Modélisation et accélération de réseaux de neurones profonds (CNN) en Python/VHDL/C++ et leur vérification et test à l'aide de l'environnement Pynq sur les FPGA Xilinx

S. Jovanović et S. Weber

Institut Jean Lamour (UMR7198) et Pôle CNFM MIGREST, Université de Lorraine, Nancy

Contact email : slavisa.jovanovic@univ-lorraine.fr

Nous présentons un ensemble de travaux pratiques qui seront dispensés au sein du Master EEA - Électronique Embarquée à l'université de Lorraine dans le cadre des modules Modélisation SystemC et Conception VLSI. Ces TP sont destinés à initier les étudiants à la compréhension, modélisation et conception des réseaux de neurones convolutifs dans des langages de description de matériel au niveau RTL (VHDL, le module Conception VLSI) et dans un langage de haut niveau (C++/SystemC, le module Modélisation SystemC). Ils sont organisés autour d'un ensemble d'outils de modélisation et de synthèse de Mentor Graphics (Modelsim, Catapult HLS) et spécifiques aux plateformes FPGA Xilinx et à l'environnement Pynq pour la simulation, test et vérification.

I. Introduction

Pour initier les étudiants de la formation Master EEA - Électronique Embarquée à l'Université de Lorraine à la modélisation de systèmes et circuits en VHDL et SystemC, nous allons mettre en place une série de travaux pratiques autour des suites logicielles de Mentor Graphics (Modelsim et Catapult HLS pour la modélisation et synthèse VHDL et SystemC), et de Xilinx (Vitis HLS et l'environnement Pynq) permettant de simuler, tester, synthétiser et vérifier de manière expérimentale les réseaux de neurones convolutifs (CNN) d'une faible complexité. La modélisation d'un réseau de neurones convolutif au niveau RTL s'inscrit dans deux unités de 30 heures: le module Conception VLSI, où l'accent est mis sur la conception de circuits numériques de calcul au niveau RTL (opérateurs arithmétiques; synthèse logique, placement et routage); et sur le module Modélisation SystemC, où l'objectif principal est de familiariser les étudiants avec les concepts de modélisation et de synthèse système de haut niveau (module, interface, canaux de communication).

Cette unité s'adresse aux étudiants de niveau Master 2 ayant déjà un bagage de connaissances solide en conception numérique (VHDL, FPGA - au moins 30h) et en conception de circuits microélectroniques de base (60h) avec la suite logicielle Cadence (30h). De plus, les étudiants ont des connaissances en programmation orientée objet (Java, Python), en programmation structurelle sur microcontrôleur (langage C) et en programmation C sous Linux (compilation croisée) leur permettant d'assimiler les concepts liés à la modélisation de haut niveau de manière plus aisée. Par conséquent, on peut considérer que les étudiants ont une expérience suffisante et non négligeable pour mener à bien le projet proposé.

Article publié par EDP Sciences et disponible sur le site https://www.j3ea.org ou https://doi.org/10.1051/j3ea/20220028

II Démarche pédagogique

La modélisation, simulation et vérification d'un réseau de neurones convolutif de base passe par les étapes suivantes :

1. Introduction à l'environnement Pynq et sa mise en place sur une plateforme Zynq (Zedboard, MicroZed, ZCU102, etc),
2. Introduction aux réseaux de neurones profonds et leur modélisation en Python sur un exemple de base avec les étapes de simulation et test vérifiées sur une plateforme FPGA équipée de l'environnement Pynq,
3. Modélisation d'une couche de convolution dans le langage SystemC/VHDL et sa synthèse (RTL, haut niveau) avec les outils de synthèse RTL (Vitis) et haut niveau (Catapult-HLS, Vivado HLS),
4. Création d'une IP FPGA (et de son driver) correspondant à la couche de convolution précédemment modélisée et son test et vérification dans l'environnement Pynq (comparaison des temps d'exécution entre la solution initiale en Python et en SystemC).

Fig.1. Les plateformes de développement qui seront utilisées pendant ces TP: a) la plateforme ZCU 102 et b) MicroZed.

Introduction à l'environnement Pynq

Dans la première étape programmée sur 2h, l'étudiant est amené à se familiariser avec les mécanismes principaux de simulation et de vérification en utilisant l'environnement Pynq, disponible sur les FPGA MPSoC de Xilinx. Dans cette étape, une introduction brève de l'environnement Pynq sera présentée aux étudiants en leur expliquant également les étapes de création d'un environnement Pynq sur une plateforme FPGA MPSoC de Xilinx. L'environnement Pynq est une couche logicielle accessible à distance (via la connexion ethernet) via un navigateur

internet simple, tournant sous l'environnement Linux installé sur la partie processeur des FPGA MPSoC de Xilinx. Il permet en particulier d'interagir avec le côté FPGA sur lequel des IP conçues en VHDL, SystemC ou autres, peuvent être déployées. Ainsi, on peut simuler, tester et commander les étapes d'exécution d'une IP déployée sur la partie FPGA depuis l'environnement Pynq. Une particularité de l'environnement Pynq est l'utilisation du langage Python et de l'approche Jupyter pour cette communication entre un navigateur et la plateforme (voir la figure 2). Dans cette première partie, l'étudiant testera sur l'environnement Pynq quelques fonctions de base déjà fournies avec l'environnement Pynq.

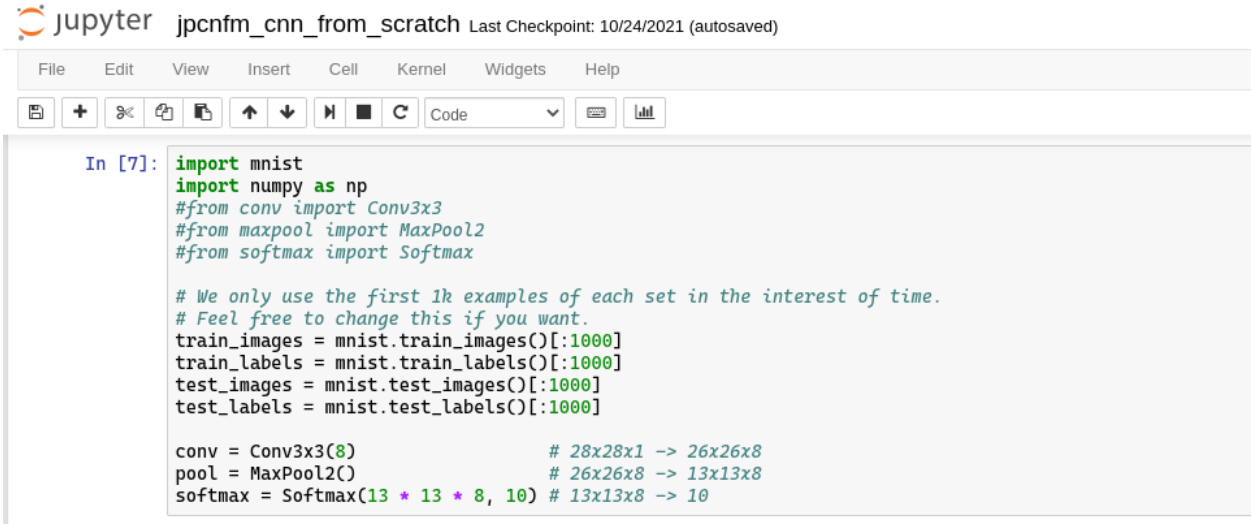

Fig.2. Un extrait de l'environnement Pynq: un script Jupyter notebook décrit en Python.

Introduction aux réseaux de neurones profonds (CNN) et leur modélisation en Python

Dans la deuxième partie de ces projets, les étudiants seront amenés à simuler et entraîner le comportement d'un réseau de neurones convolutif de base. Il s'agit en particulier d'une architecture très simple d'un réseau CNN comportant une couche de convolution, une couche de *pooling* et une couche de *softmax*. L'objectif de cette partie est d'expliquer aux étudiants les couches principales pouvant se trouver dans un réseau de neurones convolutif. Les couches précédentes, déjà modélisées en Python et fournies aux étudiants, seront en premier lieu testées sur l'environnement Pynq sans utiliser la partie FPGA. La puissance de calcul de la partie processeur des plateformes Microzed étant assez modeste, la taille des images à traiter sera limitée à 28x28. En particulier, la base MNIST comprenant une base de 60 000 images de chiffres manuscrits sera utilisée pour toutes les étapes d'entraînement et de test dans le cadre de ces TP.

Dans cette partie, les étudiants doivent analyser le code Python fourni décrivant le réseau CNN décrit précédemment. Il s'agit d'un code simple, ne comportant aucune bibliothèque avancée de modélisation de réseau de neurones profonds (de type keras, tensorflow ou autre). Le code fourni permet également d'effectuer l'entraînement du réseau CNN proposé sur la partie processeur. Un extrait de la phase d'entraînement de ce réseau est présenté figure 3.

```
MNIST CNN initialized!
--- Epoch 1 ---
[Step 100] Past 100 steps: Average Loss 2.227 | Accuracy: 21%
[Step 200] Past 100 steps: Average Loss 2.006 | Accuracy: 40%
[Step 300] Past 100 steps: Average Loss 1.657 | Accuracy: 50%
[Step 400] Past 100 steps: Average Loss 1.377 | Accuracy: 53%
[Step 500] Past 100 steps: Average Loss 0.835 | Accuracy: 75%
[Step 600] Past 100 steps: Average Loss 0.894 | Accuracy: 70%
[Step 700] Past 100 steps: Average Loss 0.704 | Accuracy: 80%
[Step 800] Past 100 steps: Average Loss 0.840 | Accuracy: 76%
[Step 900] Past 100 steps: Average Loss 0.651 | Accuracy: 80%
[Step 1000] Past 100 steps: Average Loss 0.556 | Accuracy: 78%
--- Epoch 2 ---
[Step 100] Past 100 steps: Average Loss 0.401 | Accuracy: 91%
[Step 200] Past 100 steps: Average Loss 0.510 | Accuracy: 85%
[Step 300] Past 100 steps: Average Loss 0.492 | Accuracy: 83%
[Step 400] Past 100 steps: Average Loss 0.558 | Accuracy: 85%
[Step 500] Past 100 steps: Average Loss 0.672 | Accuracy: 76%
[Step 600] Past 100 steps: Average Loss 0.561 | Accuracy: 85%
[Step 700] Past 100 steps: Average Loss 0.736 | Accuracy: 79%
[Step 800] Past 100 steps: Average Loss 0.450 | Accuracy: 85%
[Step 900] Past 100 steps: Average Loss 0.599 | Accuracy: 78%
[Step 1000] Past 100 steps: Average Loss 0.433 | Accuracy: 85%
--- Epoch 3 ---
[Step 100] Past 100 steps: Average Loss 0.405 | Accuracy: 86%
[Step 200] Past 100 steps: Average Loss 0.312 | Accuracy: 89%
[Step 300] Past 100 steps: Average Loss 0.404 | Accuracy: 88%
[Step 400] Past 100 steps: Average Loss 0.446 | Accuracy: 89%
[Step 500] Past 100 steps: Average Loss 0.575 | Accuracy: 82%
[Step 600] Past 100 steps: Average Loss 0.573 | Accuracy: 83%
[Step 700] Past 100 steps: Average Loss 0.393 | Accuracy: 90%
[Step 800] Past 100 steps: Average Loss 0.374 | Accuracy: 91%
[Step 900] Past 100 steps: Average Loss 0.365 | Accuracy: 89%
[Step 1000] Past 100 steps: Average Loss 0.452 | Accuracy: 91%

--- Testing the CNN ---
Test Loss: 0.499974448992
Test Accuracy: 0.84
```

Fig.3. Extrait d'entraînement d'un réseau CNN simple sur l'environnement Pynq déployé sur une plateforme MicroZed.

Modélisation et synthèse d'une couche de convolution dans le langage SystemC/VHDL

Dans la troisième partie de ces TP, les étudiants vont se focaliser en particulier sur la couche de convolution utilisée dans le réseau CNN décrit précédemment. Il s'agit d'une couche de convolution qui devra être modélisée de deux façons différentes : en langage de description de matériel VHDL dans le cadre du module Conception VLSI, où un accent particulier sera mis sur le choix des opérateurs arithmétiques à utiliser ; et en langage de modélisation système SystemC dans le cadre du module Modélisation SystemC. Dans les deux cas, les étudiants vont modéliser la couche de convolution en plusieurs étapes. En premier, une modélisation de très haut niveau dans les deux langages de modélisation (VHDL, SystemC) sera effectuée. Cette modélisation permettra de valider le fonctionnement attendu d'une couche de convolution. Ensuite, les étapes de raffinement vont se succéder au niveau de chaque modélisation, permettant ainsi de choisir les opérateurs arithmétiques à utiliser (par exemple, le multiplieur de Booth, l'additionneur CLA, etc.), la taille des opérandes en jouant sur le nombre de bits, l'architecture globale à utiliser (séquentielle, semi parallèle ou complètement parallèle), etc. Dans ces étapes, des fichiers de test dans les deux langages de modélisation seront fournis aux étudiants, leur permettant de tester de manière facile leurs modèles de convolution 2D. De plus, ces choix architecturaux dépendront

également des performances exigées en termes de fréquence de fonctionnement, de surface, de consommation, etc. Dans le cadre de la conception RTL, les étudiants effectueront une synthèse finale sur les outils Xilinx (Vivado, Vitis) tandis que pour la conception SystemC, la synthèse de haut niveau sera réalisée avec l'outil Catapult HLS de Mentor Graphics. Dans les deux cas, les scripts automatisés seront fournis aux étudiants leur permettant d'arriver plus rapidement aux résultats escomptés, sans passer trop de temps sur les outils dont l'apprentissage peut s'avérer chronophage.

La troisième partie de ces TP est la partie nécessitant un volume horaire entre 8 et 12h, en fonction du langage de modélisation utilisé. Il s'agit de la partie où les étudiants pourront mettre en pratique tous les aspects de conception abordés dans le cadre des deux cours. Il est à noter également que les travaux pratiques des deux modules sont programmés de manière générale sur plusieurs semaines en parallèle (courant le semestre S9) et permettent ainsi de traiter le même problème (la conception d'un même design) sous différents angles.

Création d'une IP FPGA de convolution et son test dans l'environnement Pynq

Dans la dernière étape de ce projet, les étudiants seront amenés à valider expérimentalement la couche de convolution modélisée dans la partie précédente. Dans les deux cas, les couches de convolution modélisées seront testées sur l'environnement Pynq précédemment décrit. Le test de la couche de convolution sur l'environnement Pynq sera effectué de manière indépendante du réseau de neurone CNN précédemment décrit. Pour pouvoir tester un module décrit en VHDL ou SystemC dans l'environnement Pynq, il faudra passer par l'étape de création d'un driver, permettant d'insérer et voir l'IP créée dans l'environnement Pynq. Un exemple de driver pour une simple opération d'addition est présenté en figure 4.

```
from pynq import DefaultIP

class AddDriver(DefaultIP):
def __init__(self, description):
super().__init__(description=description)

bindto = ['xilinx.com:hls:add:1.0']

def add(self, a, b):
self.write(0x10, a)
self.write(0x18, b)
return self.read(0x20)
```

Fig.4. Un exemple de création d'un driver pour un simple additionneur.

Ainsi, les IP VHDL et SystemC créées précédemment pourront être utilisées dans l'environnement Pynq comme des fonctions logicielles normales. De plus, une comparaison au niveau des résultats de convolution dans un premier temps et de la classification sur les 2 IP et le réseau de neurones initial en Python dans un second temps pourra être effectuée. De même, une comparaison au niveau du temps d'exécution des deux approches (logicielle et matérielle) pourra être effectuée. La figure 5 présente les résultats de validation d'une IP de convolution décrite en SystemC avec l'environnement Pynq.

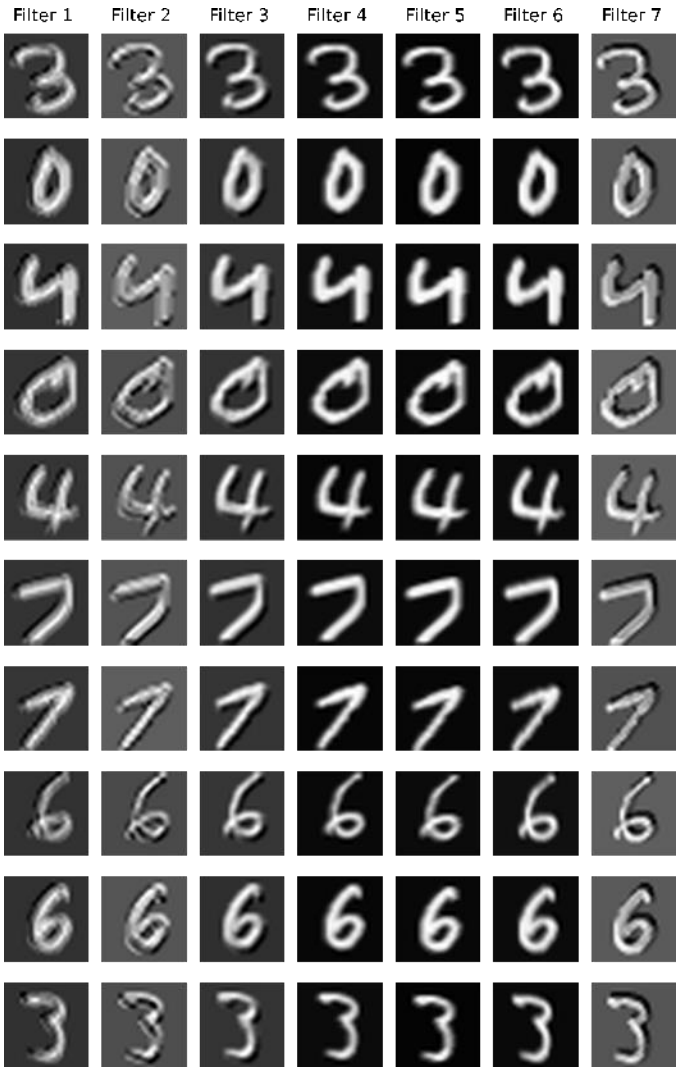

Fig.5. Résultats de convolution obtenus sur les images issues de la base MNIST.

III Conclusion

Nous avons présenté un ensemble de travaux pratiques qui seront dispensés au sein du Master EEA - Électronique Embarquée à l'université de Lorraine dans le cadre des modules Modélisation SystemC et Conception VLSI. L'objectif principal de ces TP est d'initier les étudiants à la modélisation et conception des réseaux de neurones convolutifs dans des langages de description de matériel au niveau RTL (VHDL) et de haut niveau (C++/SystemC). Ils seront organisés autour d'un ensemble d'outils de modélisation et de synthèse de Mentor Graphics (Modelsim, Catapult HLS) et spécifiques aux plateformes FPGA Xilinx et à l'environnement Pynq pour la simulation, test et vérification. La mise en place de ces TP est prévue pour la rentrée 2022/23.

Remerciements

Nous remercions, le GIP CNFM (1) coordonnateur du projet IDEFI-FINMINA (2), ainsi que le pôle MIGREST pour la participation financière à l'acquisition du matériel nécessaire à la réalisation de cette série de travaux pratiques.

Références

1. GIP-CNFM : Groupement d'Intérêt Public - Coordination Nationale pour la formation en Microélectronique et en nanotechnologies. Website : http://wwww.cnfm.fr (Accès 2021).
2. IDEFI-FINMINA : Initiative d'Excellence - Formation Innovante en MIcroélectronique et Nanotechnologies, ANR-11-IDFI-0017. Website : http://www.cnfm.fr/VersionFrancaise/actualites/FINMINA.htm (Accès 2021).