Projet

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

ELE6307 - Machines neuronales : architectures et applications

Hiver 2022
Département de génie électrique
École Polytechnique de Montréal

Dernière mise à jour: 20 avril 2022

Yoan **Fournier** 1958736

Victor **Gaudreau-Blouin** 1958297

1 Introduction

Dans les dernières années, l'utilisation de réseaux de neurones profonds (DNN) pour résoudre différentes tâches a beaucoup augmenté. Ces DNNs nécessitent des puissances de calculs considérables, mais à la fois nécessitent une bonne efficacité énergétique étant donné leur déploiement dans des appareils mobiles. Bien que les processeurs généralistes que l'on retrouve aujourd'hui ont augmenté rapidement en performance, les limitations du *Dennard Scaling* se font ressentir. Il est donc souhaitable de trouver une autre approche pour accélérer de façon efficace les calculs dans les DNNs. Une manière intéressante pour l'accélération des calculs pour des DNNs est l'utilisation de processeurs vectoriels plutôt que des processeurs scalaires standards. Ces processeurs utilisent un jeu d'instruction SIMD (*Single instruction, Multiple Data*) plutôt que SISD (*Single instruction, Single data*). Ainsi une instruction SIMD peut performer la même opération sur plusieurs données plutôt qu'une seule, ce qui permet de plus facilement augmenter le parallélisme des calculs.

Le présent rappot étudie l'efficacité énergétique d'un modèle simplifié du processeur vectoriel ARA sur différents problèmes en utilisant le simulateur *Timeloop-Accelergy*.

2 Méthodologie

L'architecture sur laquelle notre modèle se base est sur celle du coprocesseur vectoriel ARA [**?**]. Ce coprocesseur est une extension vectorielle du processeur scalaire RISCV CVA6. Le fonctionnement d'ARA se base sur le concept de *Lanes*. La figure **??** présente l'architecture d'ARA [**?**]. Le coprocesseur ARA possède un nombre d'allées, ou *Lanes*, qui sont capables de traiter des éléments vectoriels indépendamment. Le routage des données vers ces *Lanes* grandit évidemment en complexité avec le nombre de *Lanes*. Le *Sequencer* s'occupe de lire les instructions à exécuter et transmettre les informations correctes au *Vector Load and Store Unit* (VLSU), *Slide Unit* (SLDU) ou aux *Lanes*. Le VLSU est responsable de générer les bonnes addresses, amenant les données séquentiellement vers et depuis les *Lanes*. Le SLDU exécute des opérations qui n'ont pas d'indépendance entre les *Lanes*. Finalement, chaque *Lane* a son propre *sequencer* pour les instructions à exécuter sur son élément vectoriel. Le reste de la *Lane* peut être considéré comme un PE, avec sa mémoire locale, un étage de conversion et un étage d'exécution.

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.

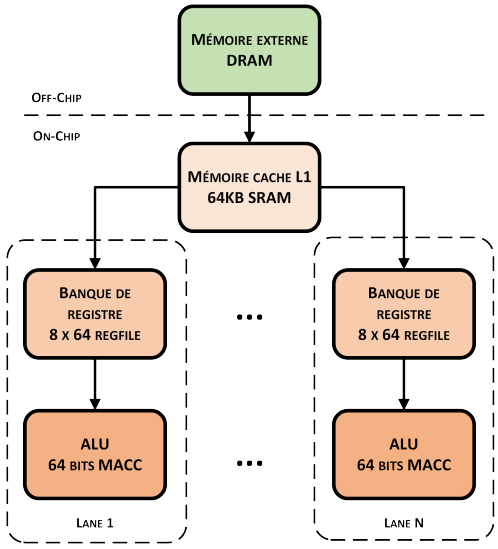

Figure 1 – Architecture de ARA sur Timeloop

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'ef-

fectuer 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*.

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 $224 \times 224 \times 3$.

3 Résultats

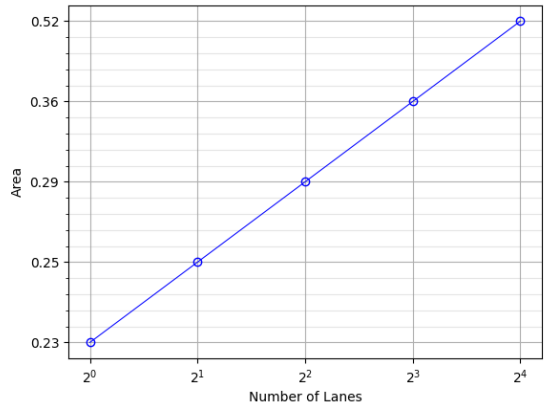

FIGURE 2 – Aire de ARA en fonction du nombre de *Lanes*

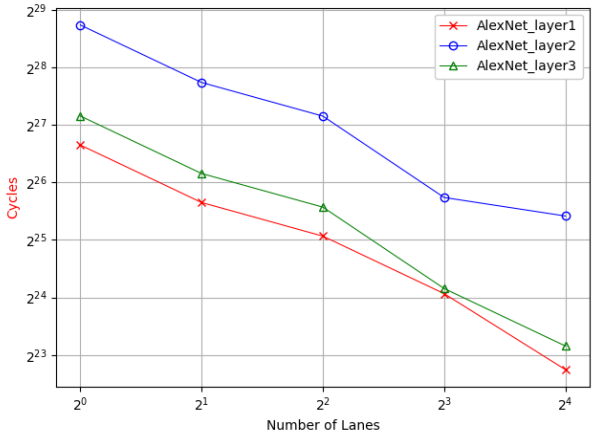

FIGURE 3 – Cycles de ARA en fonction du nombre de *Lanes* pour le problème AlexNet

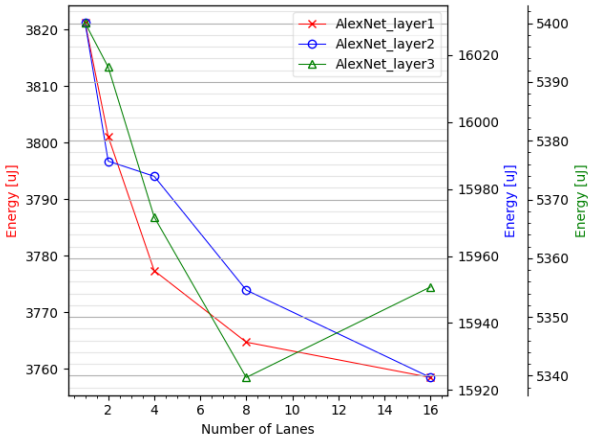

FIGURE 4 – Cycles de ARA en fonction du nombre de *Lanes* pour le problème AlexNet

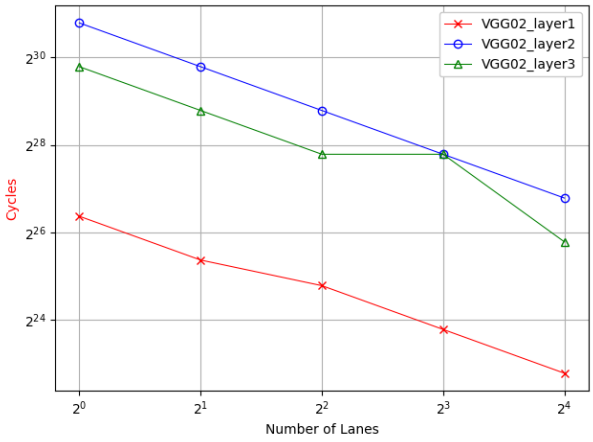

FIGURE 6 – Cycles de ARA en fonction du nombre de *Lanes* pour le problème VGG16

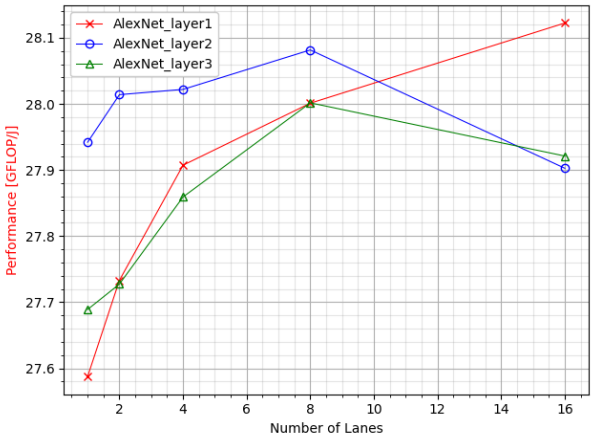

FIGURE 5 – Cycles de ARA en fonction du nombre de *Lanes* pour le problème AlexNet

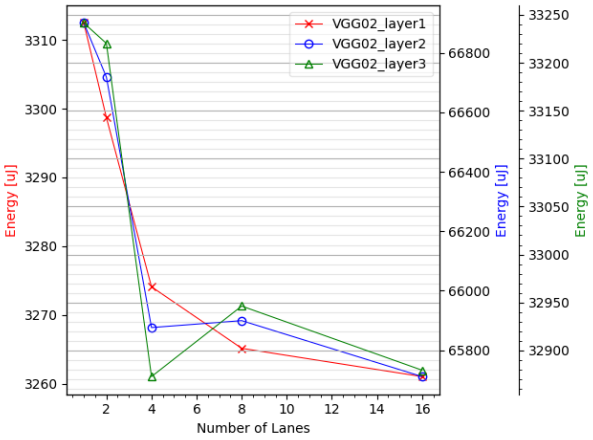

FIGURE 7 – Cycles de ARA en fonction du nombre de *Lanes* pour le problème VGG16

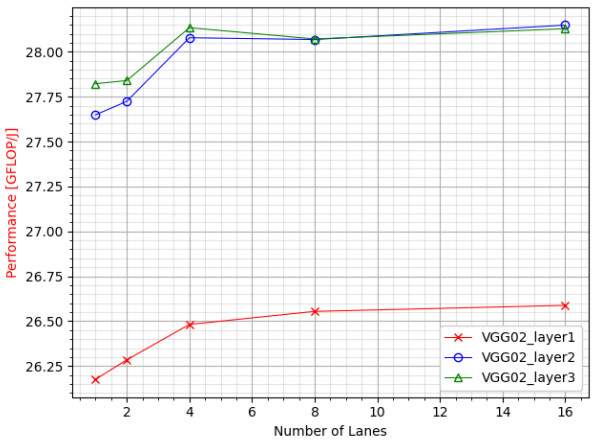

FIGURE 8 – Cycles de ARA en fonction du nombre de *Lanes* pour le problème VGG16

Problem	**AlexNet_layer2**	
Parameters	M=256 K=5x5 C=96 I=27x27	
n	16	
Mapper	Timeloop	Custom
Utilization	0.62	1
Cycles [Millions]	44.8	28.0
Energy [mJ]	15.92	15.90
Perf [GFLOP/J]	27.90	28.18

TABLEAU 1 – Mapping Timeloop et mapping custom pour AlexNet L2

4 Conclusion

Annexe

Problem	**AlexNet_layer1**					**AlexNet_layer2**					**AlexNet_layer3**				
Parameters	M=96 K=11x11 C=3 I=55x55					M=256 K=5x5 C=96 I=27x27					M=384 K=3x3 C=256 I=13x13				
n	1	2	4	8	16	1	2	4	8	16	1	2	4	8	16
Utilization	1	1	0.75	0.75	0.94	1	1	0.75	1	0.62	1	1	0.75	1	1
Cycles [Millions]	105.4	52.7	35.1	17.6	7.3	447.9	223.9	149.3	56.0	44.8	149.5	74.8	49.8	18.7	9.3
Energy [mJ]	3.82	3.80	3.77	3.77	3.76	16.03	15.99	15.98	15.95	15.92	5.40	5.39	5.36	5.34	5.35
Perf [GFLOP/J]	27.59	27.73	27.91	28.00	28.12	27.94	28.01	28.02	28.08	27.90	27.69	27.73	27.86	28.00	27.92

TABLEAU 2 – Résultats pour le problème AlexNet

[Problem	**VGG02_layer1**					**VGG02_layer2**					**VGG02_layer3**				
Parameters	M=64 K=3x3 C=3 I=224x224					M=64 K=3x3 C=64 I=224x224					M=128 K=3x3 C=64 I=112x112				
n	1	2	4	8	16	1	2	4	8	16	1	2	4	8	16
Utilization	1	1	0.75	0.75	0.75	1	1	1	1	1	1	1	1	0.5	1
Cycles [Millions]	86.7	43.4	28.9	14.5	7.2	1850	924.8	462.4	231.2	115.6	925	462	231	231	57.8
Energy [mJ]	3.31	3.30	3.27	3.27	3.26	66.9	66.7	65.9	65.9	65.7	33.2	33.2	32.9	32.9	32.9
Perf [GFLOP/J]	26.18	26.28	26.48	26.55	26.59	27.65	27.72	28.08	28.07	28.15	27.82	27.84	28.13	28.07	28.13

TABLEAU 3 – Résultats pour le problème VGG16