWC.2001.990739. URL: http://ieeexplore.ieee.org/document/990739/ (visited on 06/19/2025).

Hoste, Kenneth and Lieven Eeckhout. "Microarchitecture-Independent Workload Characterization". In: *IEEE Micro* 27.3 (May 2007), pp. 63–72. ISSN: 0272-1732. DOI: 10.1109/MM.2007.56. URL: http://ieeexplore.ieee.org/document/4292057/ (visited on 06/20/2025).

Hähnel, Marcus et al. "Measuring Energy Consumption for Short Code Paths Using RAPL". In: *ACM SIGMETRICS Performance Evaluation Review* 40.3 (Dec. 4, 2012), pp. 13–17. ISSN: 0163-5999. DOI: 10.1145/2425248.2425252. URL: https://dl.acm.org/doi/10.1145/2425248.2425252 (visited on 03/10/2025).

Keating, Michael, ed. *Low Power Methodology Manual: For System-on-Chip Design*. 2. print., corr. Series on Integrated Circuits and Systems. New York, NY: Springer, 2008. 300 pp. ISBN: 978-0-387-71818-7 978-0-387-71819-4.

Khan, Kashif Nizam et al. "RAPL in Action: Experiences in Using RAPL for Power Measurements". In: *ACM Transactions on Modeling and Performance Evaluation of Computing Systems* 3.2 (June 30, 2018), pp. 1–26. ISSN: 2376-3639, 2376-3647. DOI: 10.1145/3177754. URL: https://dl.acm.org/doi/10.1145/3177754 (visited on 04/24/2025).

Li, Sheng et al. "McPAT: an integrated power, area, and timing modeling framework for multicore and manycore architectures". In: *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*. Micro-42: The 42nd Annual IEEE/ACM International Symposium on Microarchitecture. New York New York: ACM, Dec. 12, 2009, pp. 469–480. ISBN: 978-1-60558-798-1. DOI: 10.1145/1669112.1669172. URL: https://dl.acm.org/doi/10.1145/1669112.1669172 (visited on 02/18/2025).

Licence de codecarbon. URL: https://github.com/mlco2/codecarbon?tab=MIT-1-ov-file (visited on 02/25/2025).

Malmodin, Jens and Dag Lundén. "The Energy and Carbon Footprint of the Global ICT and E&M Sectors 2010–2015". In: *Sustainability* (Aug. 2018). DOI: 10.3390/su10093027.

Marantos, Charalampos et al. "A Flexible Tool for Estimating Applications Performance and Energy Consumption Through Static Analysis". In: *SN Computer Science* 2.1 (Feb. 2021), p. 21. ISSN: 2662-995X, 2661-8907. DOI: 10.1007/s42979-020-00405-7. URL: http://link.springer.com/10.1007/s42979-020-00405-7 (visited on 05/07/2025).

Mazzola, Sergio et al. *Data-Driven Power Modeling and Monitoring via Hardware Performance Counters Tracking*. Jan. 3, 2024. DOI: 10.48550/arXiv.2401.01826. arXiv: 2401.01826 [cs]. URL: http://arxiv.org/abs/2401.01826 (visited on 06/15/2025). Pre-published.

McCalpin, John D. *STREAM: Sustainable Memory Bandwidth in High Performance Computers*. Tech. rep. A continually updated technical report. http://www.cs.virginia.edu/stream/. Charlottesville, Virginia: University of Virginia, 1991-2007. URL: http://www.cs.virginia.edu/stream/.

Mix énergétiques utilisés par CodeCarbon. URL: https://github.com/mlco2/codecarbon/blob/master/codecarbon/data/private_infra/global_energy_mix.json (visited on 02/25/2025).

Notice d'utilisation de la PMU d'ARMv8.1. URL: https://documentation-service.arm.com/static/63f365789567172d4e2aadf5 (visited on 07/18/2025).

Noudohouenou, Jose and William Jalby. "Using Static Analysis Data for Performance Modeling and Prediction". In: *2014 International Conference on High Performance Computing & Simulation (HPCS)*. 2014 International Conference on High Performance Computing & Simulation (HPCS). Bologna, Italy: IEEE, July 2014, pp. 933–942. ISBN: 978-1-4799-5313-4 978-1-4799-5312-7 978-1-4799-5311-0. DOI: 10.1109/HPCSim.2014.6903789. URL: http://ieeexplore.ieee.org/document/6903789/ (visited on 05/06/2025).

Page de Base Empreinte par l'Ademe. URL: https://base-empreinte.ademe.fr/ (visited on 08/12/2025).

Page de documentation de Scaphandre sur RAPL. URL: https://hubblo-org.github.io/scaphandre-documentation/explanations/rapl-domains.html (visited on 02/25/2025).

Page de tutoriel de bcc. URL: https://github.com/iovisor/bcc/blob/master/docs/tutorial.md (visited on 07/17/2025).

Page de tutoriel de bcc. URL: https://fr.wikipedia.org/wiki/Ensemble_de_Mandelbrot (visited on 07/17/2025).

Page eBPF du site de Brendan Gregg. URL: https://www.brendangregg.com/ebpf.html (visited on 07/17/2025).

Page sur l'intensité carbone du site OurWorldInData. URL: https://ourworldindata.org/grapher/carbon-intensity-electricity#explore-the-data (visited on 02/25/2025).

Page web de description de l'AtomMan disponible au Lip6. URL: https://monolithe.proj.lip6.fr/home_description/#atomman-x7-ti (visited on 03/28/2025).

Page web de la carte d'évaluation de l'i.mx93. URL: https://www.nxp.com/design/design-center/development-boards-and-designs/i.MX93EVK?tid=vani.MX93EVK (visited on 03/28/2025).

Page web de la RGESN. URL: https://www.arcep.fr/uploads/tx_gspublication/referentiel_general_ecoconception_des_services_numeriques_version_2024.pdf (visited on 04/17/2025).

Page web de l'équipe ALSOC. URL: https://www.lip6.fr/recherche/team.php?acronyme=ALSOC (visited on 03/28/2025).

Page Wikipedia de l'Analyse du cylce de vie. URL: https://fr.wikipedia.org/wiki/Analyse_du_cycle_de_vie (visited on 04/17/2025).

Page wikipedia sur la définition des "scope". URL: https://fr.wikipedia.org/wiki/Scope_(bilan_carbone) (visited on 08/05/2025).

Palacharla, Subbarao, Norman P Jouppi, and J E Smith. "Complexity-Effective Superscalar Processors". In: ().

Raffin, Guillaume and Denis Trystram. *Dissecting the Software-Based Measurement of CPU Energy Consumption: A Comparative Analysis*. July 19, 2024. DOI: 10.48550/arXiv.2401.15985. arXiv: 2401.15985 [cs]. URL: http://arxiv.org/abs/2401.15985 (visited on 06/15/2025). Prepublished.

Rapport de l'IEA sur la consommation énergétique et l'IA globale. URL: https://iea.blob.core.windows.net/assets/601eaec9-ba91-4623-819b-4ded331ec9e8/EnergyandAI.pdf (visited on 08/05/2025).

Répertoire de PMCs d'Intel. URL: https://perfmon-events.intel.com/ (visited on 07/18/2025).

Site de l'ACPI sur les états de processeur. URL: https://uefi.org/htmlspecs/ACPI_Spec_6_4_html/08_Processor_Configuration_and_Control/processor-power-states.html (visited on 07/23/2025).

Site de noireaudes page sur l'analyse de consommation. URL: https://recherche.noiraudes.net/ecoinfo/numres/ressources/TP/04-mesure-conso.html (visited on 07/16/2025).

Site RTE de l'indicateur éCO2mix. URL: https://www.rte-france.com/eco2mix/synthese-des-donnees?type=co2 (visited on 08/05/2025).

Site web de l'outil ecodiag. URL: https://ecoinfo.cnrs.fr/ecodiag-calcul/ (visited on 08/05/2025).

Site web du monolith hébergé au Lip6. URL: https://monolithe.proj.lip6.fr/ (visited on 03/28/2025).

Thomas BRILLAND Erwann FANGEAT, Julia MEYER Mathieu WELLHOFF. "EVALUATION DE L'IMPACT ENVIRONNEMENTAL DU NUMERIQUE EN FRANCE". In: (). URL: https://ecoresponsable.numerique.gouv.fr/docs/2024/etude-ademe-impacts-environnementaux-numerique.pdf.

Walker, Matthew J. et al. "Accurate and Stable Run-Time Power Modeling for Mobile and Embedded CPUs". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 36.1 (Jan. 2017), pp. 106–119. ISSN: 0278-0070, 1937-4151. DOI: 10.1109/TCAD.2016.2562920. URL: http://ieeexplore.ieee.org/document/7464834/ (visited on 02/24/2025).

Weaver, Vincent M. et al. "Measuring Energy and Power with PAPI". In: (2012), pp. 262–268. DOI: 10.1109/ICPPW.2012.39.

Weicker, Reinhold P. "Dhrystone: a synthetic systems programming benchmark". In: *Commun. ACM* 27.10 (Oct. 1984), pp. 1013–1030. ISSN: 0001-0782. DOI: 10.1145/358274.358283. URL: https://doi.org/10.1145/358274.358283.

Williams, Samuel, Andrew Waterman, and David Patterson. "Roofline: An Insightful Visual Performance Model for Multicore Architectures". In: *Communications of the ACM* 52.4 (Apr. 2009), pp. 65–76. ISSN: 0001-0782, 1557-7317. DOI: 10.1145/1498765.1498785. URL: https://dl.acm.org/doi/10.1145/1498765.1498785 (visited on 07/10/2025).

Wilton, S.J.E. and N.P. Jouppi. "CACTI: an enhanced cache access and cycle time model". In: *IEEE Journal of Solid-State Circuits* 31.5 (May 1996), pp. 677–688. ISSN: 00189200. DOI: 10.1109/4.509850. URL: http://ieeexplore.ieee.org/document/509850/ (visited on 03/05/2025).