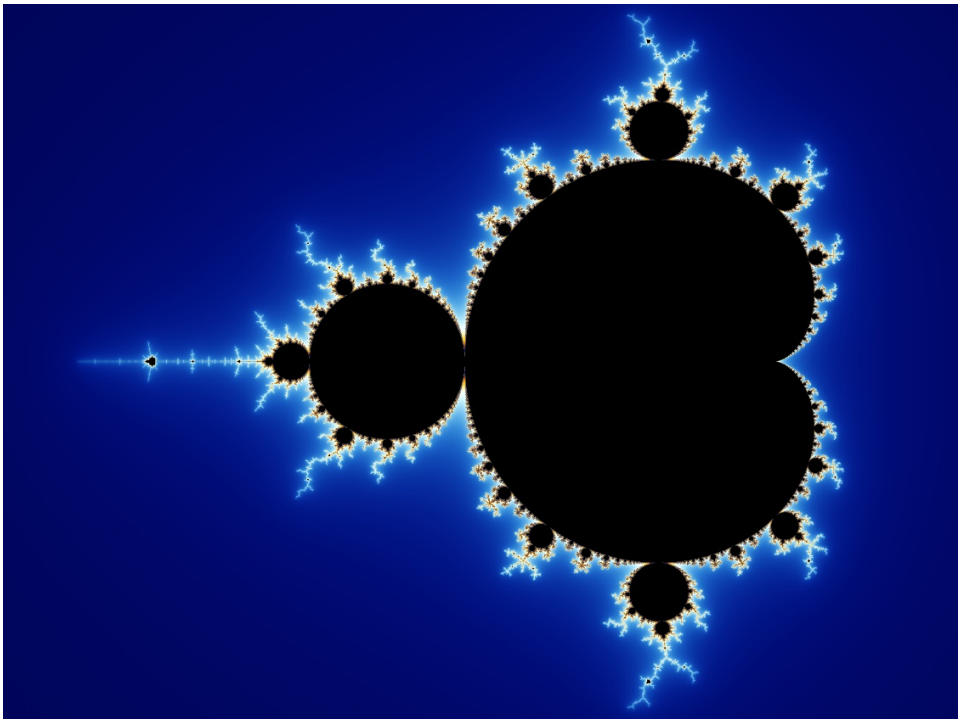
Fractales

Semestre printemps 2022

MA-LPSC

Glenn Muller



Introduction

L'objectif de ce projet est de réaliser un composant capable de calculer le résultat de l'équation de Mandelbrot pour un point donné. Le calcul doit se faire en virgule fixe. Le calculateur de Mandelbrot doit ensuite être intégré dans un système permettant l'affichage de la fractale. L'affichage se fera sur un écran externe en utilisant un bus HDMI. L'intégration du calculateur ainsi que l'afficha sont à faire sur la carte NEXYS Video du fabriquant Digilent. Elle possède une FPGA Artix-7 de chez Xilinx.

Calcul de Mandelbrot

Le calcul se fait de manière itérative en appliquant la formule suivante $\boxed{Z_{n+1} = Z_n^2 + C}$. Le calcul se fait à l'aide de nombres complexes, et si une itération génère un résultat en dehors de du cercle centré en (0,0) de rayon 2, alors le point sera en dehors de la courbe et le nombre d'itération nécessaire à atteindre la limite du cercle devra être retourné. Avant tout, un nombre d'itération maximale doit être défini. Si ce nombre est atteint sans avoir dépassé le cercle, alors le nombre d'itération retourné sera égal à 0. Sinon, le Z résultant de la dernière itération est retourné.

Équation de Mandelbrot

Comme vu dans la section précédente, l'équation de Mandelbrot comprend des nombres complexes. Cependant, il n'est pas possible d'utiliser des nombres complexes en VHDL, c'est pourquoi il faut séparer la partie réelle et imaginaire de l'équation initiale.

$$\boxed{Z = Z^2 + C}$$

$$\boxed{= (Z_{nR} + Z_{nIm} * J)^2 + (C_R + J * C_{Im})}$$

$$\boxed{= Z_R^2 + 2J + Z_R * Z_{Im} - Z_{Im}^2 + C_R + JC_{Im}}$$

Comme nous l'avons vu dans pendant le cours de présentation du projet, les équations réelles et imaginaires sont les suivantes :

$$\boxed{Z_R = Z_R^2 - Z_{Im}^2 + C_R}$$
$$\boxed{Z_{Im} = 2 * Z_R * Z_{Im} + C_{Im}}$$

Calculateur Mandelbrot

La réalisation du calculateur est décrite dans les sous-sections suivantes.

Entrée sortie

Les entrées et sorties du composant ont été données dans l'énoncé.

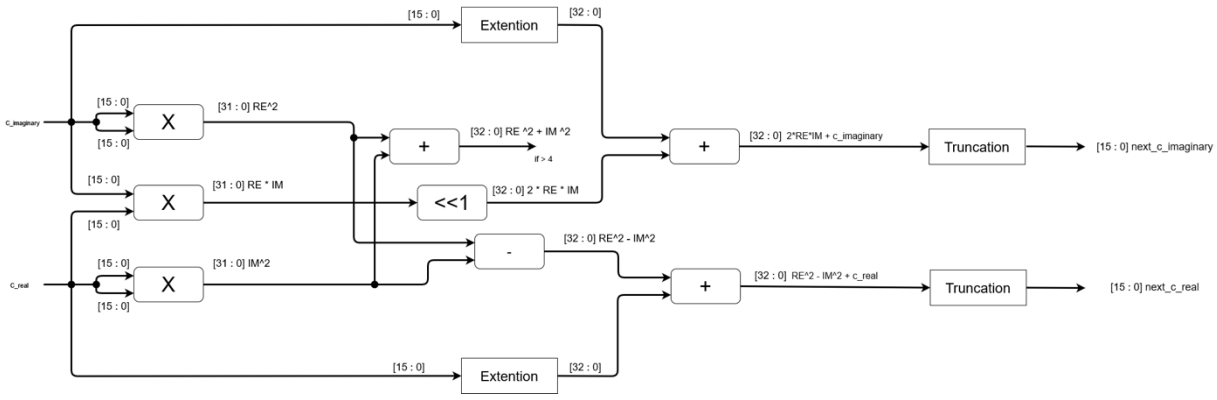
Nom	Direction	Type	Description
Clk	In	Std_logic	Horloge
Rst	In	Std_logic	Reset
ready	Out	Std_logic	Prêt à calculer
Start	In	Std_logic	Démarre le calcul
Finished	Out	Std_logic	Calcul terminé
C_real	In	Std_logic_vector(SIZE)	Partie réelle du nombre à évalué
C_imaginary	In	Std_logic_vector(SIZE)	Partie imaginaire du nombre à évaluer
Z_real	Out	Std_logic_vector(SIZE)	Partie réelle du résultat
Z_imaginary	Out	Std_logic_vector(SIZE)	Partie imaginaire du résultat
Iteration	Out	Std_logic_vector(SIZE)	Nombre d'itération effectué

Comme nous pouvons voir, certaines sorties et entrées n'ont pas de taille définie. Cela est dû au fait que le composant comporte les paramètres génériques suivants :

Nom	Type	Valeur initiale	Description
Comma	Integer	12	Nombres de bits après la virgule
Max_iter	Integer	100	Nombre d'itération maximale
SIZE	Integer	16	Taille des opérandes

Schéma calculateur Mandelbrot

Le schéma ci-dessous nous montre comment l'équation de Mandelbrot a été implémentée.



Toutes les étapes de calcul se font de manière combinatoire.

Pour obtenir une bonne précision à la fin du calcul, on augmente la résolution des résultats intermédiaires. Cela se définit de la manière suivante : si la taille des nombres en entrée est de 16 bits, alors le résultat obtenu après avoir effectué la mise au carré sera sur 32 bits. Cette hausse est due à la valeur maximale que peut atteindre le résultat.

L'élément extension est utilisé pour permettre l'opération suivante : 2*RE * IM + C_imaginary. Pour se faire, il faut augmenter la taille de l'entrée C_imaginary. Il est important de positionner la virgule au bon endroit. Il faut ajouter Size – comma + 1 de 0 sur la gauche du nombre et le nombre de 0 restant sur la droite. En faisant cela, on place la virgule au bon endroit tout en ayant la bonne taille pour effectuer l'opération.

L'opération *2 est effectuée avec un décalage à gauche. Pour garder une bonne précision, le résultat est codé sur 33 bits.

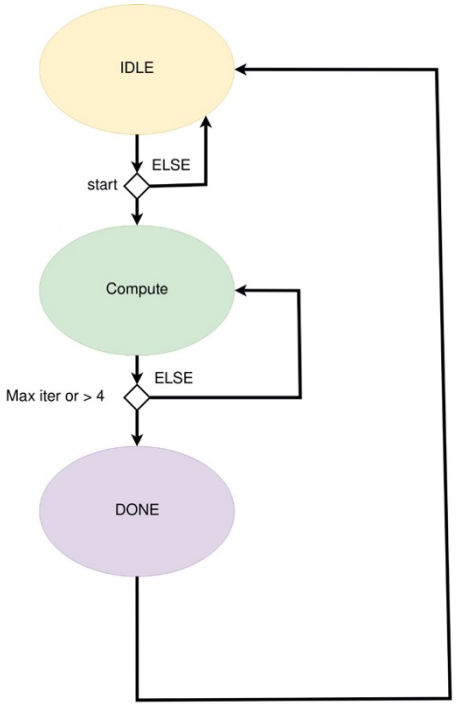
En fin de parcours, on effectue un troncage des résultats obtenus pour les mettre à la bonne taille. Ces troncages sont faits en prenant en compte les différents ajouts de bit dans les opérations intermédiaires, tout en respectant la position de la virgule spécifiée par le paramètre générique COMMA.

Compteur d'itération

Un compteur d'itération a été implémenté. Il est utilisé pour compter le nombre d'itération effectué lors du calcul. Il a été paramétré pour s'incrémenter de 1 à chaque coup de clock et il est possible de l'activer, de le désactiver et de le réinitialiser.

Machine d'état

L'exécution de l'équation de Mandelbrot a été implémentée dans le composant mandelbrot_calculator qui utilise une machine d'état.



L'état IDLE indique que le calculateur est prêt à être utilisé. Cela se traduit par une mise à 1 de la sortie READY, le compteur d'itération est mis à 0 et est désactivé. La sortie FINISH est mise à 0.

Quand on lance la séquence de calcul (mise à un de l'entrée START), le compteur est activé et les entrées C_real et C_imaginary sont utilisés pour le calcul.

L'état COMPUTE réalise le calcul de Mandelbrot. Après chaque coup de clock, le résultat obtenu est utilisé comme nouvelle entrée pour une nouvelle itération jusqu'à qu'une des deux conditions suivantes soit valide :

- Le calcul divergera si l'opération « RE^2 + IM^2 » est plus grand que 4, la machine d'état bascule en mode DONE.
- La limite d'itération a été atteinte, la machine d'état bascule dans l'état DONE, le compteur est mis à 0, ce qui permet d'afficher du noir dans le milieu de la fractale au lieu de blanc.

L'état DONE, la sortie FINISH est activée pendant un coup de clock. Le compteur est reset et la sortie READY est activée. Finalement, on bascule en mode IDLE.

Test du calculateur

Le test du calculateur s'est fait à l'aide d'un code python servant de référence et d'un banc de test. Le banc de test permet de spécifier des valeurs aux entrées C_real et C_imaginary. Les entrées du calcul sont les mêmes pour le code python et le banc de test.

Le banc de test se nomme tb_mandelbrot_firmware. Comme dit précédemment, il donne des valeurs prédéfinies en entrée du calculateur. Il applique également un reset au calculateur, puis une séquence de calcul est lancée.

Le code python se contente d'afficher les valeurs intermédiaires pour les valeurs en entrée (les mêmes que sur le banc de test). Ce code python a été réalisé rapidement et a permis à bien comprendre le calcul à implémenter en VHDL.

Intégration du calculateur

Le bloc mandelbrot_calculator a été implémenté dans le domaine FPGA User Clock Domain. Le bloc c_gen a également été implémenté dans le même domaine d'horloge.

C_gen

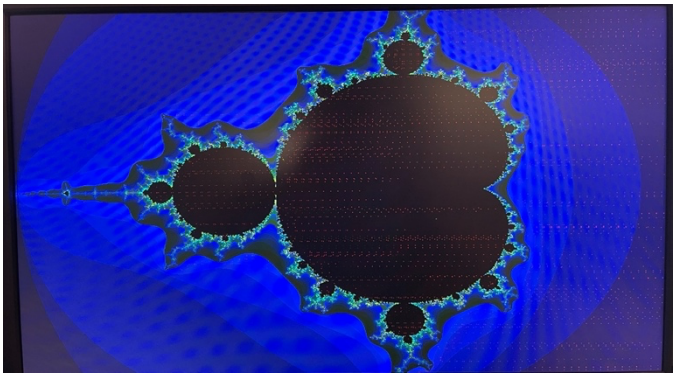
Le bloc c_gen permet de générer les nombres réels et imaginaires pour le pixel qui est actuellement en traitement.

Ce composant a été modifié pour qu'il fournisse de nouvelles valeurs lorsque le calculateur a fini de traiter le pixel. Pour cela, une entrée a été ajoutée EnableNext. Cette entrée est connectée à la sortie FINISH du calculateur. Dans le composant c_gen, cette entrée est connectée à l'entrée enablenext du composant complexe_generator.

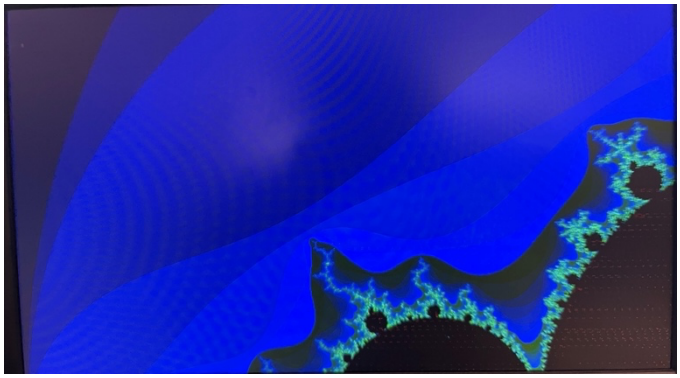
La fonction zoom a été également implémentée. Pour cela, il a fallu ajouter deux boutons aux entrées ZoomInxSI et ZoomOutxSI. Comme cette fonction était déjà implémentée dans le composant, l'ajout des boutons entrés est suffisant.

Résultats

L'image ci-dessous nous montre le résultat obtenu :

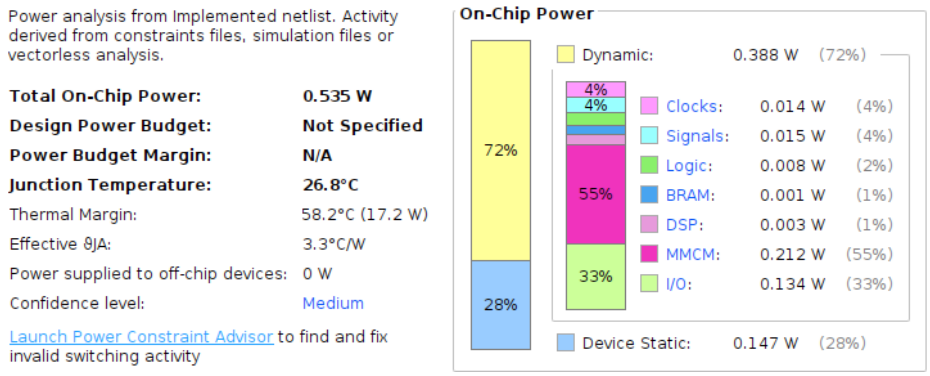


On peut voir que la fractale est visible et que le calculateur fonctionne comme souhaité initialement. Sur la figure ci-dessous, on peut voir le résultat obtenu après avoir utilisé la fonction zoom.



Performance énergétique

La consommation électrique de la FPGA est de seulement 0.535 W. Ce qui est extrêmement peu en comparaison à d'autres systèmes embarqués.



Sur la figure ci-dessus, on peut très facilement voir que la plus grande partie de la consommation est générée par les PLL. Le deuxième plus gros consommateur sont les I/O. Cela peut s'expliquer à la gestion du port HDMI par lequel passe toutes les informations d'affichages vers l'écran.

Performance Temporelle

Les performances temporelles obtenues lors de l'implémentation du projet ne respectent pas les contraintes définies. Le rapport temporel nous le montre par les valeurs obtenues dans la colonne Setup. En effet le Worst Negatif Slack est négatif. Cela indique que le chemin a échoué. Cela se confirme avec la valeur négative de Total Negative Slack. Pour résoudre ce souci de timing, une recherche plus approfondie est nécessaire. Une hypothèse est que le système implémenté est trop lent. Une solution serait de réaliser un pipeline sur le calcul de Mandelbrot et plus précisément sur la gestion des pixels.

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	-2.589 ns	Worst Hold Slack (WHS):	0.144 ns	Worst Pulse Width Slack (WPWS):	2.845 ns
Total Negative Slack (TNS):	-316.918 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	466	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	5619	Total Number of Endpoints:	5619	Total Number of Endpoints:	531

Timing constraints are not met.

Conclusion

Lors de ce projet, il nous avait été demandé de réaliser un calculateur de Mandelbrot ainsi que de son intégration dans un système permettant son affichage sur un écran externe via une sortie HDMI. A la fin de ce projet, le système est capable d'afficher le courbe de Mandelbrot, ainsi que d'effectuer un zoom sur cette dernière. Les améliorations du système sont multiples, mais deux seront très intéressantes à implémenter : la première est d'ajouter une gestion des pixels en parallèle pour obtenir des meilleurs résultats temporels. La deuxième amélioration qui serait possible une fois que la première ait été implémentée serait d'implémenter une fonction zoom plus poussée. En effet, cela serait un bon défi pour pousser les performances du système plus loin. Visuellement parlant, cela serait également plus intéressant.