HAUTE ÉCOLE D'INGÉNIERIE ET DE GESTION DU CANTON DE VAUD

Prof. Yann Thoma

Architecture hardware

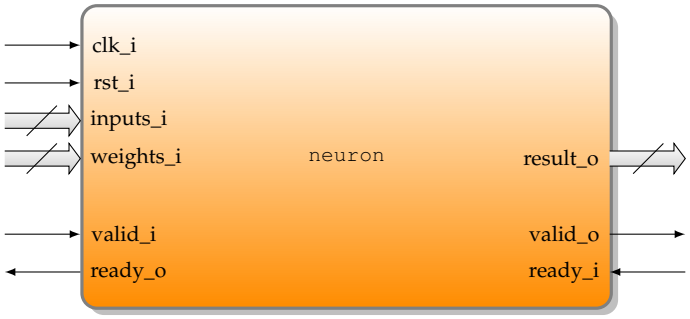
semestre printemps 2023 - 2024

Objectifs pédagogiques

Ce laboratoire a pour but de réaliser un neurone artificiel combinatoire, séquentiel, et pipeliné.

Cahier des charges

Dans l'idée de réaliser un réseau de neurones sur FPGA, il vous est demandé de réaliser un neurone, et ce selon différentes architectures.



Le nombre d'entrées ainsi que la taille des données est définie dans un paquetage, de même que l'emplacement de la virgule. En effet, les calculs doivent se faire en virgule fixe, et l'emplacement de la virgule indique avant quel bit celle-ci se trouve.
Il s'agit ici uniquement du calcul de la somme pondérée des entrées. ⚠ , il se pourrait que votre chef de projet vienne avec une spécification supplémentaire demandant d'ajouter la fonction de seuil dans le neurone.

Entrées/sorties

Les entrées/sorties de l'entité sont les suivantes :

```
entity neuron is
generic(
    COMMA_POS : integer := 0
);
port (
    clk_i     : in  std_logic;
    rst_i     : in  std_logic;
    inputs_i  : in  neuron_input_t;
    weights_i : in  neuron_weights_t;
    valid_i   : in  std_logic_vector;
    ready_o   : out std_logic_vector;
    result_o  : out std_logic_vector;
    ready_i   : in  std_logic;
    valid_o   : out std_logic
);

    -- A nice VHDL possibility, to have constants available in each architecture
    constant NBINPUTS : integer := inputs_i'length;
    constant DATASIZE : integer := result_o'length;
end neuron;
```

Le neurone prend en entrée un certain nombre de valeurs, et leurs poids associés. Pour chaque "entrée", il y un signal de validité indiquant que la donnée est prête, et un signal en sortie pour

indiquer que la donnée est consommée. Elle l'est donc si les deux signaux `valid_i(X)` et `ready_o(X)` sont actifs.
Le résultat est considéré valide lorsque `valid_o` est actif, et est consommé lorsque `valid_o` et `ready_i` sont actifs simultanément.
Nous pouvons noter ici l'utilisation de vecteurs non contraints, et de tableaux non contraints de vecteurs non contraints.

1 Banc de test

Un banc de test vous est fourni. Il est exploitable pour les trois architectures. Utilisez les scripts *sim.do* afin d'avoir toujours la possiblité de simuler rapidement avec l'un ou l'autre des designs.

Combinatoire

La première implémentation doit être purement combinatoire. Les entrées `rst_i` et `clk_i` ne sont donc pas utilisées. Les signaux `valid_i` et `ready_i` doivent être transmis directement aux sorties `valid_o` et `ready_o`.

Séquentiel

La version séquentielle doit traiter les données en essayant de minimiser les ressources nécessaires. Il s'agira donc clairement de n'avoir qu'un seul multiplicateur.

Pipeliné

La version pipelinée doit séparer les multiplications des additions dans deux étages de pipeline. Si vous le désirez vous pouvez aussi découper les additions sur plusieurs étages.

Synthèse

Nous n'allons pas intégrer les systèmes sur carte, mais sommes néanmoins intéressés par la synthèse. Pensez évidemment à faire des synthèses durant le développement afin de ne pas simuler du code non synthétisable. Aussi, nous serons intéressés à savoir combien de ressources sont nécessaires pour vos trois implémentations, pour la cible correspondant à la DE1SoC. Pour ces synthèses, une entité `neuron_synth` vous est fournie, de même qu'un paquetage nécessaire. Il est fait de manière à réduire le nombre d'entrées/sorties pour éviter des problèmes de placement-routage, et ajoute des registres en entrée/sortie pour faciliter la mesure de fréquence max.
Au final, prenez un peu de recul sur ce travail et analysez ce qui pourrait en être fait dans le cadre d'un réseau de neurones complet. Dans vos réflexions, proposez également des idées architecturales si les poids devaient être stockés en interne du neurone.

Travail à rendre

Vous devez rendre toutes les sources, ainsi qu'un petit rapport.
Votre rapport doit contenir les informations suivantes :

1. Présentation des choix que vous avez fait pour les trois designs
2. Schémas des trois implémentations (il est tout à fait possible de prendre une photo d'un dessin)
3. Résultat en synthèse des trois implémentations (logique + nombre de registres)

4. Une analyse de ces synthèses, analysant les pour/contre des différents designs.
5. Une/des propositions architecturales si les poids devaient être stockés dans le neurone.