Projet

Analyse énergétique d'un processeur vectoriel pour des calculs de DNN
Rapport intermédiaire

ELE6307 - Machines neuronales : architectures et applications

Hiver 2022
Département de génie électrique
École Polytechnique de Montréal

Dernière mise à jour: 3 avril 2022

Yoan **Fournier** 1958736

Victor **Gaudreau-Blouin** 1958297

Introduction

Afin de modéliser l'efficacité énergétique du coprocesseur vectoriel ARA, le simulateur Timeloop a été utilisé. Ce simulateur utilise une structure hiérarchique pour modéliser une architecture. Une structure détaillant le problème à résoudre par l'architecture est aussi spécifiée. La commande *timeloop-mapper* permet ensuite de trouver automatiquement la correspondance idéale du problème sur l'architecture matérielle.

L'architecture simplifiée de ARA, comme mentionnée dans la proposition du projet, peut se modéliser en termes de *Lanes*. Chaque *Lanes* à sa plus simple expression consiste en une banque de 8 registres ainsi qu'une unité arithmétique capable d'effectuer des multiplications et des additions. Bien que sur ARA, l'unité d'addition est séparée de l'unité de multiplication, les deux unités peuvent fonctionner durant le même cycle sur des données indépendantes. Dans le cadre de la modélisation, on simplifie en utilisant la structure *intmac* fournie par l'outil Timeloop-Accelergy. Nous considérons que la différence causée par cette simplification est minimale. En effet, les opérations dans une Lane sont pipelinées et les opérations de multiplication et addition peuvent être exécutées en même temps. La seule différence principale est la latence causée par le pipelinage des opérations. Cependant, ce n'est pas une métrique de performance étudiée dans le cadre de ce projet.

L'architecture utilisée est présentée à la figure 1. On y observe une mémoire DRAM principale reliée à la mémoire cache L1 de 64 KB. La cache L1 est reliée à un nombre paramétrable de *Lanes*. Chaque Lane comporte sa banque de 8 registres et son unité de multiply-accumulate.

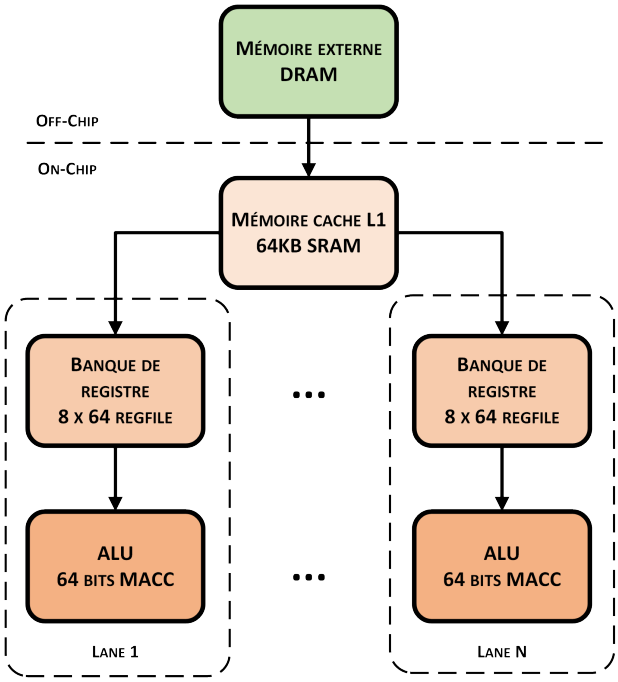

FIGURE 1 – Architecture de ARA sur Timeloop

Afin d'étudier la performance de l'architecture en fonction du nombre de *Lanes*, le problème utilisé est la première couche convolutionnelle du modèle VGG16. Cette couche comporte 64 filtres avec un *kernel* de 3×3. L'entrée de la couche est une image de dimension $224 \times 224 \times 3$.

Résultats

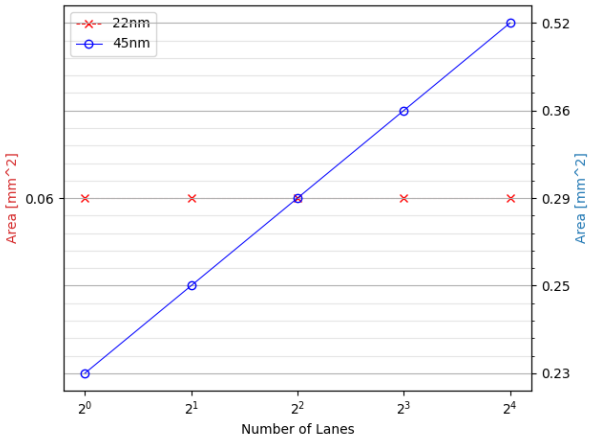

FIGURE 2 – Aire de ARA en fonction du nombre de *Lanes* pour la couche 1 de VGG16

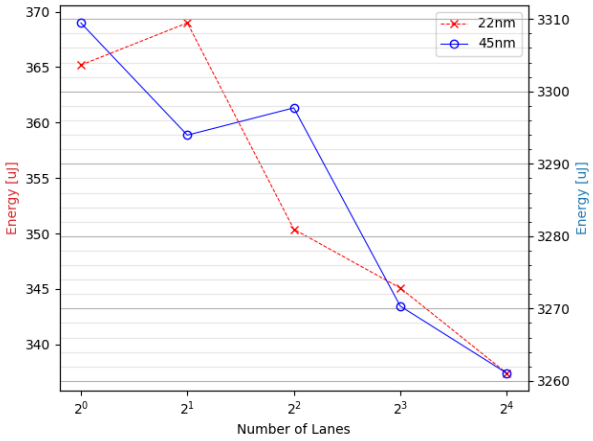

FIGURE 4 – Énergie de ARA en fonction du nombre de *Lanes* pour la couche 1 de VGG16

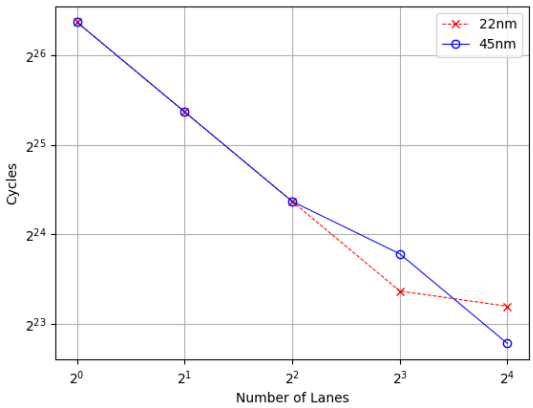

FIGURE 3 – Cycles de ARA en fonction du nombre de *Lanes* pour la couche 1 de VGG16

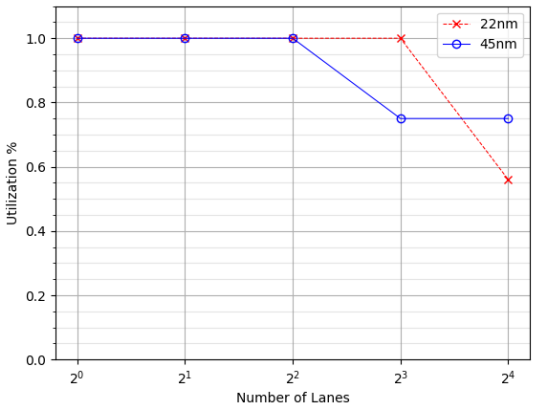

FIGURE 5 – Utilisation de ARA en fonction du nombre de *Lanes* pour la couche 1 de VGG16

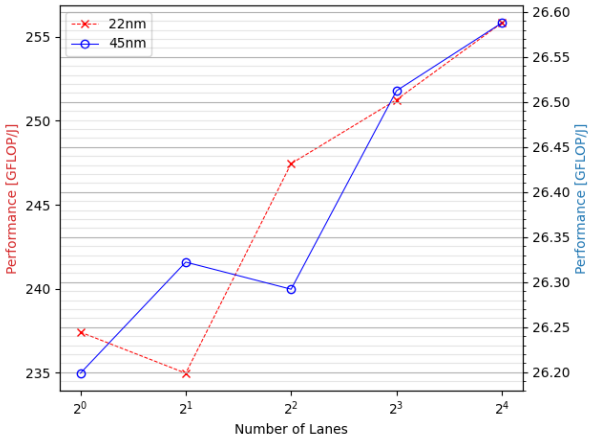

FIGURE 6 – Performance de ARA en GFLOP/J en fonction du nombre de *Lanes* pour la couche 1 de VGG16

On montre ci-haut les résultats d'aire, d'énergie, de nombre de cycles, d'utilisation et de performance de ARA en fonction du nombre de *Lanes* utilisé. Certains résultats sont attendus, comme la relation linéaire entre le nombre de *Lanes* et le nombre de cycles. En effet, en doublant le nombre de *Lanes*, il est normal de s'attendre à effectuer la tâche en la moitié du temps, pour autant que ces tâches sont indépendantes. Une certaine non-linéarité est observée pour les cas de 8 et 16 *Lanes*, mais ceci est dû à l'utilisation non optimale des ressources disponibles, comme on peut le constater sur le graphique d'utilisation. La technologie 22 nm tombe à 55 % d'utilisation à 16 *Lanes*, alors que la technologie 45 nm atteint une utilisation un peu plus haute, à 75 %, mais à partir de 8 *Lanes*.

L'aire utilisée croît de manière linéaire. Une particularité à noter est que la différence d'aire sur la technologie 22 nm est négligeable, parce que la cache occupe une très grande proportion de l'espace total. Le graphique d'énergie n'est pas aussi direct. On peut noter une tendance descendante dans le coût énergétique total en augmentant le nombre de *Lanes*, mais la courbe n'est pas exactement logarithmique. Ceci est dû en partie à la façon que l'assignation (*mapping*) du circuit est réalisée par Timeloop. Comme le problème d'assignation est complexe, il n'est pas garanti de donner la solution optimale pour un ou plusieurs critères. Nous avons la liberté de contraindre son espace de recherche pour accélérer la recherche, mais en baissant la probabilité de trouver la solution optimale. Il est donc possible que la solution optimale suive une courbe plus linéaire et surtout décroissante sur tout son intervalle. Un point intéressant à noter est que la technologie 45 nm semble avoir une amélioration moins grande que 22 nm, avec 1.5 % de différence entre l'énergie d'une *Lane* et celle à 16 *Lanes*, versus 8.9 % entre sa consommation maximale et minimale. Finalement, le graphe de performance illustre la combinaison de tous ces facteurs. Ce graphe est obtenu avec la formule

$$\text{GFLOP/J} = \frac{\%utilization \times \#cycles \times \#lanes}{E_{tot}}$$

. On observe une tendance croissante semblant logarithmique. Encore une fois, on peut constater que la technologie 22nm a une meilleure croissance de performance relative, avec 8.5 % versus 1.5 % pour 45 nm. De plus, on observe que la performance de la technologie 22 nm avec une seule *Lane* est très supérieure à la meilleure performance de la technologie 45 nm. Ceci nous indique que bien qu'on puisse observer des augmentations de performance en augmentant le nombre de *Lanes*, le plus grand facteur est la technologie utilisée.

Suite du projet

Pour le rapport final du projet, les aspects suivants seront étudiés plus en détail :

1. **Utilisation des PE pour 8 et 16 Lanes :** Dans les résultats, on observe que l'allocation n'est pas optimale pour 8 et 16 *Lanes*. L'utilisation des PE n'est pas de 100%. Il serait intéressant de voir si cela est causé par une recherche pas assez exhaustive des allocations ou si une allocation optimale est tout simplement impossible. Le rapport final cherchera à optimiser les paramètres de *timeloop-mapper* afin de tenter de converger vers de meilleures solutions.
2. **Étudier les allocations optimales :** Ce présent rapport ne présente pas l'allocation optimale trouvée par Timeloop. Le rapport finale présentera plus en détail cette allocation. En effet, le problème est automatiquement partitionné par Timeloop afin d'optimiser au maximum l'énergie consommée par l'architecture. Souvent cette métrique s'optimise en essayant de réduire au maximum les accès mémoires et d'optimiser la localité des données. L'analyse les allocations effectuées par Timeloop permettra de mieux comprendre l'approche optimale a utilisée et permettra de déceler les goulots d'étranglement dans l'architecture proposée.
3. **Tester l'impact des paramètres du problème :** Le seul problème étudié dans le cadre de ce rapport intermédiaire est la couche 1 du modèle VGG16. Afin d'avoir une meilleure idée de la performance de l'architecture, il serait intéressant de faire varier les paramètres du problème comme le kernel ou la taille des tenseurs d'entrée. On pourrait y observer une diminution de l'utilité d'avoir plusieurs Lanes lorsque les problèmes sont plus petits puisqu'il devient plus difficile de paralléliser les calculs.