h e p i a

Haute école du paysage, d'ingénierie et d'architecture
MRU - LPSCP
Prof. Fabien Vannel, Joachim Schmidt

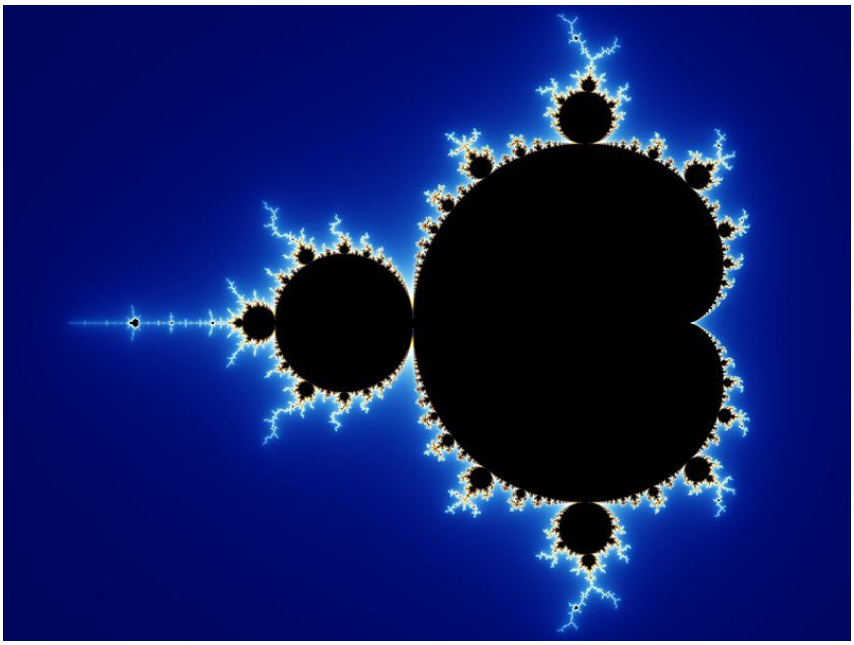
Hes·so
Haute Ecole Spécialisée de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences Western Switzerland

Fractales
semestre printemps 2021

Projet

1 Objectifs

Cet énoncé a pour but d'aider à la conception d'un composant capable d'effectuer les calculs nécessaires à l'affichage de la courbe de Mandelbrot.



1.1 Description

Nous allons commencer par réaliser un composant capable de calculer le résultat de l'équation de Mandelbrot pour un point donné. Les calculs doivent se faire en virgule fixe.

La courbe de Mandelbrot est construite en appliquant itérativement la fonction $Z_{n+1} = Z_n^2 + C$, où C est le point à évaluer, et $Z_0 = 0$. Le calcul se fait à l'aide de nombres complexes, et si une des itérations génère un point en dehors du cercle centré en $(0, 0)$ de rayon 2, alors le point est en dehors de la courbe. Si tel est le cas, le nombre d'itérations pour atteindre la frontière doit être retourné.

Un nombre maximum d'itérations est défini, et si ce nombre est atteint sans que Z n'ait dépassé le cercle de rayon 2, le nombre d'itérations retourné sera 0. Dans tous les autres cas, le Z

résultant de la dernière itération sera retourné, afin de pouvoir afficher correctement la courbe.

Il existe de nombreuses variantes à cette courbe de Mandelbrot. Internet sera une bonne source pour trouver des formules mathématiques. Par exemple $Z_{n+1} = Z_n^3 + C$, ou $Z_{n+1} = Z_n^4 + C$

1.2 Réalisation du bloc CALCUL

L'entité suggérée du composant pourra être la suivante :

```
entity mandelbrot_calculator is
generic ( comma     : integer := 12;  -- nombre de bits après la virgule
    max_iter : integer := 100;
    SIZE     : integer := 16);
port(
    clk         : in    std_logic;
    rst         : in    std_logic;
    ready       : out   std_logic;
    start       : in    std_logic;
    finished    : out   std_logic;
    c_real      : in    std_logic_vector(SIZE-1 downto 0);
    c_imaginary : in    std_logic_vector(SIZE-1 downto 0);
    z_real      : out   std_logic_vector(SIZE-1 downto 0);
    z_imaginary : out   std_logic_vector(SIZE-1 downto 0);
    iterations  : out   std_logic_vector(SIZE-1 downto 0)
);
end mandelbrot_calculator;
```

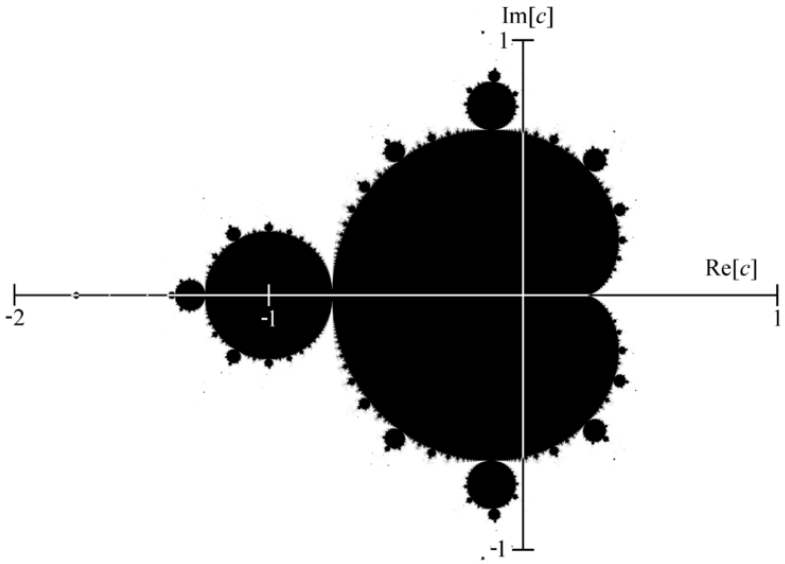
Les paramètres génériques permettent de définir la taille des opérandes, la place de la virgule, ainsi que le nombre maximum d'opérations.

Le nombre à évaluer est fourni sous la forme d'une partie réelle (c_real), et une partie imaginaire ($c\_imaginary$). Le résultat, donc le dernier Z l'est également de la même manière, avec ($z\_real, z\_imaginary$). Enfin le nombre d'itérations est également retourné.

Votre calculateur doit indiquer s'il est prêt à effectuer un calcul, en activant $ready$. Si tel est le cas, l'entrée $start$ lance un calcul. Lorsque celui-ci est terminé, vous devez activer $finished$ pendant un coup d'horloge, en fournissant les sorties z_* et $iterations$.

Votre implémentation est libre quant à son fonctionnement. Vous pouvez très bien effectuer un calcul complet avant d'en accepter un autre, ou réaliser une structure un peu plus pipelinée (qui devrait nettement complexifier le système).

Il vous est vivement suggéré de réaliser un banc de test pour vérifier le bon comportement de votre système.



2 Déroulement du projet

Le projet représente l'équivalent de 7 séances de laboratoire. Il est effectué individuellement.

Chaque personne devra, dans la réalisation de son projet, mettre un effort supplémentaire sur un point particulier à choisir dans la liste suivante :

1. Gestion d'une fractale de taille plus grande en utilisant la mémoire externe DDR ;
2. Précision augmentée de la fractale en travaillant avec des nombres à virgule fixe de plus grande taille et intégration d'un pipeline dans le bloc calcul permettant d'augmenter la vitesse de calcul. Analyse des performances et utilisation des ressources en fonction de différentes tailles.
3. Précision augmentée de la fractale en travaillant avec des nombres à virgule flottante et analyse des ressources et performances ;
4. Exécution en parallèle des calculs par la réplication de plusieurs blocs de calculs identiques ;

Le projet devra être géré au travers d'un environnement de gestion de projet GIT. Le professeur et l'assistant devront disposer des droits de consultation.

3 Evaluation du projet

Le projet sera évalué de la façon suivante :

- Présentation individuelle sous la forme de questions/réponses d'une durée d'environ 10 minutes. Tout support (informatique ou manuscrit) sous la forme de schéma sera le bienvenue entre autre afin de présenter l'architecture globale de votre système, d'indiquer les type d'implémentation choisies (IP core, VHDL, ...) et donner des résultats et analyses de performances.
- Rapport écrit à rendre. Ce rapport doit reprendre les différents points cités dans la présentation et bien détailler l'architecture du système. Il doit également donner des informations sur l'utilisation des ressources du FPGA (en pourcentage d'utilisation des différents blocs) et les analyser. Même chose pour les fréquences d'horloges utilisées et possible. Un effort sera donné sur l'implémentation du calcul de la fractale, entre autre en discutant les résultats d'intégration et d'utilisation dans le FPGA des ressources spécifiques. Un mémoire d'une dizaine de page devrait pouvoir être suffisant. Les schémas étant plus que souhaités.
  Dans le cas où la réalisation d'un élément supplémentaire de la liste ci-dessus à abouti, le mémoire sera en contrepartie facultatif. Sinon le mémoire devra donner les compléments d'explications nécessaire à comprendre les parties manquantes à développer.
- Démonstration
- Participation régulière au projet. L'analyse des *commits* sous GIT pourra être utilisés à ces fins.

Documents informatiques à rendre : archive du projet Xilinx, rapport, documents utilisés pour la présentation.