Master E3A

Cours A2 : Systèmes Electroniques Embarqués

Projet : Multicores embarqué pour Big Data et Machine Learning

DOU Yuhan
FORCIOLI Quentin
GHAOUI Mohamed Anis
TERRACHER Audrey

Encadré par : **Quelqu'un**

Master 2 : Système Embarqué et Traitement de l'Information

Année : 2019-2020

1 Introduction

2 Méthodologie et approche

L'équipe consistant en 4 personnes, 2 personnes sont confiés la tâche de conception des logicielles à implémenter. i.e. Choisir les algorithmes à étudier, choisir le stratagème d'implémentation et la définition des besoins en terme de mécanisme C/C++, de jeu de données et de tests à valider. Ensuite, Ils passent à une première implémentation qui n'est pas nécessairement optimisée afin de ne pas bloquer les autres parties de l'équipe. Une fois une première implémentation proposée au HLS, l'optimisation de cette implémentation pour ARM commence. Ce processus itératif consiste en l'analyse de l'architecture et contraintes de l'architecture cible comparée à l'architecture des ordinateurs usuels (PC bureau).

La partie HLS consiste à analyser les implémentations C/C++ proposées par la partie logicielle embarquée, d'adapter ces implémentations d'algorithme aux contraintes matérielles sur la carte Zboard. En effet, certains mécanismes logiciels sont simplement impossibles à réaliser en matériel et donc la conception HLS sera une version modifiée afin de créer une IP implémentable et interfaçable dans un premier par ARM puis par μblaze.

3 Conception logicielle embarquée

4 Conception Synthèse Haut Niveau

IP sera encapsulée en elle-même. Ce qui fait que l'implémentation matérielle n'aura aucun contrôle sur ce qui se passe dans l'IP. Afin de pouvoir optimiser l'IP HLS créée, il faut dans un premier temps faire une analyse des dépendances pendant l'exécution de l'algorithme. Cette analyse est faite en 2 temps :

- En premier, automatiquement par Vivado HLS qui possède des outils permettant de détecter des dépendances, d'essayer de les corriger/éliminer ou de suggérer à l'utilisateur d'utiliser les directives proposées.
- En second, l'utilisateur, à l'aide des directives HLS, modifie la synthèse sans modifier le code lui même.

Les directives les plus intéressantes, pour les algorithmes demandant plusieurs itérations et ayant une implémentation séquentielle tel que Kmeans, sont `FLATTEN`, `PIPELINE` et `UNROLL`.

Flatten : Elle consiste à essayer de simplement aplatir la boucle en la rendant en une simple suite d'instructions. Ceci permettra de concevoir un flot de donnée matériel en RTL. Cette directive n'est possible que si toutes les boucles internes peuvent être aplaties et donc qu'il n'y ait pas de dépendances en données.

Pipeline : Applicable aux boucles, fonctions et opérateurs, cette directive divise les opérations en plusieurs étages successifs de conception simple qui à chaque cycle effectuent une opération en 1 cycle. Le pipeline est très efficace pour diminuer la latence des opérateurs mais occupent une plus grande surface sur le FPGA contrairement à une version multi-cycles.

Unroll : Quand un calcul dans une boucle possède des dépendances de données et une latence plus grande que la durée d'une itération, il est conseillé de dérouler la boucle d'un facteur. par exemple au lieu de traiter les données 1 à 1, on peut traiter n à n. Cette directive peut occuper beaucoup plus de surface FPGA. Il faut donc l'utiliser uniquement quand le nombre d'itération est faible.

Une fois un opérateur/fonction optimisé, il faut analyser le nombre d'accès mémoire qu'il effectue en lecture/écriture. Si un port ne fourni qu'un port d'accès, une seule lecture/écriture peut être fait en même temps dans le même cycle (en prenant en compte que la lecture prend deux cycles selon HLS). Il existe plusieurs stratagèmes pour

5 Implémentation matérielle

6 Conclusion