

# Laboratoire 9: Convolution

Département: **TIC**

Unité d'enseignement: **SCF**

Auteur(s):

- **CECCHET Costantino**

Professeur:

- **DASSATTI Alberto**
- **YANN Thomas**

Assistant:

- **JACCARD Anthony I.**

Date:

- **22/05/2024**

*[Page de mise en page, laissée vide par intention]*

## Introduction

Ce laboratoire va nous permettre de développer de A à Z un système complet pour calculer une convolution sur FPGA.

## Implémentation

Notre objectif est de réaliser une convolution sur une image en utilisant un noyau de convolution de taille 3x3, en utilisant un FPGA.

Pour ce faire, nous allons utiliser la DE1-SoC.

Notre première étape est de réaliser une IP qui va permettre de réaliser la convolution, nous sommes partis sur une IP qui utilise des FIFO pour stocker les valeurs de l'image et de registres pour stocker les valeurs du noyau.

Dès que les données sont dans la FIFO d'entrée, on peut commencer les calculs, on va multiplier les valeurs de l'image par les valeurs du noyau et les additionner pour obtenir la valeur de la convolution. Ces opérations se font ligne par ligne de 3 pixels.

Nous utilisons des FIFO de 32 bit pour stocker 3 pixels de l'image à la fois, et des registres de 32 bit pour stocker les valeurs du noyau. Ceci simplifie le calcul de la convolution. Chaque ligne calculée est stockée dans une FIFO de sortie et pour finir cette FIFO est lue 3 fois d'affilée pour obtenir les 3 valeurs de la convolution on les additionne et on les stocke dans une FIFO de sortie. Cette dernière FIFO est donc lue par notre utilisateur pour obtenir la valeur de la convolution.

## Conception

voici la liste des registres et des FIFO utilisées dans notre IP:

offset	nom	taille	description
0x00	Constant	32 bit	valeur constante 0xBADB100D
0x04	kernel[0-2]	32 bit	valeurs du noyau de convolution
0x08	kernel[3-5]	32 bit	valeurs du noyau de convolution
0x0C	kernel[6-8]	32 bit	valeurs du noyau de convolution
0x10	img[0-2]	32 bit	valeurs de l'image par 3 pixels
0x1C	return value	32 bit	valeur de la convolution
0x20	control register	16 bit	registre de contrôle pour la lecture ou écriture des FIFO

## déroulement de la convolution

Comment ce déroule la convolution:

1. Avant de commencer une convolution, l'utilisateur doit charger les valeurs du noyau dans les registres correspondants.
2. Une fois le noyau chargé, l'utilisateur peut commencer à charger les valeurs de l'image dans la FIFO d'entrée ces données doivent être chargées par 3 pixels, c-à-dire la première ligne du segment 3x3 que l'on va convoluer, puis la deuxième ligne et enfin la troisième ligne.
3. Une fois 3 pixels chargés, le calcul de la convolution commence, on multiplie les valeurs de l'image par les valeurs du noyau et on additionne le tout pour obtenir la valeur de la convolution. Cette dernière est disponible que lorsque les 3 lignes ont été calculées. Si les données ne sont pas encore prêtes, l'utilisateur peut attendre ou charger de nouvelles données tant que la FIFO d'entrée n'est pas pleine. Avant chaque lecture et écriture, l'utilisateur doit vérifier que la FIFO est prête à être lue ou écrite.

Des registres de contrôle sont utilisés pour vérifier si les FIFO sont prêtes à être lues ou écrites. En lisant à l'offset 0x20, si la FIFO est prête à être lue, le deuxième bit est à 1, si la FIFO est prête à être écrite, le premier bit est à 1. Les 4 bits suivants sont utilisés pour voir le status des FIFO.

bit	description
0	fifo d'entrée pas pleine
1	fifo de sortie pas vide
2	fifo de processing pleine
3	fifo de processing vide
4	fifo de sortie pleine
5	fifo d'entrée vide

Ces bits sont utiles si l'on souhaite savoir ou en est notre convolution.

## Pseudo code

L'utilisateur doit suivre les étapes suivantes pour réaliser une convolution:

1. Charger les valeurs du noyau dans les registres correspondants.
2. Charger les valeurs de l'image dans la FIFO d'entrée.
3. Attendre que la FIFO de sortie ne soit pas vide.
4. Lire la valeur de la convolution.
5. Répéter les étapes 2 à 4 jusqu'à ce que toutes les valeurs de l'image aient été lues.

## VHDL

Nous utilisons des FIFO de Intel pour stocker les valeurs de l'image et de la convolution, voici un exemple de la déclaration d'une FIFO, pour une FIFO de 256x32: 

```
vhdl - definition of the scfifo 256x32
GENERIC MAP (
  add_ram_output_register => "OFF", intended_device_family => "Cyclone V",
  lpm_numwords => 256, lpm_showahead => "OFF", lpm_type => "scfifo",
  lpm_width => 32, lpm_widthu => 8, overflow_checking => "ON", underflow_checking => "ON",
  use_eab => "ON" )
```

Cette fifo est importée dans notre IP et utilisée pour stocker les valeurs de l'image et de la convolution.

les images sont stockées de la manière suivante:

bits 31-24	bits 23-16	bits 15-8	bits 7-0
pixel 0	pixel 1	pixel 2	-
pixel 3	pixel 4	pixel 5	-
pixel 6	pixel 7	pixel 8	-

Pour les valeurs du noyau, elles sont stockées de la manière suivante:

bits 31-24	bits 23-16	bits 15-8	bits 7-0
valeur 0	valeur 1	valeur 2	-
valeur 3	valeur 4	valeur 5	-
valeur 6	valeur 7	valeur 8	-

Ceci simplifie les calculs de la convolution en séparant les valeurs de l'image et du noyau par ligne.

## Driver

Un driver est nécessaire pour communiquer avec notre IP, ce driver est écrit en C et permet de charger les valeurs du noyau et de l'image, de lire les valeurs de la convolution et de vérifier le status des FIFO.

Le driver est basé sur celui d'un labo précédent, il a été modifié pour correspondre à notre IP.

Il utilise les read et write pour communiquer avec notre IP et ioctl pour modifier les valeurs aux quelles on souhaite accéder.

Les offsets sont donc les suivants: pour lire:

offset	nom	taille	description
0x00	kernel[0-2]	32 bit	valeurs du noyau de convolution
0x01	kernel[3-5]	32 bit	valeurs du noyau de convolution
0x02	kernel[6-8]	32 bit	valeurs du noyau de convolution
0x03	return value	32 bit	valeur de la convolution
0x04	control register	16 bit	registre de contrôle pour la lecture ou écriture des FIFO
0x05	Constant	32 bit	valeur constante 0xBADB100D

Pour écrire:

offset	nom	taille	description
0x00	kernel[0-2]	32 bit	valeurs du noyau de convolution
0x01	kernel[3-5]	32 bit	valeurs du noyau de convolution
0x02	kernel[6-8]	32 bit	valeurs du noyau de convolution
0x03	img[0-2]	32 bit	valeurs de l'image par 3 pixels

Ceci facilitera l'utilisation de notre IP par le biais de notre driver.

Lors de l'écriture ou de la lecture si les offsets ne sont pas corrects, le driver renverra une erreur.

Le driver sera compilé avec la commande suivante depuis le PC hôte:

```
make
```

Puis il sera copié sur la carte SD de la DE1-SoC pour être utilisé.

```
'''bash cp convolution.ko '''
```

Puis une fois sur la carte, il sera chargé avec la commande suivante:

```
insmod convolution.ko
```

## Programme utilisateur

Le programme utilisateur à tout d'abord été écrit en utilisant le fichier /dev/mem pour communiquer avec notre IP, puis il a été modifié pour utiliser le driver que nous avons écrit.

Ceci dans un but de simplification de l'utilisation de notre IP et de la phase de debug sans driver initialement.

## lancer le projet

Pour lancer le programme il faut suivre les étapes suivantes depuis la DE1-SoC:

```
'''bash ./convolution '''
```

Le kernel de convolution est déjà chargé dans le programme, il est possible de le modifier en modifiant le code source du programme. Les make files sont fournis pour le driver et le programme utilisateur.

Les fichiers doivent être copiés sur la carte SD de la DE1-SoC pour être utilisés.

## Conclusion

Ce laboratoire nous a permis de réaliser une IP de convolution sur FPGA, nous avons pu voir comment communiquer avec cette IP en utilisant un driver et un programme utilisateur.

Ceci nous a permis de décentraliser une operation sur FPGA et de la rendre accessible à un utilisateur. Nous avons pu voir comment utiliser des FIFO pour stocker des données et comment les lire et les écrire.