GEI1064 Conception en VLSI
Filtre adaptatif FIR-LMS
Laboratoire A3

Jeremy Poupart et M. Ahmed Ouameur

OBJECTIFS

- ✓ Appliquer différentes techniques d'optimisations sur votre fonction FIR-LMS (version *int*) et comparer les résultats en termes de latences et de ressources utilisées (réf. chapitre 7 du tutoriel)
- ✓ Visualiser sur l'afficheur de formes d'ondes les entrées et la sortie de votre filtre FIR-LMS et valider son bon fonctionnement (réf. chapitre 8 du tutoriel)

OPTIMISATION DU CODE (REF. CHAPITRE 7)

Pour cette partie, vous devrez **reprendre la solution fournie du laboratoire 2 et changer les types de variables pour des entiers sur 32 bits (int)**. Plusieurs scénarios d'optimisations vous seront demandés. **Vous devrez donc créer plusieurs solutions**.

Vous devrez créer une solution pour chacun des points suivants:

- ✓ Optimiser seulement les boucles de votre code.
- ✓ Optimiser seulement les sous-fonctions.
- ✓ Optimiser votre fonction complète.

Vous pouvez utiliser une directive de type *dataflow* dans votre fonction principale pour assurer un transfert efficace des données entre vos sous-fonctions. La directive de type *dataflow* permet également un meilleur transfert de données entre les boucles. De plus, les sous-fonctions peuvent être amenées au même niveau hiérarchique que la fonction principale en ajoutant une directive de type *inline.* Pour optimiser votre code, des directives telles que *pipeline, unroll, array_partition,* etc, peuvent être utilisées. Pour plus d'information, vous pouvez vous référer à la documentation officielle de Vivado HLS.

VERIFICATION RTL (REF. CHAPITRE 8)

Pour cette partie, vous devrez créer un *RTL trace file* afin de visualiser et valider les formes d'ondes de votre design. **Vous devrez créer un test Bench** qui contiendra les points suivants :

- ✓ Une section du code C (Test Bench dans HLS) qui créera vos vecteurs d'entrées initialisés (valeurs d'entrées et valeurs de références) de taille fixe que vous choisirez (typiquement entre 5 et 10) ainsi que vos variables d'entrées initialisées (valeur du pas d'adaptation et du nombre

d'échantillons pour l'entraînement). Vous pouvez utiliser les données du code Matlab générer ces vecteurs.

- ✓ Une section du code C (Test Bench dans HLS) qui appliquera ces valeurs à la fonction du filtre FIR-LMS

Le but de cette section est donc de démontré le fonctionnement de votre filtre en RTL en comparant les entrées et les sorties obtenues. Vous devez donc faire correspondre vos valeurs de références à une combinaison linéaire des valeurs d'entrée. Vos variables devront être de type *float*.

QUESTIONS GÉNÉRALES

- ✓ Justifiez vos choix de directives pour chacune de vos optimisations à l'aide du *schedule viewer (analysis perspective).* Comparez avant et après.
- ✓ Analysez et comparez la différence en termes de latences, de ressources utilisées et de période minimale d'horloge pour chacun des types d'optimisations demandés. Il se peut que certaines combinaisons d'optimisations aient les mêmes performances que d'autres. Dans ce cas, cela signifie qu'elles sont tout simplement équivalentes.
- ✓ Discutez l'effet de la directive *inline* sur les ressources utilisées et les latences observées. À prendre note qu'il se peut que Vivado HLS utilise automatiquement la directive *inline.* Vous devrez donc la forcer à *off* pour faire votre analyse.
- ✓ Selon vous, dans quelles circonstances une directive de type *inline off* pourrait être utile ?
- ✓ Montrez un exemple concret du bon fonctionnement de votre filtre à l'aide de l'afficheur des formes d'ondes et expliquez.