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

Laboratoire 03 :

Generateur de mandelbrot

Département : EIE
Unité d'enseignement : LPSC

Auteurs : Quentin Müller
Professeur : Fabien Vanel
Assistant : Joachim Schmidt
Classe : A04
Date : 31 mai 2022

Table des matières

1 Introduction **3**
1.1 Fractale de mandelbrot 3

2 Analyse du projet **4**
2.1 Analyse de la donnée 4
2.2 Analyse pratique 5
2.3 Test bench python 6

3 Implémentation sur la FPGA **8**

4 Conclusion **9**
4.1 Utilisation des ressources 9
4.2 Gestion des timing 9
4.3 Travail supplémentaire 9
4.4 Finalité 10

5 Signatures **10**

6 Annexes **11**

1 Introduction

Dans le cadre du cours de Logique Programmable pour Systèmes Complexes (LPSC), il nous a été demandé de concevoir et implémenter une fractale de mandelbrot. Pour ce faire, on nous a fournis un projet de base avec déjà la possibilité d'afficher sur un écran une image près programmée.

Le projet consiste donc à créer un iterateur de mandelbrot et la modification du programme principale afin de travailler sur deux horloges différentes.

1.1 Fractale de mandelbrot

Avant de commencer le projet il est important de comprendre comment est générer la fractale de mandelbrot visible sur la figure 1.

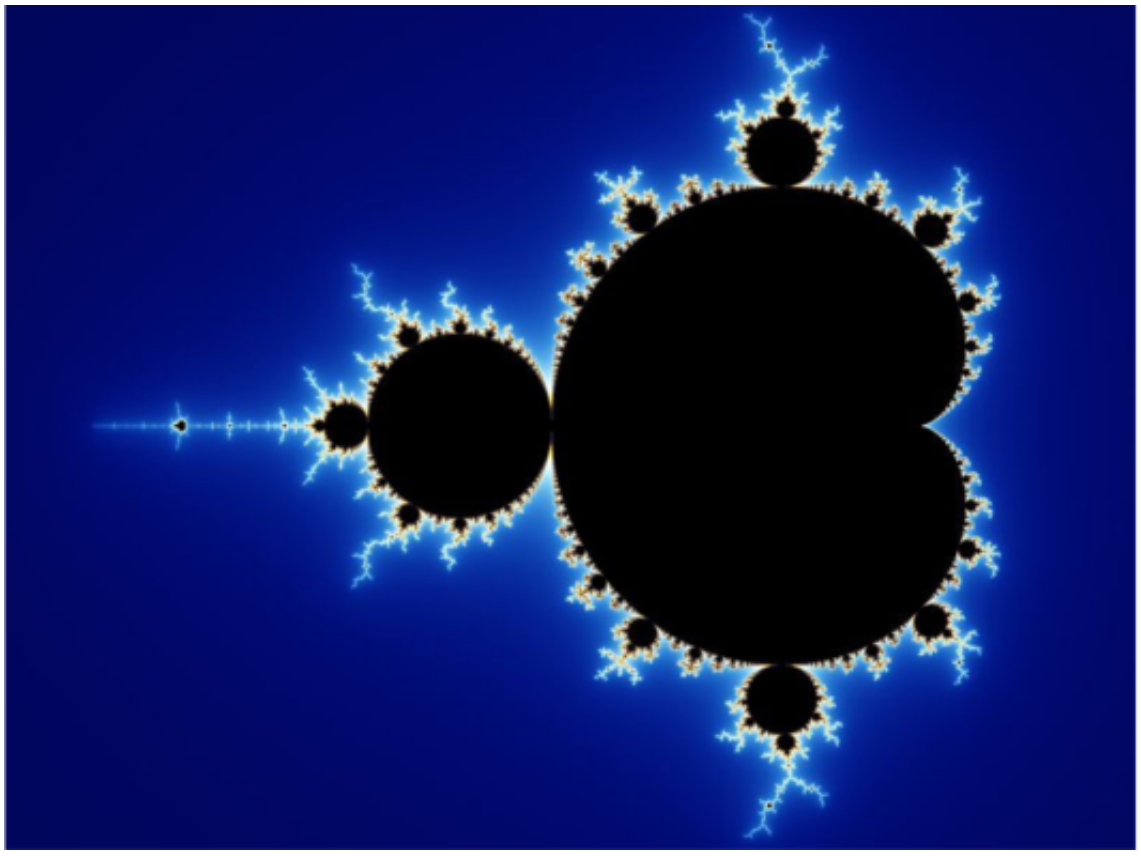

FIGURE 1 – Fractale de mandelbrot

La fractale de mandelbrot est construite de manière itérative par le calcul suivant.

$$Z_{n+1} = Z_n^2 + C \tag{1}$$

Avec $Z_0 = 0 + 0j$ car ce calcul se passe dans le domaine complexe. C est la coordonnée en x pour la partie réelle et y pour la partie imaginaire du pixel en cours

de calcul. Si le point Z_{n+1} se trouve en dehors du cercle de rayon 2 alors se point est en dehors de la courbe de mandelbrot. Le nombre d'itération maximale est fixé à 100.

2 Analyse du projet

2.1 Analyse de la donnée

Ce projet s'est dérouler en trois phases. La première étant l'analyse est la compréhension du code fournit pour l'utilisation de l'HDMI. Dans un premier temps nous avons vérifier le bon fonctionnement du code, puis modifier celui-ci afin de travail à deux fréquence d'horloges différentes. Car l'HDMI doit travailler à une fréquence particulière en fonction de la résolution demandée.

La deuxième est la réalisation et le test d'une étage de calcul de l'itérateur de mandelbrot. Pour ce faire voici comment est construit un étage de mandelbrot.

$$\begin{cases} Z_{n+1} = Z_n^2 + C \\ Z_n = Z_{real} + iZ_{imag} \\ C = C_{real} + iC_{imag} \end{cases} \rightarrow Z_{n+1} = (Z_{real} + iZ_{imag})^2 + C_{real} + iC_{imag}$$
$$Z_{n+1} = Z_{real}^2 - Z_{imag}^2 + 2 \cdot Z_{real} \cdot Z_{imag} i + C_{real} + C_{imag} i \tag{2}$$
$$\begin{cases} \Re(Z_{n+1}) = Z_{real}^2 - Z_{imag}^2 + C_{real} \\ \Im(Z_{n+1}) = 2 \cdot Z_{real} \cdot Z_{imag} + C_{imag} \end{cases}$$

Et il ne faut pas oublier le calcul du module de Z_{n+1} afin de vérifier la divergence

$$\begin{aligned} r &= \sqrt{Z_{real}^2 + Z_{imag}^2} \\ r^2 &= Z_{real}^2 + Z_{imag}^2 \end{aligned} \tag{3}$$

Mais comme nous allons le comparer à une valeur fixe (2), il est tout à fait possible voir obligatoire sur une FPGA de ne pas calculer le rayon r mais le rayon au carré r^2 qui sera fixé à une limite de (4).

Le troisième point est l'optimisation du projet par une des quatre points fixes dans la données visible ci-dessous.

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

2.2 Analyse pratique

Dans cette section, nous allons parler des problèmes liés à l'utilisation d'une FPGA dans ce projet. Car celle-ci ne peut pas utiliser de chiffre à virgule flottante comme un microcontrôleur.

Comme les calculs sont réalisés sur une FPGA, il est plus adapté d'utiliser des nombres en virgule fixe. Comme les bus d'entrée d'un DSP sont sur 18bits les nombre à virgule fixes seront eux aussi sur 18bits avec la répartition suivante :

— Un bit de signe.

— Trois bits avant la virgule (pour aller de 7 à -8).

— Quatorze bits après la virgule ce qui permet d'avoir une résolution de $6.10{\cdot}10^{-5}$.

Après cette analyse rapide du projet voici donc le bloc d'un étage du calculateur de mandelbrot.

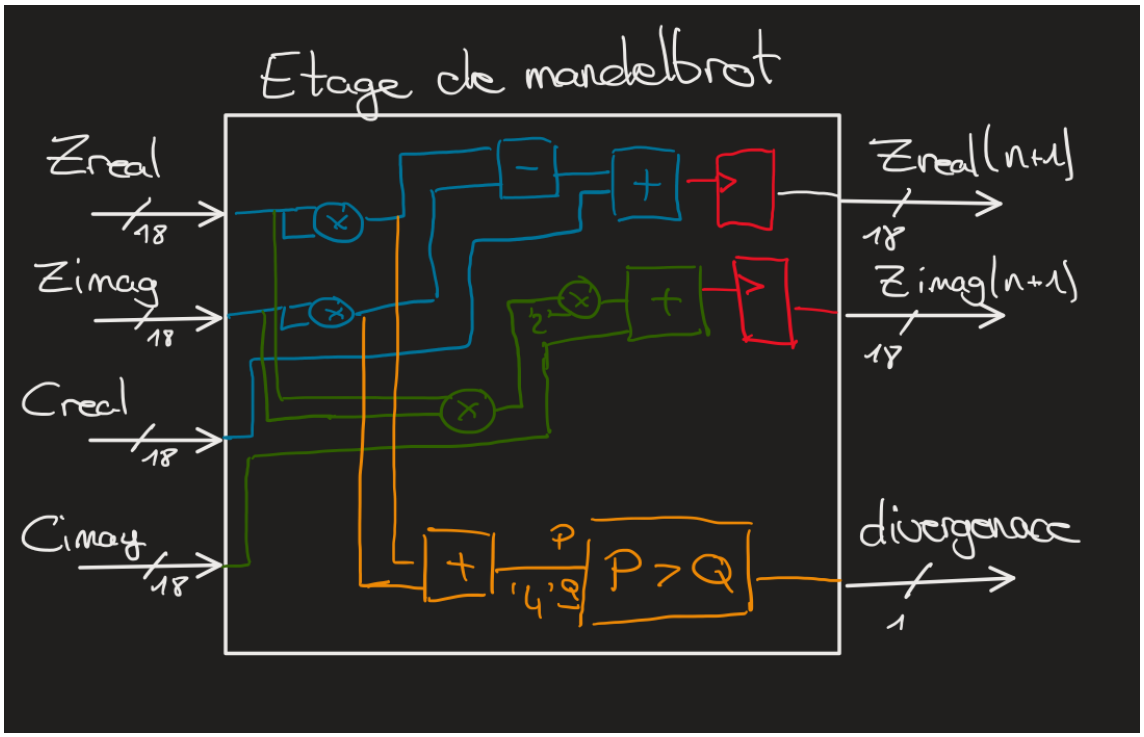

FIGURE 2 – Étage de mandelbrot après analyse

Puis une fois que ce bloc sera validé par un test bench alors il sera possible de l'inclure dans une machine d'état afin de faire l'itération mais aussi le stockage des valeurs de sorties.

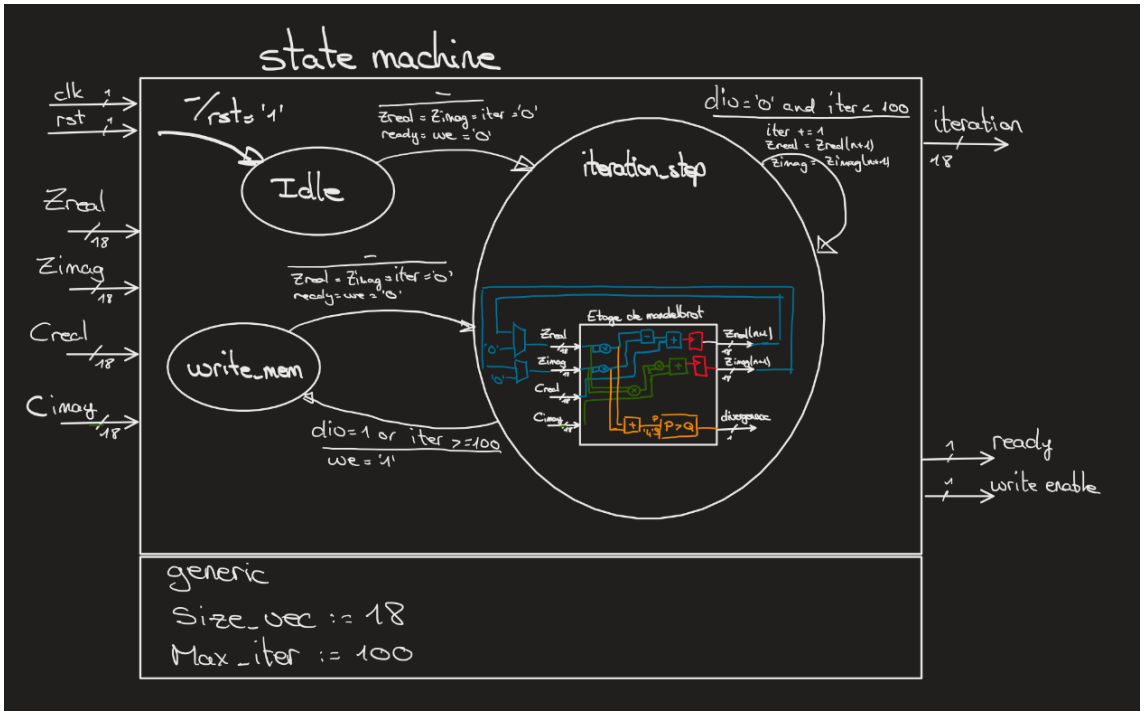

FIGURE 3 – Machine d'état de mandelbrot après analyse

2.3 Test bench python

Avant de passer au codage en VHDL il est important de bien comprendre le système à développer. C'est pour cela que j'ai commencé par écrire une code sur python pour faire exactement la même fractale de mandelbrot d'abord avec des virgules fixes pour être sur de la méthode puis avec des virgules fixes pour avoir des valeurs de comparaison pour la suite du projet.

Voici sur les figures le résultat de cette première implémentation sur python.

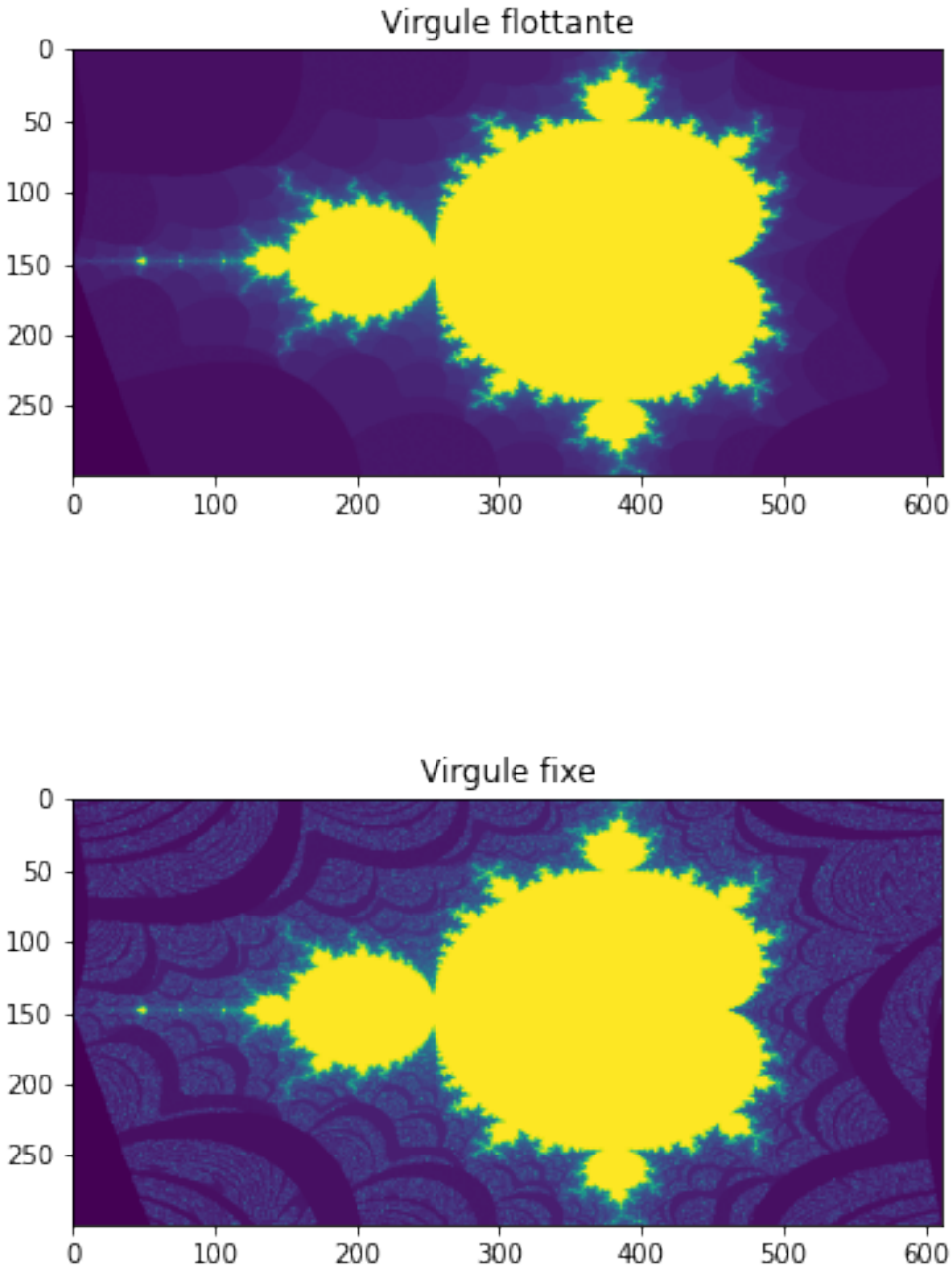

FIGURE 4 – Comparaison entre les deux fractales

3 Implémentation sur la FPGA

Le projet est constitué de trois étages l'étage du calculateur de mandelbrot puis en dessus la machine d'état de mandelbrot enfin au dessus de tout cela il y a le projet fourni par le professeur qui gère la mémoire BRAM dans laquelle on viens écrire la valeur du nombre d'itération afin d'afficher correctement la fractale.

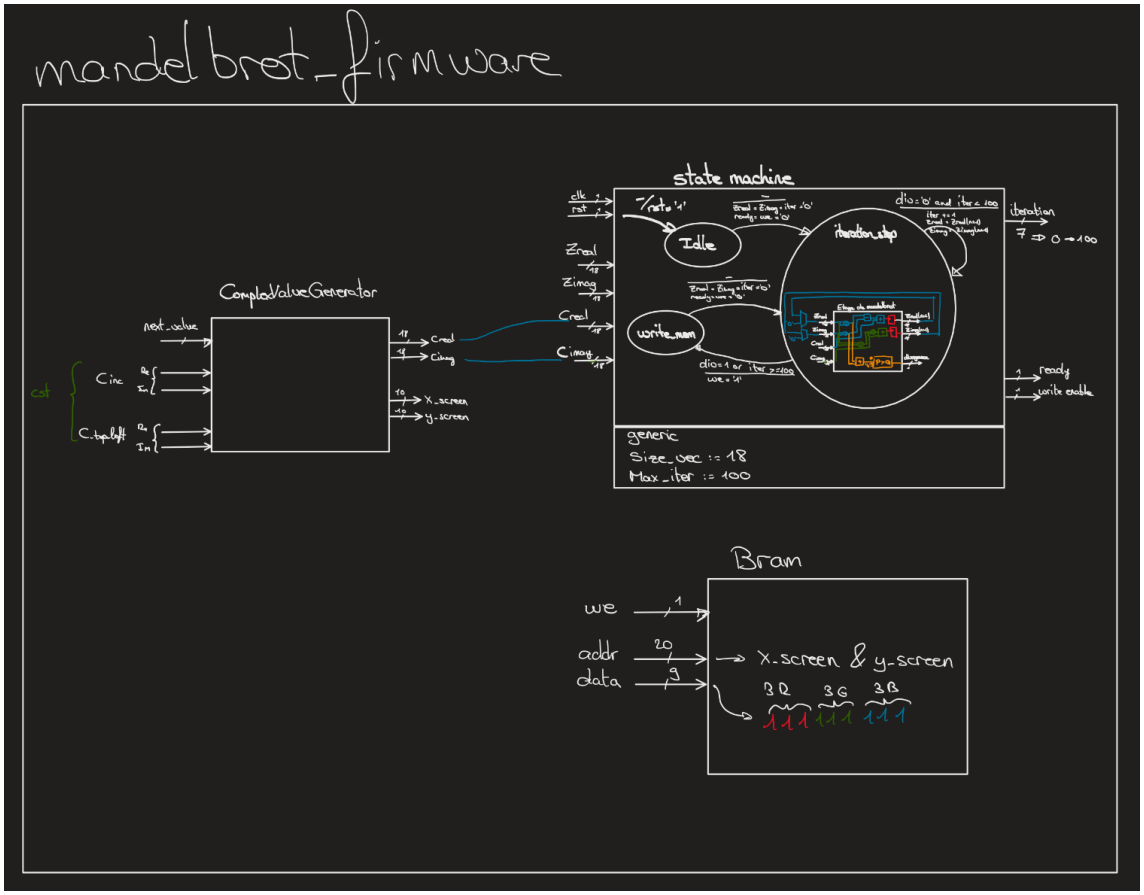

FIGURE 5 – Projet complet

Sur la figure 5, nous pouvons observer les trois principaux blocs utilisé pour le projet soit la machine d'état faite 'maison' le bloc BRAM une IP fournie par vivado utilisé pour changer de domaine d'horloge et le bloc ComplexValueGenerator fourni par le professeur. D'autres blocs sont utilisés comme le bloc de gestion de l'HDMI mais celui n'a pas du tout été toucher dans ce laboratoire alors Je n'en parlerais pas dans ce rapport.

4 Conclusion

4.1 Utilisation des ressources

Sur la figure 6, nous pouvons observer l'utilisation des ressources de la FPGA qui n'est de loin pas utilisée à fond ceci principalement du au fait que je n'ai pas eu le temps de mettre en place le pipeline. Ni dans le calculateur ni dans l'itérateur.

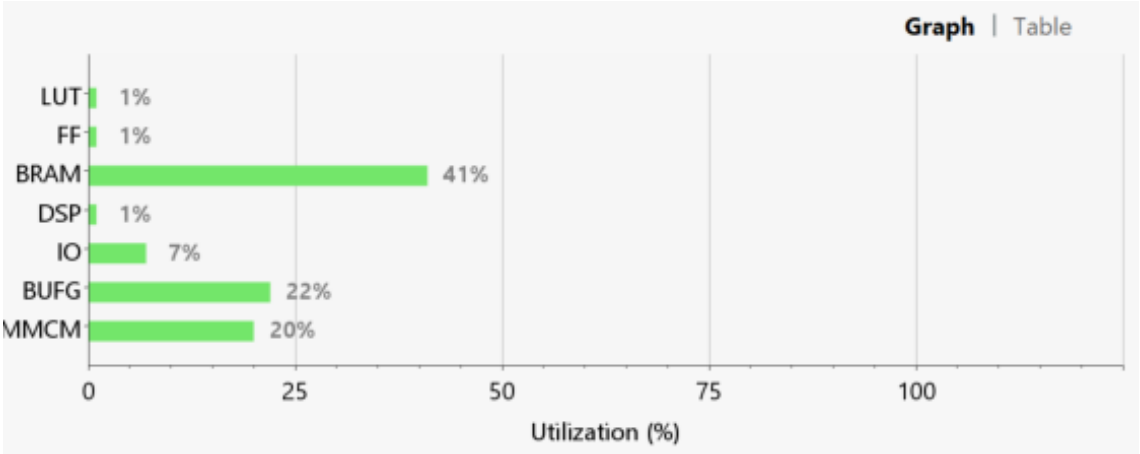

Figure 6 – Utilisation des ressources

4.2 Gestion des timing

Afin de ne pas violé les règles du cas critiques, j'ai décidé de descendre la fréquence de mon mandelbrot à 50$[MHz]$. Je n'ai pas fait de test pour essayer d'augmenter cette fréquence mais avec celle-ci j'ai n'ai plus aucune violation.

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	0,407 ns	Worst Hold Slack (WHS):	0,127 ns	Worst Pulse Width Slack (WPWS):	1,747 ns
Total Negative Slack (TNS):	0,000 ns	Total Hold Slack (THS):	0,000 ns	Total Pulse Width Negative Slack (TPWS):	0,000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	5558	Total Number of Endpoints:	5558	Total Number of Endpoints:	502

All user specified timing constraints are met.

Figure 7 – Visualisation des timing

4.3 Travail supplémentaire

Comme dit auparavant je n'ai pas réussit à mettre en place un des points supplémentaires proposés. Mais j'ai tout de même tenté une réalisation d'un étage de mandelbrot avec du pipeline. Mais à cause d'un manque de temps et d'autres projets en parallèle, je n'ai pas pu le rendre fonctionnelle. Ci-dessous une explication rapide du système voulu.

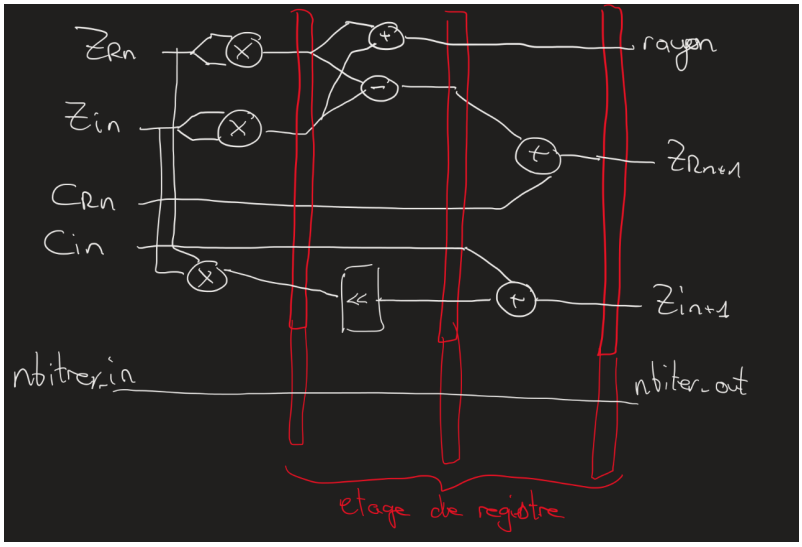

FIGURE 8 – Schéma bloc du pipeline

Le pipeline se compose de trois étages dans le premier se situe le calcul des valeurs de Z réel et imaginaire au carré ainsi que la multiplication des deux valeurs. Le deuxième étage au calcul du rayon avec les deux valeurs au carré, de la suite du calcul de la valeur suivante de Z réel et aussi du fois 2 sur le signal intermédiaire de z imaginaire. Enfin dans le troisième étage nous avons addition des valeurs de l'étage deux avec les valeurs de C. Avec un pipeline à trois étage nous avons la possibilité de calculer trois pixels en même temps ce qui réduira le temps de calcul de trois fois mais pour faire cela il faut correctement gérer l'entrée et la sortie du pipeline ainsi que les adresses des pixels en cours de traitement et du nombre d'itération avant leurs sortie du bloc de calculs.

4.4 Finalité

Pour conclure, ce projet m'a permis de mettre en oeuvre tous les concepts théoriques vus en classes et à aussi permis de mieux les appréhender et les comprendre. Une bonne analyse initiale et des simulations en cours de développement a permis une réalisation fonctionnelle sur la FPGA. Je suis satisfait de mon travail que je juge correctement réalisé selon les directives du cours. Je tiens à remercier Joachim Schmidt pour sa disponibilité et ses conseils avisés.

5 Signatures

Lausanne le 31 mai 2022

Quentin Müller

Table des figures

1 Fractale de mandelbrot 3

2 Étage de mandelbrot après analyse 5
3 Machine d'état de mandelbrot après analyse 6
4 Comparaison entre les deux fractales 7
5 Projet complet 8
6 Utilisation des ressources 9
7 Visualisation des timing 9
8 Schéma bloc du pipeline 10

Liste des tableaux

Liste des équations

1 Opération de mandelbrot 3
2 Calcul d'un étage de mandelbrot 4
3 Calcul du rayon de mandelbrot 4

6 Annexes