Université Paris-Saclay
UFR des Sciences
Magistère SPI

Méthodes d'accélération de CNN sur architectures reconfigurables

Réalisé par :

Mohamed Younes BENAMARA

Encadré par :

Abdelhafid ELOUARDI

Gaele PERRUSSON

Table des matières

20 janvier 2025

Résumé

Ce travail de recherche est effectué afin d'étudier des methodes avancées pour accélérer les réseaux de neurones convolutifs (CNN) sur des architectures reconfigurables, telles que les FPGA, et ce dans l'objectif d'etendre la faisabilité et la portabilité de ces méthodes sur des systémes où la consommation energetique et la puissance de calcul sont asujetti à des contraintes. En analysant des techniques comme la quantification, la compression de réseau et l'optimisation des couches convolutives, nous cherchons à améliorer l'efficacité temps réel tout en réduisant la consomation. Cette étude inclut également des stratégies de parallélisme et de pipeline exploitant les capacités des architectures reconfigurables. Le travail à faire consiste à expliquer et detailler la méthode presente dans l'article A1, puis de comparer cette méthode aux méthodes présentes dans les autres articles A2 A3 et A4.

1 La méthode A1

1.1 Introduction

L'architecture RISC-V offre des capacités d'extension pour ses instructions qui permettent personnalisation approfondie pour des applications spécifiques telles que le calcul de réseaux de neurones (CNN) sur les plateformes périphériques. Cet article propose un ensemble de sept instructions SIMD personnalisées, destinées à optimiser les calculs de convolution, vation et mise en commun dans les CNN. Utilisant l'algorithme de Winograd, ces instructions visent pour réduire considérablement le nombre de cycles d'exécution des modèles CNN, et un processus Le serveur RISC-V modifié appelé RI5CY-Accel a été conçu et implémenté sur FPGA pour évaluer cela approche.

1.2 Extensions des Instructions RISC-V

1.2.1 Contexte et Motivation

La norme RISC-V définit un ensemble d'instructions de base qui favorisent une conception efficace. codage et exécution des commandes. Cependant, pour répondre aux besoins d'applications spécifiques Plus précisément, le jeu d'instructions RISC-V permet l'ajout d'instructions personnalisées. Ce type de personne la personnalisation est particulièrement pertinente dans le cas des CNN, où les opérations de convolution représentent plus de 90 % de la charge de travail. Les consignes Les constructions de base nécessitent un accès fréquent à la mémoire, ce qui entraîne de la latence et de la consommation haute énergie.

1.2.2 Ensemble d'Instructions Personnalisées

Les auteurs ont introduit sept instructions SIMD pour accélérer les opérations CNN :

— **CONV23** : instruction de convolution entre une matrice d'entrée 4x4 et un noyau 3x3.
— **WB23** : instruction de sauvegarde des résultats de CONV23 en mémoire.
— **MP_RI, MAX_POOL, MP_WB** : instructions pour les opérations de max pooling.
— **RELU** : activation selon la fonction ReLU.
— **W_WB** : mise à jour du noyau de convolution.

1.3 L'Algorithme de Convolution

1.3.1 Comparaison des Algorithmes de Convolution

Pour accélérer les calculs de convolution, plusieurs algorithmes peuvent être envisagés, notamment-im2col, la transformée de Fourier rapide (FFT) et l'algorithme de Winograd. Ce dernier est bien adapté aux petits noyaux comme ceux de 3x3, réduisant la complexité des calculs en limitant le nombre de multiplications.

L'algorithme de transformation de Fourier rapide (FFT) n'est pas optimal pour les petits noyaux car cela nécessite des étapes de transformation supplémentaires (transition dans le domaine fréquentiel et retour au domaine spatial), avec une surcharge en calcul et en mémoire. Pour les petits noyaux, le le coût de ces transformations dépasse les gains, ce qui rend la FFT plus efficace pour les grandes convolutions où le nombre de multiplications est considérablement réduit.

L'approche im2col transforme l'image en une matrice de « patchs » pour utiliser des multiplicateurs. Mais cela augmente considérablement la taille des données. Pour les petits noyaux, La méthode génère une redondance importante dans la matrice de correction, ce qui consomme des ressources inutiles et augmente le coût de préparation des données. Il est donc mieux adapté à des noyaux et des architectures plus gros avec des capacités de multiplication matricielle optimisées.

1.3.2 Introduction à l'Algorithme de Winograd

L'algorithme de Winograd, lorsque combiné à une convolution 3x3, réduit le nombre de multiplications de 36 à 16 lorsqu'il est appliqué à des sous-matrices de l'image et du noyau suivies par une addition. Avec l'utilisation de projections du noyau, de ses transposées et de matrices de transformations Hadamard, la technique permet de réduire le temps de calcul sans pour autant augmenter le nombre de cycles nécessaires à la sommation.

1.3.3 Illustration de l'Algorithme de Winograd

L'algorithme Winograd est un algorithme qui optimise un calcul de convolution. Si vous faites une convolution entre une image de 4×4 sur une image de 3×3 Nicolas Winograd a trouvé une méthode pour réaliser cette opération en effectuant seulement 16 multiplications au lieu des 36 que l'on peut faire en utilisant une approche plus directe.

Formulation Matricielle de Winograd

L'algorithme Winograd est basé sur la transformation de matrices de convolution en produits de matrices. En notation matricielle, le produit s'exprime comme suit :

$$\mathbf{R} = \mathbf{A}^T \cdot \left((\mathbf{G} \cdot \mathbf{K} \cdot \mathbf{G}^T) \odot (\mathbf{B}^T \cdot \mathbf{D} \cdot \mathbf{B}) \right) \cdot \mathbf{A}$$

où : - $\mathbf{D}$ est la matrice d'entrée 4×4, - $\mathbf{K}$ est le noyau de convolution 3×3, - $\mathbf{R}$ est le résultat 2×2, - $\mathbf{G}$, $\mathbf{B}$, et $\mathbf{A}$ sont les matrices de transformation spécifiques à l'algorithme de Winograd, et - $\odot$ représente le produit de Hadamard (multiplication élément par élément).

Définition des Matrices de Transformation

Les matrices de transformation sont définies comme suit pour la convolution $F(2 \times 2, 3 \times 3)$:

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & -1 & -1 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix}, \quad \mathbf{G} = \begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 1 \end{bmatrix}$$

Soit la matrice d'entrée $\mathbf{D}$ et le noyau de convolution $\mathbf{K}$ donnés par :

$$\mathbf{D} = \begin{bmatrix} d_{00} & d_{01} & d_{02} & d_{03} \\ d_{10} & d_{11} & d_{12} & d_{13} \\ d_{20} & d_{21} & d_{22} & d_{23} \\ d_{30} & d_{31} & d_{32} & d_{33} \end{bmatrix}, \quad \mathbf{K} = \begin{bmatrix} k_{00} & k_{01} & k_{02} \\ k_{10} & k_{11} & k_{12} \\ k_{20} & k_{21} & k_{22} \end{bmatrix}$$

En utilisant l'algorithme de Winograd, le produit matriciel est calculé en 4 étapes :

1. Transformation de l'entrée : $\mathbf{B}^T \cdot \mathbf{D} \cdot \mathbf{B}$

2. Transformation du noyau : $\mathbf{G} \cdot \mathbf{K} \cdot \mathbf{G}^T$

3. Produit de Hadamard entre les matrices transformées de l'entrée et du noyau

4. Transformation inverse pour obtenir le résultat final.

$$\mathbf{R} = \begin{bmatrix} r_{00} & r_{01} \\ r_{10} & r_{11} \end{bmatrix}$$

où chaque élément de $\mathbf{R}$ est obtenu à partir des sommes et produits obtenu par Winograd, réduisant ainsi le nombre total de multiplications.

1.4 Module d'Accélération

Le module d'accélération ressemble à un coprocesseur intégré à une unité de traitement RI5CY. Dès lors qu'une instruction personnalisée est exécutée, le module d'accélération accède directement aux données via l'antenne. Le module d'accélération comporte trois blocs dont la matrice de convolution, un bloc de mise en commun et une fonction d'activation spécialisés pour minimiser la latence et le débit de mémoire.

1.4.1 Fonctionnement en tant que co-processeur

— **Co-processeur intégré** : En tant que coprocesseur, il peut exécuter les instructions spécifiques gourmandes en traitement (comme la convolution, le pooling et l'activation) sans passer par le pipeline RISC-V habituel. Lorsqu'une instruction spécifique est envoyée, le processeur principal la décode et redirige cette instruction vers le bloc d'accélération.

1.4.2 Lecture et écriture via le Load Store Unit

— **Load Store Unit** : Le module d'accélération utilise le LSU pour accéder directement aux données en mémoire, ce qui facilite les opérations de lecture et d'écriture. Cela permet au module de charger les données (images, noyaux, etc.) et stocker les résultats intermédiaires sans important temps de latence.

1.4.3 Structure interne

Le module d'accélération est structuré en trois unités principales, chacune étant responsable d'une opération spécifique :

— **Unité de Convolution** : Cette unité effectue des opérations de convolution en appliquant le noyaux aux matrices d'entrée. Il est optimisé à l'aide de l'algorithme de Winograd precedemment vu, qui réduit le nombre de multiplications nécessaires. L'unité de convolution fonctionne dans plusieurs États pour organiser les étapes de lecture des données, de calcul et de stockage des résultats.

— **Unité de Pooling** : Cette unité effectue des opérations de regroupement, telles que le max-pooling, qui consiste réduire la taille des données en extrayant les valeurs maximales des sous-matrices de la matrice d'entrée. Il fonctionne à l'aide de trois instructions dédiées :

  — MP_RI : pour charger les données dans l'unité de pooling.
  — MAX_POOL : pour calculer la valeur maximale dans chaque sous-région.
  — MP_WB : pour écrire la valeur de pooling calculée en mémoire.

— **Unité d'Activation** : Cet appareil utilise la fonction d'activation ReLU, qui met toutes les valeurs négatives à zéro, laissant les valeurs positives. Il peut également ajouter un biais optionnel à chaque valeur. Tout comme les autres unités, il maintient toujours l'altitude des lignes graphiques pour peu quelle est significative pour prédire et minimiser le volume de transferts de données vers la mémoire.

1.4.4 Optimisation de la Latence et des Transferts de Mémoire

Dans Fusion d'Instructions, les opérations de convolution, de regroupement et d'activation sont effectuées sans avoir besoin de transferts de mémoire entre chaque étape. Par exemple, après une convolution, les données peuvent être directement utilisé par l'unité de pooling puis passées à l'activation avant d'être stockées en mémoire. Cette fusion permet de réduire la latence totale en minimisant les transferts et améliore ainsi la latence de chaque étape de calcul. Chaque unité de calcul conserve en interne les données intermédiaires le plus longtemps possible. Cela limite l'accès mémoire à des étapes cri-tiques, comme le début et la fin des calculs. L'avantage est une meilleure utilisation du cache et une réduction des délais dus aux lectures et écritures externes. En résumé, cette architecture modulaire permet au module d'accélération de traiter efficace- ment les opérations CNN à fortement paralléliser. Elle permet de réduire les goulots d'étrange- ment d'accès à la mémoire et d'optimiser la latence de chaque étape de calcul.

En résumé, cette architecture modulaire permet au module d'accélération de traiter efficacement les opérations CNN à forte intensité de calcul, réduisant ainsi les goulots d'étranglement d'accès à la mémoire et optimisant la latence de chaque étape de calcul.

1.5 États et Transitions du Module CONV23

L'instruction CONV23 fonctionne comme une machine à cinq états : IDLE, GET-DATA, CAL1, CAL2 et CAL3. La convolution s'effectue ainsi en trois cycles d'horloge dédiés aux calculs Winograd Ce procédé améliore la rapidité de calcul en regroupant les étapes d'accès et de transformation en mémoire dans une séquence continue et efficace.

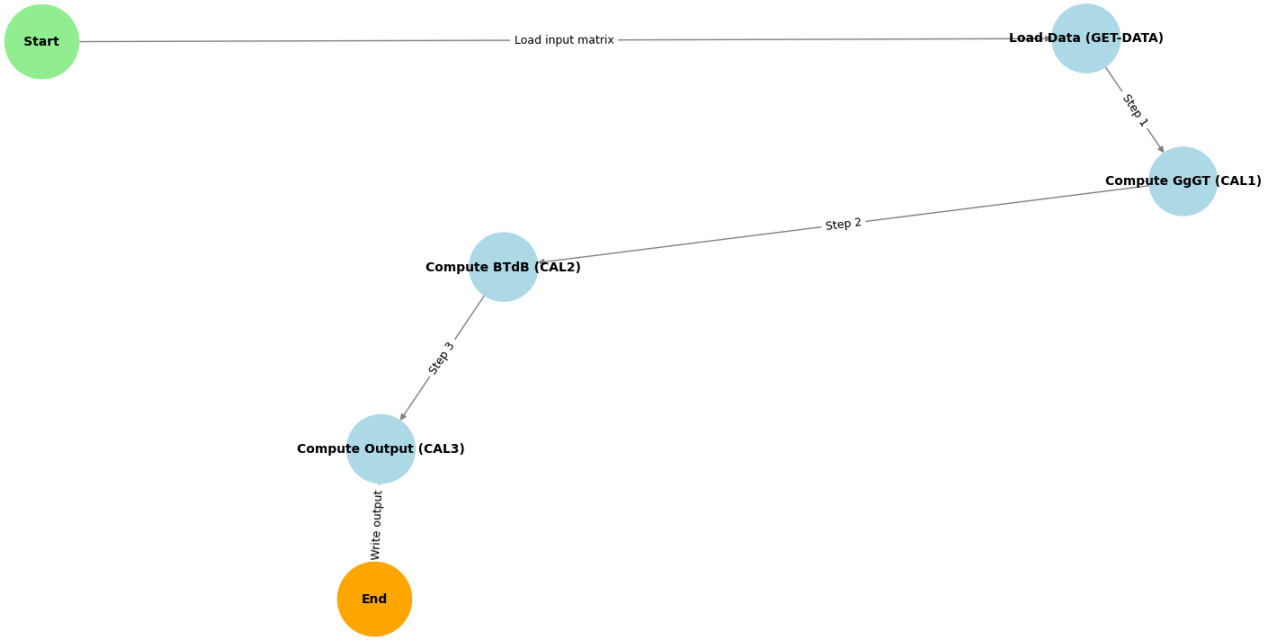

FIGURE 1 – Representation schématique de CONV23

1.6 Optimisation des Opérations de Pooling et d'Activation

1.6.1 Unités de Pooling et d'Activation

Le pooling est implémenté à l'aide de trois instructions personnalisées, permettant l'exécution déroulement rapide de l'opération en parallèle des autres calculs. Pour l'activation, l'instruction RELU permet l'activation de tous les éléments en un seul cycle, réduisant la latence de 75 % par rapport à instructions standards.

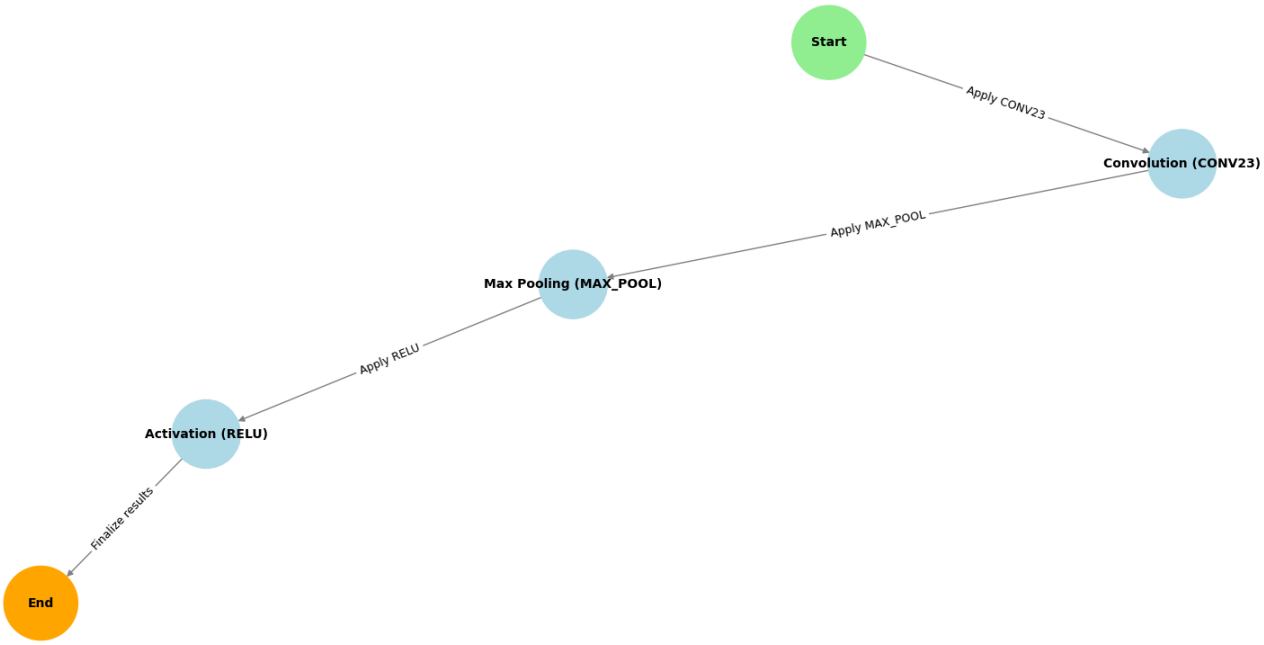

FIGURE 2 – Representation de l'execution d'un cycle

1.6.2 Fusion des Instructions pour Minimiser les Accès Mémoire

Pour combiner convolution et pooling en une seule opération, il n'est pas nécessaire de mettre à jour la mémoire. En outre, lors de cette combinaison, les accès aux données sont minimisés et les performances énergétiques de la puce sont augmentées, ce qui entraîne une augmentation significative des performances des réseaux.

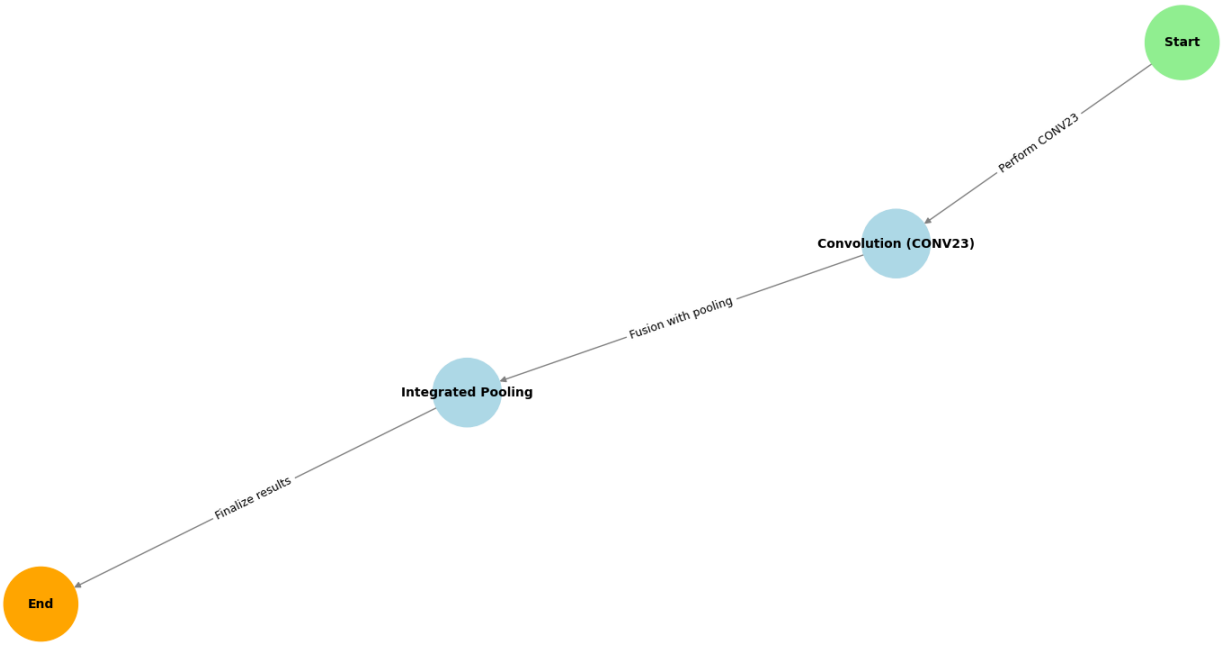

FIGURE 3 – Fusion de couches

2 Comparaison avec les Méthodes Alternatives

Dans ce chapitre, nous comparons la méthode de A1 avec les approches décrites dans A2,A3 et A4.

2.1 Comparaison avec A2

Les algorithmes de détection d'objets à base de réseaux de neurones convolutifs (CNN) sont très demandeurs en opérations de calcul et en accès mémoire, ce qui a tendance à créer des goulots d'étranglement et à faire chauffer les ordinateurs. Pour répondre à ces défis tout en gardant l'efficacité d'un système de détection d'objets temps réel, nous proposons dans cet article une architecture matérielle pour accélérer le fonctionnement du modèle CNN YOLO dans un FPGA avec un haut débit et une faible consommation énergétique. Pour ce faire nous utilisons la faible précision et une architecture Pipeline.

2.1.1 Méthodologie

1. **Quantification Matérielle** :
   - Modèles quantifiés à 1 bit et activations à 3-6 bits, réduisant la consommation des ressources matérielles.
   - Le modèle tient en mémoire (Block RAM) tout le temps, plus de lectures en flash
2. **Architecture de Streaming Scalable** :
   - Traitement des couches convolutives en pipeline.
   - Transmission directe des données intermédiaires à la couche suivante sans mémoire externe.

3. **Réutilisation des Données** :
   — Utilisation des Features Maps en tant que données intermédiaires pour réduire le nombre de buffer et les coûts.
4. **Batch Processing** :
   — Prise en charge du traitement par lots pour exploiter l'unité de traitement graphique.

2.1.2 Optimisations Clés

1. **Normalisation et Quantification** :
   — Le matériel exécute la normalisation par lot.
   — Remplace la fonction d'activation ReLU par Leaky ReLU avec des coefficients prédéfinis.
2. **Réorganisation des Boucles** :
   — Réorganise et tuile les boucles pour un scheduling et une plus grande réutilisation.
3. **Pipelines Paramétrés** :
   — Le nombre de groupes et la taille des filtres des couches convolutives peuvent être modifiés sans effort.

2.1.3 Architecture Matérielle

1. **Convolution** :
   — Tampons linéaires pour traiter les lignes en parallèle.
   — Tampons ping-pong pour minimiser la latence des poids.
2. **Pooling** :
   — Comparaison ligne par ligne avec des tampons de profondeur réduite.
3. **Maximisation des Ressources** :
   — Paramètres de parallélisme (Ti et To) optimisés pour équilibrer débit et coût matériel.

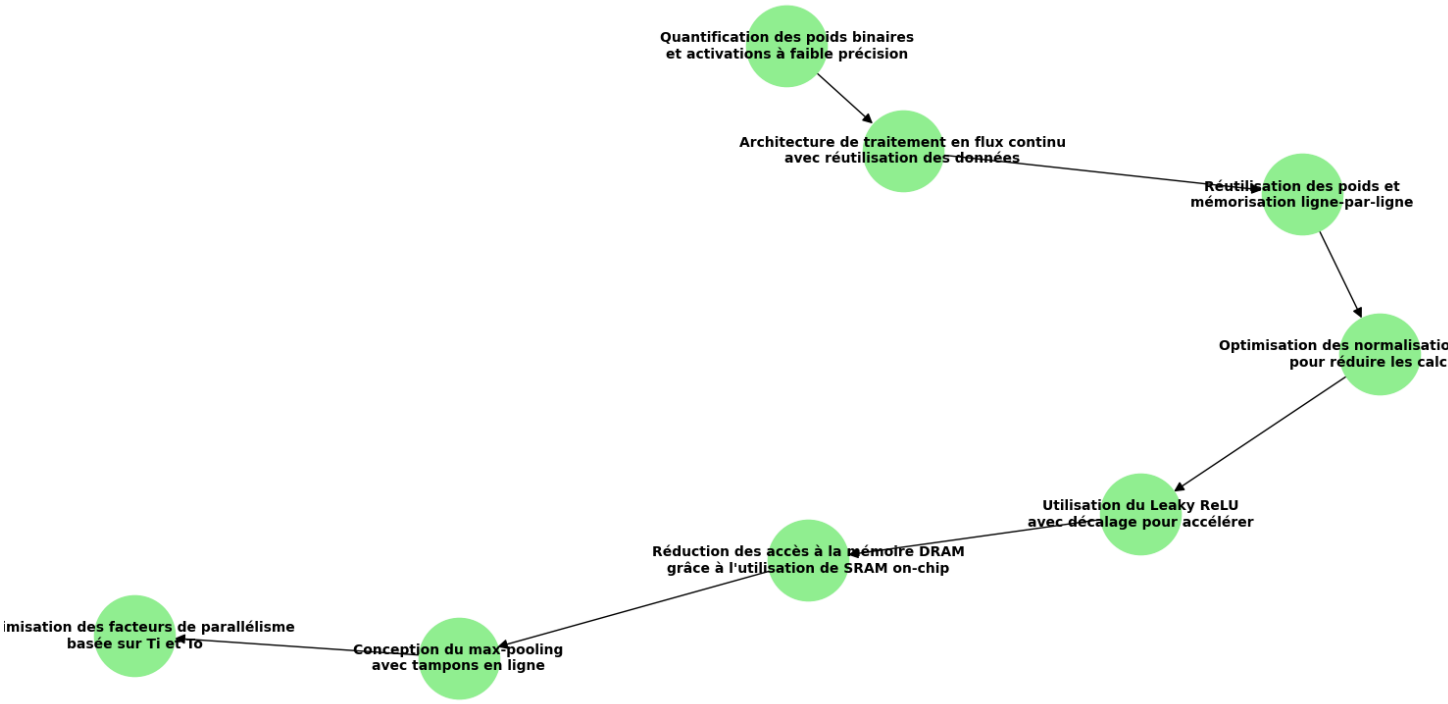

FIGURE 4 – Schématisation de la méthode

2.1.4 Résumé des Méthodes

RISC-V avec SIMD et Winograd

- **Objectif :** Cette contribution propose de mettre en place une solution efficace et évolutive dédiée à l'exécution d'algorithmes de réseaux de neurones à convolution sur les puces de Silicon et d'introduire des instructions SIMD personnalisées dans le cœur de processeur RISC-V.
- **Optimisations :**
  - **Instructions SIMD personnalisées :** Afin de réduire la latence des algorithmes de réseaux de neurones à convolution sur les puces de Silicon, des instructions SIMD personnalisées comme CONV23 et RELU sont ajoutées au jeu d'instructions du processeur RISC-V. L'objectif est de réduire de beaucoup la latence des algorithmes CNN sur les puces de Silicon.
  - **Algorithme de Winograd :** Réduit le nombre de multiplications nécessaires pour les convolutions avec des noyaux 3×3, passant de 36 à 16 multiplications pour des matrices d'entrée 4×4.
  - **Module d'accélération intégré :** Conçu comme un co-processeur pour exécuter des instructions spécialisées, minimisant les cycles d'accès mémoire.
- **Performances clés :**
  - Approprié pour des réseaux légers comme LeNet et VGG16.
  - Réduction des cycles d'exécution et des latences pour les tâches spécifiques.

FPGA YOLO

- **Objectif :**Comment fournir un haut degré de performances à YOLO lors de l'inférence d'objets en temps réel.

— **Optimisations :**
  — **Quantification basse précision :** Utilisation de poids binaires et d'activations à faible résolution (3-6 bits) pour réduire la consommation mémoire et les accès DRAM.
  — **Architecture en streaming :** traitement de données ligne par ligne des images, réutilisation des valeurs intermédiaires, ne quasiment plus de trafic mémoire externe.
  — **Planification optimisée :** utiliser plusieurs fois les valeurs/opérands des poids et valeurs locales pour minimiser l'ajout de matériel supplémentaire
— **Performances clés :**
  — Débit jusqu'à 1.877 TOPS avec une consommation énergétique de 18.29 W.
  — Supporte des réseaux complexes comme YOLO et leurs variantes.

Critère	**RISC-V SIMD + Winograd**	**FPGA YOLO**
Matériel cible	Processeur RISC-V	FPGA
Optimisation mémoire	Instructions dédiées	Réutilisation intensive interne
Complexité matérielle	Faible	Élevée
Support algorithme	Winograd (réduction des multiplications)	Quantification basse précision
Performance	Efficace pour CNN légers	Excellente pour réseaux complexes
Consommation énergétique	Très faible	Optimisée
Flexibilité	Extensions programmables	Configurable à travers le matériel
Déploiement	Adapté aux systèmes embarqués	Réseaux nécessitant un haut débit

TABLE 1 – Comparaison entre RISC-V SIMD + Winograd et FPGA YOLO.

2.1.5 Analyse Technique

Optimisation des Calculs Convolutifs :

— **RISC-V SIMD :** La multiplication partielle de Winograd est idéale avec des noyaux plus petits car elle cause moins de multiplications, tout en maintenant des performances élevées.
— **FPGA YOLO :** La quantification réduit la largeur des données et les coûts en énergie et mémoire associés.

Gestion des Données :

— **RISC-V SIMD :** Les instructions SIMD personnalisées permettent puisque elles com- binent plusieurs opérations en une, ce qui implique moins de copies en mémoire.
— **FPGA YOLO :** L'architecture en streaming délaisse presque complètement les accès à la DRAM pour préférer stocker, traiter et partager les données instantanément.

Scalabilité et Applicabilité

— **RISC-V SIMD :** Idéal pour les applications embarquées nécessitant des optimisations sur mesure où l'autonomie est un enjeu.

— **FPGA YOLO :** Idéal pour les tâches de traitement grâce à des débits élevés et une haute précision, comme la détection d'objets sur des images vidéo en direct.

2.1.6 Conclusion

L'association des technologies RISC-V SIMD et Winograd fonctionne au mieux sur des actionnaires avec des restrictions telles que des bandes de craie, car davantage de gadgets. A l'inverse, un modèle YOLO qui ne montre aucune faiblesse en termes de dépense martiale et dispose d'une performance standard a l'avantage de s'adapter performance nécessitant un debit video élevé.

2.2 Comparaison avec A3

2.2.1 Résumé de la Méthode

Dans un environnement edge, avec des ressources matérielles limitées telles que les drones ou les voitures autonomes, l'efficacité énergétique de CNN est cruciale. Nous présentons dans cette étude un accélérateur configurables de CNN sur un mobile FPGA, permettant une consommation énergétique optimisée et un traitement en temps-réel.

2.2.2 Conception Basée sur le Niveau RTL

La méthodologie adoptée repose sur une optimisation des blocs de calcul (multiplicateurs et additionneurs) pour améliorer l'efficacité énergétique. Les unités de calcul sont conçues de manière modulaire et hiérarchique pour s'adapter à diverses architectures CNN comme ResNet-20 et YOLO.

2.2.3 Optimisations Matérielles pour la Réduction de Puissance

— **Clock Gating (CG)** : Cette technique consiste à désactiver les cycles d'horloge inutiles pour réduire la consommation dynamique.
  — *Local Explicit Clock Enable (LECE)* : Utilise un signal d'activation pour éviter de toggle inutilement.
  — *Bus-Specific Clock Gating (BSCG)* :Utilise des gates XOR pour économiser de l'energi dans les compareurs de signaux.

— **Optimisation du Multiply-Accumulate (MAC)** : Une version modifié du MAC ajoute des flip-flops et des gates pour éviter de recharger au mauvais moment, ce qui réduit ainsi la consommation énergétique. En gros On trivialise la BSCG dans le MAC.

2.2.4 Architecture de l'Accélérateur

La conception de l'accélérateur repose sur des blocs modulaires personnalisés :

— **Bloc IP principal** : Contient des unités MAC, des planificateurs d'accès mémoire, et des modules de convolution.

— **Modularité** : Utilisation et conception hierarchique de blocs IP custom pour faciliter un portage rapide sur toutes les architectures de réseaux de neurones profonds.

Le noyau est testé sur la carte FPGA PYNQ-Z1 et la structure ResNet-20 entraînée sur le jeu de données CIFAR-10. La procédure classique de déroulement d'un projet qui va du lancement du projet jusqu'à la validation physique est la suivante :

1. Génération de code RTL de base à partir d'un ou deux scripts Python à l'aide de l'outil Tensil.
2. Optimisation de grande envergure pour amoindrir la surface et la quantité de ressources.
3. Conception d'instance de votre noyau en VHDL/Verilog à l'aide de Vivado pour ensuite le lancer.
4. Simulation sur FPGA et mesure de la consommation d'energie.

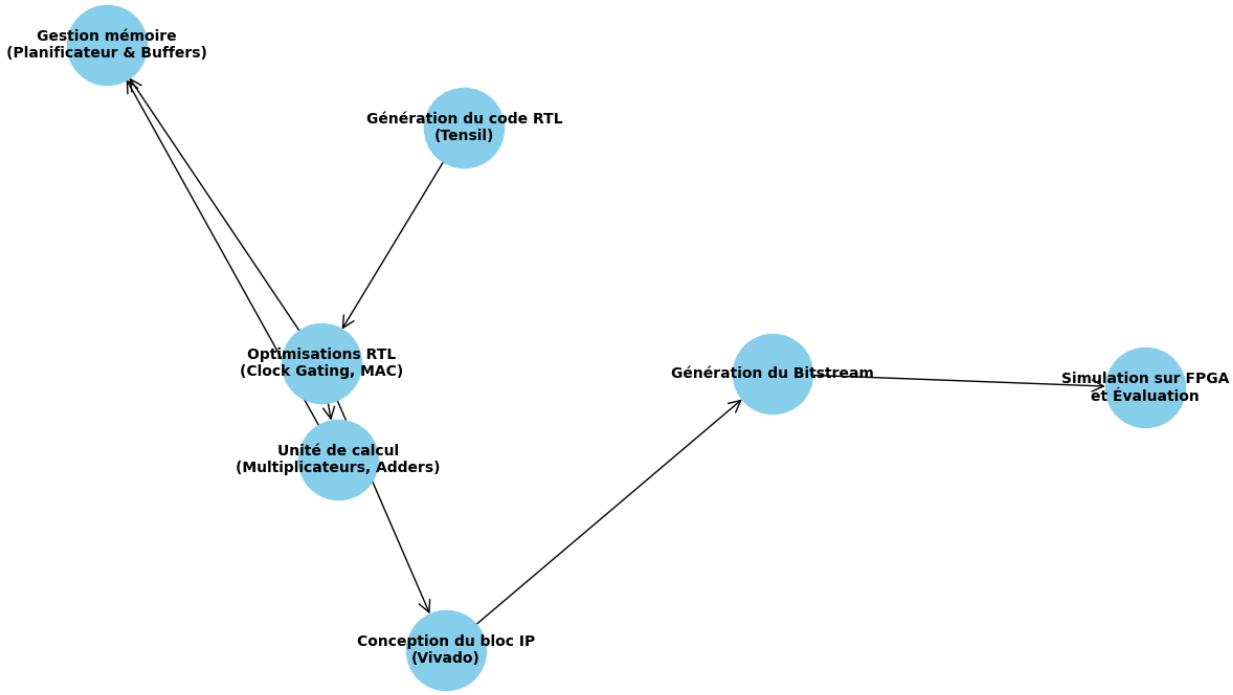

FIGURE 5 – Schematisation du flux de conception

2.2.5 Comparaison Technique

TABLE 2 – Comparaison des méthodes RISC-V et FPGA

Critère	**RI5CY-Accel (RISC-V)**	**Architecture FPGA Reconfigurable**
Approche principale	Instructions SIMD personnalisées (*CONV23*, *WB23*, etc.).	Optimisations RTL (*clock gating*, modularisation des unités *MAC*).
Optimisation des convolutions	Algorithme de Winograd réduisant les multiplications de 36 à 16.	Gestion efficace des unités *MAC* et du pipeline.
Gestion mémoire	Exécution en mode batch pour limiter les accès mémoire redondants.	Fusion des couches convolutionnelles et de pooling pour minimiser les transferts.
Cible matérielle	Processeur RISC-V modifié (*RI5CY-Accel*) implémenté sur FPGA PYNQ-Z2.	FPGA PYNQ-Z1/Z2 avec architecture reconfigurable.
Performance	Réduction de la latence d'exécution de 76.6% à 88.8%.	Efficacité énergétique mesurée à 43.9 GOPs/W.
Consommation énergétique	Économie d'énergie jusqu'à 85.1%.	Réduction de 16% de la consommation dynamique totale.
Flexibilité	Optimisation spécifique aux CNN, mais peu adaptable à d'autres modèles.	Adaptabilité pour divers modèles CNN comme ResNet et YOLO.
Complexité de développement	Relativement simple grâce à l'utilisation d'un jeu d'instructions dédié.	Développement complexe avec des optimisations au niveau RTL.

2.2.6 Optimisation des Convolutions

La méthode RISC-V exploite des instructions spécifiques (*CONV23*) et l'algorithme de Winograd pour réduire les cycles nécessaires. En revanche, l'approche FPGA optimise les convolutions via une gestion des unités *MAC* et un pipeline matériel.

2.2.7 Gestion Mémoire et Latence

— RI5CY-Accel :Réduction des latences mémoire plus d'images batch, et donc réalise plus d'accès mémoire en mémoire proche, limitant les accès mémoire au reste en mémoire DRAM (HBM2). De cette manière cet accès mémoire en est réduit la taille de bande passante mémoire nécessaire.

— FPGA : Fusionne les étapes de convolution ou bien d'opération de max-pooling limitant le nombre d'accès mémoire effectuées.

2.2.8 Performance et Consommation Énergétique

— RI5CY-Accel : Réduction de la latence d'exécution de 76.6% à 88.8% pour des modèles comme LeNet et ResNet18. Économie d'énergie atteignant 85.1%.
— FPGA : Efficacité énergétique mesurée à 43.9 GOPs/W avec une réduction significative des cycles par convolution.

2.2.9 Avantages et Limites

Avantages

— RI5CY-Accel : Spécialisation légère pour des applications en réseau convolutionnel.
— FPGA : Flexibilité et adaptabilité pour divers des réseaux de convolution.

Limites

— RI5CY-Accel : Moins flexible, nécessite un ensemble d'instruction spécifique.
— FPGA : Plus compliqué de développer matériellement (optimisation RTL).

2.2.10 Conclusion

La combinaison RISC-V est idéale pour des applications très spécifiques qui exigent des performances extrêmes sans compromis sur l'énergie, par contre le kit FPGA vous donne l'utilité et la modularité requises pour faire tourner toute une série d'applications de CNN à performances moyennes et un consommation d'énergie raisonnablement basse.

2.3 Comparaison avec A4

Ce guide détaille une architecture et des IPs pour encoder des réseaux de convolution sur FPGA avec un double but : être simple à développer, être réutilisable dans différentes applications. En outre, le code généré doit avoir des hauteurs de performance tout en pré- sentant un faible temps de latence et occupation minimum des ressource de la carte. La technique est validée par la réalisation plus précisément de la première implémentation de la technique LeNet.

2.3.1 Structure de la Méthodologie

La méthode repose sur une architecture pipeline spécifique composée de trois types de blocs principaux :

— **Blocs de traitement (PB)** : ces blocs exécutent des opérations spécifiques à une ou plusieurs couches CNN.
— **Blocs de flux de données (DFB)** : ils organisent les données de manière à éviter les goulets d'étranglement
— **Blocs d'organisation de la mémoire (MOB)** :leur rôle est de gérer le transfert des données depuis la mémoire externe et la mémoire interne.

Tout bloc de ces trois grandes catégories est connecté aux autres en fonction des attributs de ses interfaces (en anglais, Streaming Interfaces ou SIFs). L'approche systématique suivie pour définir les connexions, équilibrer les temps de propagation dans ces connexions, et insérer là où elle est nécessaire des mémoires tampons de données, est l'un des points communs des trois catégories de blocs.

Points Clés

— Maximisation de la capacité de traitement en DSP (pour les opérations de multiplication accumulateur(MAC)) par rapport à la surface des puces.
— Réduction de la latence et optimisation du coût de circuit en utilisant une configuration adaptative qui exploite la redondance dans le modèle.
— Support de diversité d'architectures CNN tout en restant extensible vers les architectures futures.

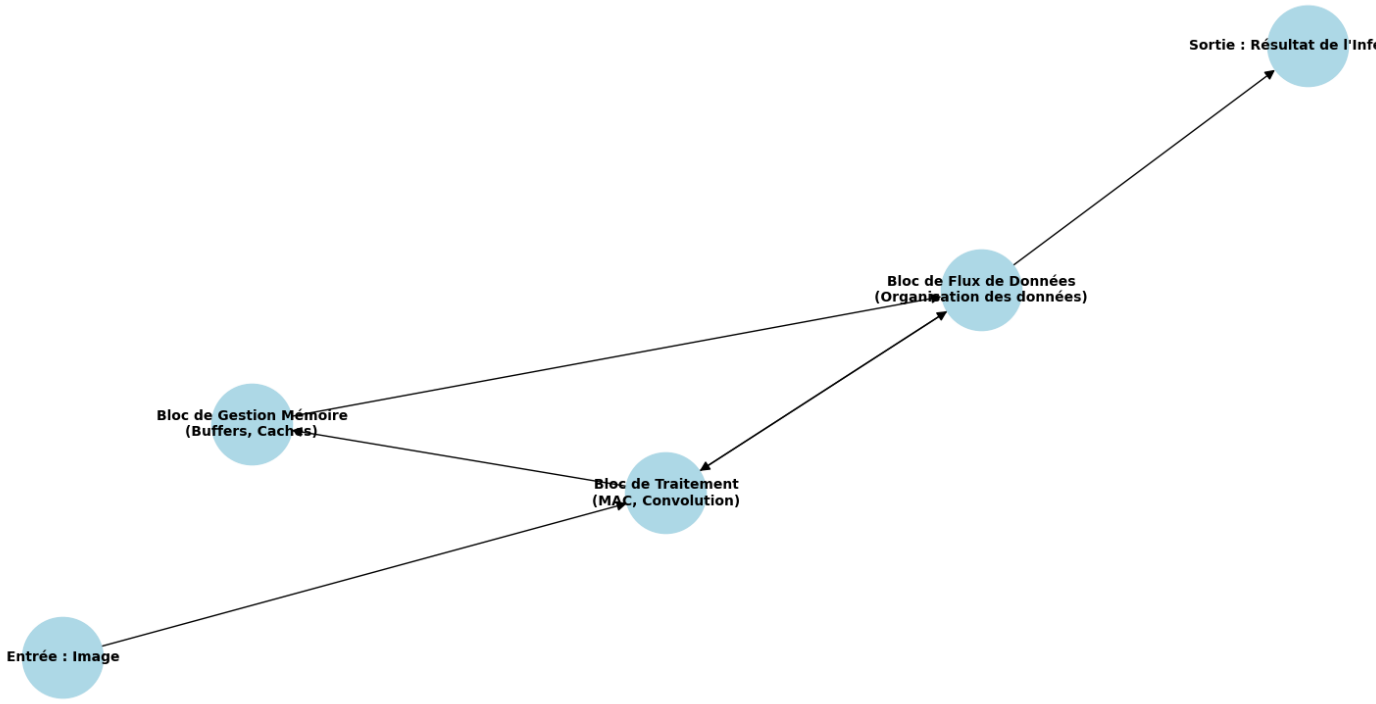

FIGURE 6 – Schématisation de la méthode

2.3.2 Architecture et Conception

Deux approches ont été retenues au niveau micro-architecture. La première est basée sur une architecture modulaire avec des blocs de traitement (PB), des blocs de flux de données (DFB) et des blocs d'organisation de mémoire (MOB). La modélisation autour de ces différents éléments permet de structurer efficacement les calculs et de minimiser les goulets d'étranglement dans les flux de données.

La seconde repose sur un processeur RISC-V personnalisé, RI5CY-Accel, intégrant un module d'accélération ASIC. Ce module ajoute sept instructions SIMD qui peuvent être directement utilisées pour les opérations CNN (notamment les convolutions, poolings, activations, etc.). Ces instructions, parmi lesquelles CONV23, sont conçues pour couvrir un très large panel de tâches et usages tout en offrant une grande performance énergétique et de surface. CONV23 est l'instruction proposée pour le calcul des convolutions 3x3 en utilisant un schéma d'algorithme de Winograd réduit, ce schéma permettant de faire des convolutions 3x3 avec un nombre très réduit de multiplications.

2.3.3 Optimisations Mathématiques

Dans la conception FPGA, l'accent est mis sur une disposition personnalisée des interconnexions afin d'équilibrer les délais et de réduire l'utilisation des ressources matérielles. Les blocs de processing DSP (DSP blocks) sont maximisés pour les MAC (assemblies de multiplication accumulateur).

Deuxièmement, dans la conception RISC-V, l'infrastructure classique adopte l'algorithme de Winograd, qui comporte moins de multiplications pour une convolution 3x3. Non seulement le processus de calcul, mais aussi la quantité de données en mémoire prend moins de temps, et si l'on combine cela avec une réutilisation efficace des données (Data Reuse), la latence par unité d'opérations complet est fortement améliorée via des options mathématiques relatives à cet algorithme.

2.3.4 Performances

Les performances des deux approches sont évaluées sur des réseaux comme LeNet, VGG16 et ResNet18.

— Tout d'abord, depuis l'implémentation matérielle de VGG16 via le pipeline ultra optimisé présenté dans la méthode FPGA, la latence est fortement diminuée avec une consommation énergétique mise à part de la génération des images. La réduction de la latence est due au pipeline ultra optimisé pour VGG16. Cependant, la consommation énérgétique n'est pas spécifiquement adressée à part celle consommée par la génération des images. Même si cela est assez appréciable, il est à noter que la consommation de la génération des images n'est jamais prise en compte.

— Ensuite, par l'intermédiaire de l'implémentation matérielle complète de VGG16 sous RISC-V, la latence moyenne est diminuée de 88.8% (c'est par exemple entre 11.1 fois et 140 fois plus rapide) et la consommation énergétique est fortement réduite. Tout cela en travaillant exclusivement sur les calculs de convolution. Pour alléger les mots, prenons un exemple avec le calcul de la couche CONV23 pour laquelle il faut 21 cycles d'horloge et 1 fois sur 140 sur l'implémentation matérielle et non 140.

2.3.5 Flexibilite et Évolutivité

L'approche FPGA est idéale pour les systèmes équipés de matériel d'IA, car vous y trouvrez une grande variété de modèles de CNN que vous pouvez y exécuter.

Contrairement à cela, l'architecture RISC-V est conçue pour être extensible—la plupart des designs ouverts et commerciaux contiennent des instructions personnalisées personnalisables—eea ! être ; tlns possibilités sont infinies ! doi Les possibilités offertes par les instructions personnalisées vous permettent de prolonger la durée de vie des microarchitectures en implémentant de nouveaux modèles CNN sans changement significatif du matériel.

2.3.6 Consommation Énergétique

La différence majeure entre ces deux approches, c'est d'optimiser les performances dans le cas des FPGA, et la consommation d'énergie pour les RISC-V. Dans le deuxième cas, cela se traduit par une réduction drastique des accès mémoire, allant jusqu'à une économie de 87.8% pour des réseaux comme VGG16.

2.3.7 Limitations

Chaque méthode présente des limitations spécifiques :

— L'approche FPGA depends fortement des spécifications matérielles, ce qui limite son applications pour des architectures CNN complexes.

— L'approche RISC-V nécessite des modifications non-négligeable de l'ISA et par conséquent des compilateurs pour prendre en charge les instructions personnalisées. De plus, elle est moins adaptée aux convolutions avec des noyaux de grande taille (Winograd n'est bénéfique que pour les petits noyaux).

2.3.8 Conclusion

Cette approche fournit une base robuste pour le développement de CNNs sur FPGA, facilitant l'incorporation de nouveaux modèles et améliorant la ré-utilisabilité des conceptions. Le but derrière A4 n'est pas explicitement de réduire la consommation énergétique et de fait augmenter l'efficience comme l'on pouvait voir pour les méthodes precedentes, mais plutot de poser une base de réalisation des CNN sous FPGA afin de gagner du temps sur un exercice chronophage si celui ci doit être reconfiguré pour chaque types de réseaux de neurone.

3 Conclusion

Dans cette conclusion, nous mettons en lumière les défis rencontrés, les bénéfices apportés par ces méthodes et proposons des pistes pour l'amélioration de l'efficacité des CNN sur architectures reconfigurables.

3.1 Bilan des Problèmes Rencontrés

Au delà de la relative complexité du sujet, le problème principale lors de la réalisation de cette étude était le manque de connaissance à priori sur les CNNs, ce qui a nécessité dans un premier temps une consolidation des connaissances à ce sujet et plus particulierement les réseaux de neurones utilisé pour des application temps reels (Type YOLO dans A2). D'autre part, j'ai trouvé une certaine difficulté à comparer les méthodes presentes dans les articles les unes par rapport aux autres et plus spécifiquement les résultats de performance de chaque article étant donné que chaque article teste les performances de sa méthode sur un CNN particulier (YOLO pour A2, RESNET20 pour A3 et LeNet pour A4), ce qui fait qu'une comparaison de performance basé sur ces résultats semblait peu pertinente.

3.2 Apports du Travail en Tant qu'Étudiant

Ce travail m'a permis d'acquerir des connaissances qui me seront utiles dans mon parcours en tant qu'étudiant en systèmes embarqués, la possibilité d'ajouter des instructions à un ISA tel que RISC V en A1 ou les méthodes d'optimisation de cycle d'horloge en A3, pour ne citer que. Cette étude m'a aussi permit d'approfondir et de consolider mes connaissances en réseau de neurone à travers des modèles et architecture de réalisation que je n'ai pas croisé auparavant, YOLO pour la vision par exemple.

3.3 Perspectives d'Avenir

Les futurs travaux pourront explorer des méthodes plus poussées de quantification et des techniques de compression pour réduire encore plus la consommation d'énergie. Cela pourra également passer par l'intégra- tion d'algorithmes d'apprentissage plus sophistiqués, tels que les réseaux de neurones profonds (DNN), dans des environnements de faible puissance

3.4 Aspects que Vous Auriez Aimé Explorer

Dans la continuité de ce travail d'étude, j'aurai aimé suite à l'étude de la portabilité des reseaux de neurones convolutionnel sur environnement à restriction materiel et énergetique, étudier la portabilité cette fois des réseaux Transformer et éventuellement les Tiny LLMs et dans quel mesures et sous quelles contraintes cela pourait-il se faire.

Références

[A1] Optimizing CNN Computation Using RISC-V Custom Instruction Sets for Edge Platforms. *Proceedings of the IEEE International Conference on Edge Computing*, 2023.

[A2] A High-Throughput and Power-Efficient FPGA Implementation of YOLO CNN for Object Detection. *IEEE Transactions on Circuits and Systems for Video Technology*, 2022.

[A3] A Reconfigurable CNN-Based Accelerator Design for Fast and Energy-Efficient Object Detection System on Mobile FPGA. *ACM Transactions on Embedded Computing Systems*, 2021.

[A4] Methodology for CNN Implementation in FPGA-Based Embedded Systems. *International Journal of Reconfigurable Computing*, 2020.

[A5] Fused-layer CNN accelerators. *Proc. 49th Annu. IEEE/ACM Int. Symp. Microarchitec- ture, Oct. 2016, Art. no. 22.*

[A6] FINN : A framework for fast, scalable binarized neural network inference. *IEEE Int. Symp. Field-Program. Custom Comput. Mach., Feb. 2017, pp. 65–74.*

——— ***Fin*** ———