ENSSAT
LANNION

Projet « Synthèse VHDL » 2019-2020

Cahier des charges

Préambule : Organisation / gestion de projet

Ce projet comporte 32h encadrées.

Chaque nouvelle séance de 4h commencera par une réunion de 10/15 minutes pour mettre en commun les avancées de chacun, les obstacles rencontrés, les objectifs. Vous serez les acteurs de cette réunion, il faudra donc préparer ce que vous y direz.

Pour ce projet, vous aurez un "**client**" qui ne sera pas votre encadrant de projet. Pour le groupe G1, votre client sera M. Casseau, pour le groupe G2 votre client sera Mme Dubois. Le client aura pour rôle de vérifier que vous avancez correctement dans la production des différentes étapes/versions du projet. A chaque fois que vous aurez validé une version, vous devrez en informer votre client.

Un fichier Crah (Compte rendu d'activité hebdomadaire) permettra de suivre l'avancement du projet. Vous devrez le mettre à jour et l'envoyer à votre client au plus tard **24h après chaque séance** de projet.
Le projet est divisé en plusieurs parties (les différentes versions à produire). Un planning prévisionnel est déjà précisé dans le fichier Crah. Par exemple, vous devrez avoir fini la version 0 avant la fin de la séance 1 et la version 1 au bout de 4 séances. Les colonnes version 0, 1, 2 etc. servent à préciser le temps passé (en heures). La colonne « Rapport » sert à indiquer le temps passé à la rédaction du rapport. La dernière colonne « Commentaire, bilan » sert à résumer le travail réalisé. N'oubliez pas de préciser la date de la séance en première colonne.

L'**évaluation du projet** se fera à partir des rapports écrits (intermédiaire et final), des codes commentés (VHDL synthétisable et test-benches), des simulations analysées (dans les rapports), et tiendra compte des étapes que vous avez validées avec votre client et de l'état d'avancement final.

Le matériel nécessaire au projet est disponible dans la mallette pédagogique que vous avez dû emprunter lors de votre première année. Seule la carte « Analog Test Shield » vous sera fournie par votre encadrant et devra être restituée à la fin du projet.
Une carte Explorer Digilent peut être un bon complément pour générer ou visualiser les signaux analogiques (du son par exemple) ou numériques (bus d'interfaçage).

Présentation du projet

Le but du projet est de développer une chaine de traitement audio qui devra tourner sur une carte FPGA.

Dans ce projet, nous utiliserons la plateforme DE0-Nano-SoC et l'outil de CAO Quartus.
Quelques **informations sur l'usage de Quartus + ModelSim** (pour la simulation) se trouvent sur Moodle dans le cours SNUM2a "Synthèse VHDL", partie «Tutorial … ».
La plateforme DE0-Nano-SoC contient entre autre :

- un circuit ARM+FPGA de type Cyclone V SE 5CSEMA4U23C6N d'Intel/Altera
- un convertisseur ADC 12 bits LTC 2308 interfacé par un bus série compatible SPI.

Nous utiliserons également une carte d'extension Analog Test Shield qui contient un convertisseur DAC 12 bits MPC 4725 piloté par un bus série I2C.
Les **documentations sur la plateforme DE0-Nano-SoC, l'ADC LTC 2308, la carte d'extension et le DAC MPC 4725** sont sur Moodle dans le cours SNUM2a "Synthèse VHDL", partie «Projet».

Principe du projet : Acquisition/restitution d'une séquence audio

Le schéma de principe de notre chaîne de traitement audio est donné en Figure 1. On fera l'acquisition de la séquence audio via un microphone (ou une autre source audio ou un signal sinusoïdal pour les premiers tests) + module ADC. La séquence d'entrée audio sera stockée dans une première mémoire double ports (*mem_acquisition*). Cette séquence sera ensuite lue selon les besoins du traitement et le résultat après traitement sera stocké dans une seconde mémoire double ports (*mem_restitution*). La séquence transformée sera finalement restituée via une lecture de cette mémoire + module DAC + un haut-parleur.

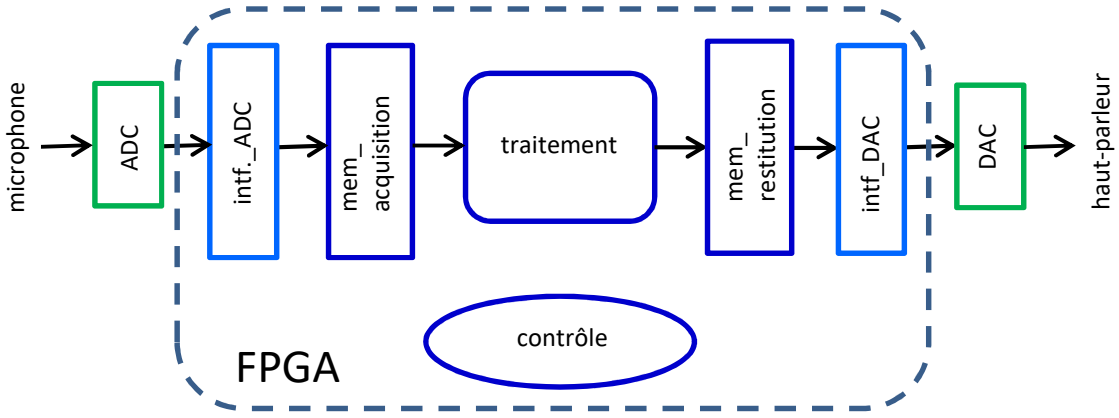

Figure 1 : Schéma de principe de la chaîne de traitement audio

Vous avez déjà travaillé en 1ère année avec l'ADC LTC 2308 disponible sur la plateforme DE0-Nano-SoC. Vous disposerez pour ce projet d'une description VHDL (ADC2.vhd) déjà faite d'une interface de cet ADC (bus SPI) qui permet de récupérer directement un échantillon audio 12 bits en parallèle de l'entrée "microphone" (Figure 1). Cette interface est conforme au cahier des charges du projet 1ère année "Logique programmable" (disponible sur Moodle).
De même, vous disposerez d'une description VHDL (DAC.vhd) d'une interface qui permet d'envoyer les échantillons traités au DAC MPC 4725 et de le piloter (bus I2C). L'entrée de cette interface est un échantillon audio 12 bits en parallèle.
Votre travail consistera à concevoir les autres parties de la chaîne de traitement.

Version 0 : Validation de l'utilisation des interfaces convertisseurs ADC et DAC

On se propose dans cette version 0 de valider l'acquisition d'un signal audio et sa restitution, sans aucun traitement autre que le stockage de l'échantillon audio d'entrée en provenance de l'interface ADC dans **un registre** 12 bits. La sortie de ce registre est directement mise en entrée de l'interface DAC.
La prise en main de cette version permettra de valider la bonne (compréhension et) utilisation des convertisseurs audio via leurs interfaces.

On partira d'un projet Quartus sous forme d'archive, *DAC.qar* , disponible sous Moodle (*Synthèse VHDL / Projet / Projet Quartus de départ*). Lancez Quartus et restaurez cette archive (*Project/Restore Archived project*). Le projet est déjà configuré et comprend la partie interface de l'ADC (à ne pas modifier), la partie registre 12 bits, la partie interface du DAC (à ne pas modifier), les interconnexions, le placement/configuration des entrée-sorties du FPGA (vers l'ADC, le DAC, etc.) pour être cohérent avec les cartes.

0.1) Analysez les interfaces des convertisseurs grâce à leurs documentations et faites le lien avec les descriptions vhdl.
Quelques questions à se poser : quel est le top level, quels et où sont les signaux audio, comment récupère-t-on le signal audio en sortie de l'interface ADC (synchronisation), quelles sont les entrées/sorties vers l'extérieur (microphone, haut-parleur), que vaut la fréquence d'échantillonnage, etc. ?

0.2) Faire la synthèse. Analysez le résultat (vue schématique (*Tools/Netlist Viewer* …), affectation des entrées/sorties du FPGA (*Assignment/Assignment editor* et *Pin planner*), etc.).

0.3) Dessinez proprement la vue architecture de cette version 0 ; cela vous permettra de bien comprendre cette version simple et vous fera gagner du temps pour la suite.

0.4) Charger la version 0 sur la carte et tester son fonctionnement. Pour simplifier les choses, on pourra mettre en entrée de l'ADC un signal sinusoïdal d'amplitude 1V et d'offset 1V. Visualiser les différents signaux (par exemple : le signal audio à différents « endroits » de la chaine de traitement).

Vous ne passerez à la version 1 qu'après avoir montré à votre encadrant le bon fonctionnement de cette version 0.

Version 1 : Chaîne de traitement élémentaire

Préambule : Description d'une mémoire RAM

Le but de ce préambule est d'expérimenter des écriture différentes de RAM et de trouver comment la décrire en VHDL pour qu'elle puisse être mappée dans les blocs RAM qui existent dans le FPGA.

1.1) Supposons qu'on échantillonne un signal à une fréquence d'échantillonnage 10 kHz et que les échantillons font 12 bits. On souhaite travailler sur un signal de plusieurs secondes (5 secondes par exemple). Quelle quantité mémoire est nécessaire ? Comparer à la quantité mémoire disponible dans les blocs RAM du FPGA ciblé. Est-ce compatible ?

1.2) *Impact du style d'écriture :*

Décrire en VHDL une mémoire RAM simple port asynchrone 64x12 bits (adresses, data_in, data_out, we). Faire la synthèse de cette mémoire et observez la manière dont l'outil l'a implantée (rapport de synthèse et vue technologique).

1.3) *Décrire en VHDL une mémoire RAM simple port synchrone 64x12 bits (adresses, data_in, data_out, clock, we). Faire la synthèse de cette mémoire et observez la manière dont l'outil l'a implantée (rapport de synthèse et vue technologique).*

1.4) *Tirer de ces expérimntations les conclusions sur le type de mémoire/style d'écriture qu'il faudra utiliser dans ce projet.*

Développement de la V1.

Nous allons maintenant valider le principe d'acquisition/restitution audio. Il s'agit stocker un signal audio d'une durée de 5 secondes (par exemple) dans *mem_acquisition*, puis de recopier le contenu de *mem_acquisition* dans *mem_restitution* (cf. Figure 1) et enfin d'écouter (haut-parleur) le contenu de *mem_acquisition*. Il n'y a aucun traitement particulier du signal audio (ce sera pour les versions suivantes). Ces opérations seront faites d'une manière séquentielle et bien séparées dans le temps. On utilisera pour cela une IHM basique : un interrupteur (bouton poussoir) *B1* pour réaliser l'acquisition, un interrupteur *B2* pour déclencher la recopie de *mem_acquisition* dans *mem_restitution* et un interrupteur *B3* pour déclencher l'écoute du contenu de *mem_restitution*.
Bien que 3 compteurs sont suffisants pour cette version 1, on utilisera 4 compteurs (1 par port de chaque mémoire) ce qui permettra d'anticiper les versions suivantes (*compteur1* sert à écrire dans *mem_acquisition*, *compteur2* sert à lire dans *mem_acquisition*, *compteur3* sert à écrire dans *mem_restitution*, *compteur4* sert à lire dans *mem_restitution*). Réfléchir un tant soit peu aux horloges de ces compteurs.
On ajoutera un interrupteur *B0* qui mettra à zéro les compteurs 1 et 3.
L'ordre d'exécution sera le suivant :

- avec *B0*, on mettra à zéro les pointeurs d'adresses en écriture des mémoires *mem_acquisition* et *mem_restitution*.
- L'enregistrement audio aura alors lieu lorsque l'utilisateur appuiera sur *B1*. Tant qu'on reste appuyer sur *B1*, on mémorise. Si on relâche *B1* et qu'on appuie à nouveau sur *B1* ensuite, l'enregistrement en mémoire reprend à l'adresse mémoire suivante de celle de la fin de l'enregistrement juste précédent.
- L'appui sur *B2* déclenche la recopie de *mem_acquisition* dans *mem_restitution,* depuis l'adresse 0 jusqu'à l'adresse de fin d'enregistrement dans *mem_acquisition*.
- On lancera l'écoute du contenu de *mem_restitution* via *B3*. On démarre à l'adresse 0 et on arrête de lire à la fin de la séquence audio mémorisée.

On complètera l'IHM avec 4 LEDs, une par compteur (LED allumée lorsque le compteur compte, LED éteinte sinon).

1.5) Coder les blocs nécessaires à la version 1 (compteurs, UC, …). Penser aux interconnexions (*B0, B1, B2, B3,* LEDs). Faire la synthèse. Vérifier la vue schématique (*Tools/Netlist Viewer* …), le nombre de blocs BRAM instanciés, etc. Charger la version 1 sur la carte et tester son fonctionnement.

Vous ne passerez à la version 2 qu'après avoir montré à votre encadrant le bon fonctionnement de votre chaîne de traitement élémentaire.

Version 2 : Filtrage

Nous pouvons maintenant envisager divers traitements conformément à la Figure 1. Le traitement à réaliser pour la version 2 est un filtrage passe bas. On pourra s'inspirer du FIR utilisé lors des TPs de ce module.

2.1) La fréquence de coupure souhaitée est environ 1000Hz. A vous de déterminer la taille du filtre, les coefficients, etc.

2.2) Coder ce traitement. On partira de la version 1 pour faire cette version 2.0. Coté IHM, on aura toujours *B0* pour faire la remise à zéro des compteurs 1 et 3, *B1* pour réaliser l'acquisition du signal audio dans *mem_acquisition*. *B2* servira à déclencher la lecture de *mem_acquisition* ainsi que les traitements et stocker les résultats dans *mem_restitution*. B3 servira pour déclencher l'écoute du contenu de *mem_restitution*.
Faire la synthèse etc. Charger la version 2.0 sur la carte et tester son bon fonctionnement.

Version 3 : *Open contest*

Vous avez maintenant à votre disposition une chaîne de traitement audio dont vous maîtrisez l'usage. Charge à vous d'imaginer un traitement audio à implanter… et à nous surprendre.