# 1 Réalisation

Tous les codes sont disponibles sur la page GitHub du projet [1]

## 1.1 Organisation

Des fichiers "wrapper" sont utilisés pour gérer la connexion entre le bloc de calcul (loop ou pipeline) et l'interface de la BRAM. La structure du projet est décrite dans la figure 1

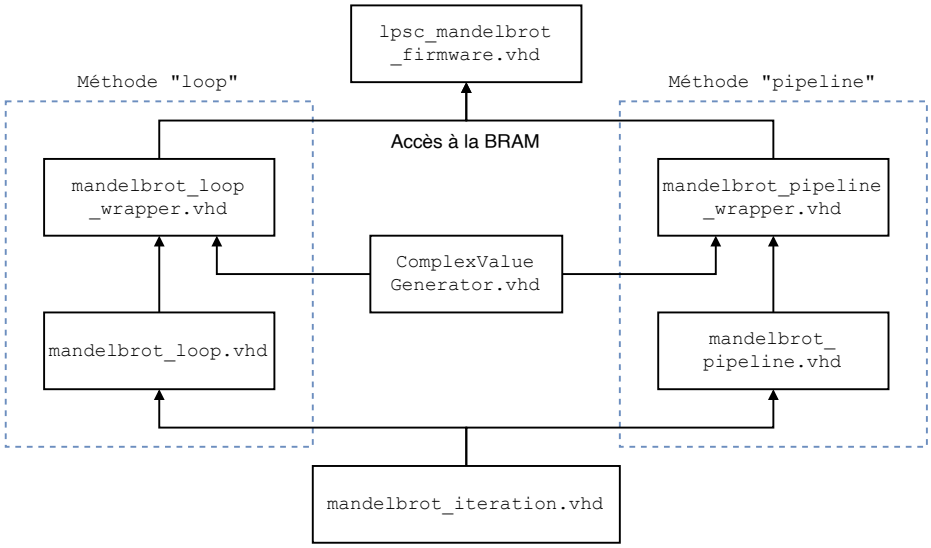


FIGURE 1 – Organisation des fichiers

Les fichiers `mandelbrot_loop.vhd` et `mandelbrot_pipeline.vhd` se chargent de calculer une itération complète pour un pixel. Dans le cas du loop un pixel est réalisé à la fois, dans le cas du pipeline, les pixels sont envoyés à chaque coups d'horloge et les résultats sont présent après un délai de 100 coups d'horloge (le nombre maximal d'itérations).

## 1.2 Itération

L'itérateur (`mandebrot_iteration.vhd`) est basé sur un process synchrone qui réalise tous les calculs en un coup d'horloge. Il est primordial d'avoir des registres sur toutes les sorties afin d'utiliser les itérateurs dans le pipeline ainsi que dans le loop.

## 1.3 Loop

Pour boucler, il suffit simplement de relier les sorties de l'oblitérateur sur les entrées (car elles sont équipées de registres). Lorsqu'un signal `start` est actif, les entrées sont remplacées par des valeurs initiales.
Il existe aussi un process synchrone pour gérer les sorties du loop(`done` et `iterations`). Le process synchrone désactive la sortie `done` lorsque le calcul est lancé et la ré-active lorsque le calcul est terminé, il va aussi mettre à jour la valeur de `iterations`.

1. `https://github.com/SebastienDeriaz/LPSC_TP3`

## 1.4 Loop wrapper

Le loop wrapper va utiliser le fichier `ComplexValueGenerator.vhd` pour générer les nombres complexes et les transmettre au loop. Lorsqu'un pixel est calculé, le prochain est envoyé. Le loop wrapper est basé sur une machine d'état doit voici le graph

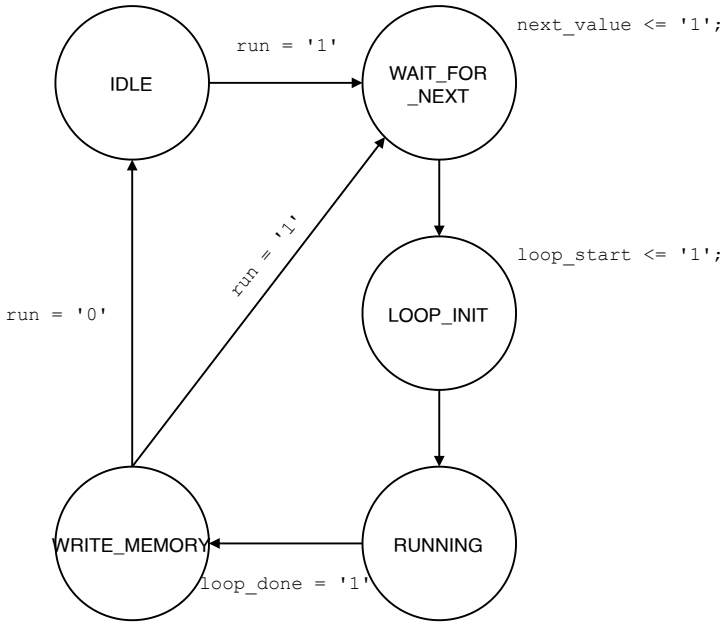


FIGURE 2 – Machine d'état du loop wrapper

Le signal `next_value` demande un nouveau nombre au générateur de nombre complexe (0 dans les autres états). Le signal `loop_start` démarre le calcul du point (0 dans les autres cas).
La sortie du loop wrapper est une interface mémoire pour écrire dans la BRAM du `lpsc_mandelbrot_firmware.vhd`

## 1.5 Pipeline

La base du pipeline est une boucle `for generate` qui instancie 100 itérateurs. Le premier itérateur prend comme entrée les entrées du pipeline (point à calculer), le dernier itérateur fourni les sorties du pipeline (nombre d'itérations) et les itérateurs "centraux" sont reliés ensemble à l'aide de signaux. Afin de déterminer si la sortie du pipeline est valide (lorsque les données fournies à l'entrée sont arrivés à la sortie), le nombre d'itérations à la sortie doit être plus grand que 0 (car au moins une itération est toujours réalisée).
Les valeurs de `Cr` et `Ci` sont également propagées dans des registres pour fournir à chaque itération le bon point de départ. Ceci aurait pu être réalisé dans les itérateurs pour simplifier un peu le code du pipeline.

## 1.6 Pipeline wrapper

Le pipeline wrapper est plus simple que le loop wrapper, il doit simplement transmettre les points fournis par le générateur de nombres complexes au pipeline à chaque coup d'horloge. Les valeurs à la sortie du pipeline sont directement écrites dans la BRAM du `lpsc_mandelbrot_firmware.vhd`.

### 1.6.1 Zoom

Pour vérifier la vitesse de rafraichissement du pipeline, un zoom a été implémenté en faisant varier le point de départ et l'incrémentation du générateur de nombre complexe. Cette approche permet d'utiliser uniquement des compteurs et une incrémentation au lieu de stocker tous les zoom dans une mémoire (comme avec `c_gen.vhd`).
Le zoom représente une grosse partie du process synchrone du pipeline wrapper mais consiste simplement en un compteur qui fourni une horloge à 10 Hz pour effectuer l'itération de zoom et un autre compteur pour revenir à l'état initial (pas de zoom) après une centaine. Il y a aussi un temps d'attente de ≈1 min dans l'état de zoom minimal et maximal afin de laisser le temps d'observer la fractale.