Utilisation de la sortie audio (Jack)

1. INTRODUCTION

La sortie Jack de la carte Nexys 4 (nommée « Mono Audio Out ») est reliée au FPGA à travers un filtre de Butterworth discret (filtre passe bas du 4ème ordre). Ce filtre de Butterworth est en charge de transformer le signal « binaire » sortant du FPGA (+Vcc/Gnd) en une tension « analogique » variant entre 0 et +5v (cf. la partie 15 du manuel de référence de la carte Nexys 4). Avec des bons timings, il est possible de reconstruire un signal audio à l'aide d'une seule sortie binaire et de ce filtre analogique.

Votre première tâche dans cette seconde partie est de concevoir le module situé en amont du filtre analogique. Pour cela vous devrez transformer des informations numériques représentant l'amplitude du signal audio sous la forme d'un signal au format PWM.

2. LE FORMAT/MODULATION « PWM »

Le format PWM (Pulse Width Modulation) est un format de représentation de l'information basé sur la longueur d'une impulsion sur un créneau temporel borné (cf. la documentation de la carte, section 15.1 Pulse-Width Modulation).

L'exemple suivant décrit le comportement temporel du système lorsque p cycles d'horloge sont disponibles pour coder un échantillon audio. Si l'amplitude du signal audio est égale à d, alors la sortie de la modulation PWM doit être égale à 1 durant les premiers d cycles. Le reste de la période, soit (p-d cycles), la sortie doit être maintenue à la valeur 0.

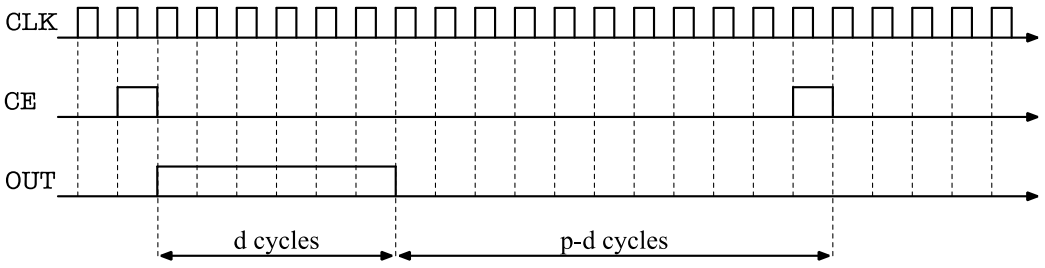


Figure 1 : Chronogramme d'une partie des E/S du module PWM.

Nous souhaitons ici gérer des séquences audio échantillonnées à 44100 Hz. Afin de calculer la résolution du module PWM, il va falloir prendre en compte de la fréquence d'horloge présente au sein du FPGA (100 MHz). En effet, cette dernière va limiter la dynamique des données utilisables.

Tache n°1

- Calculer la valeur maximale des échantillons audio (valeur de p) sachant que l'horloge présente au sein du FPGA est de 100 MHz.

2.1 Développement du composant

Maintenant que vous connaissez la dynamique d'entrée du module PWM, il ne vous reste plus qu'à le décrire en VHDL. La structure du module à concevoir est résumée dans la figure suivante :

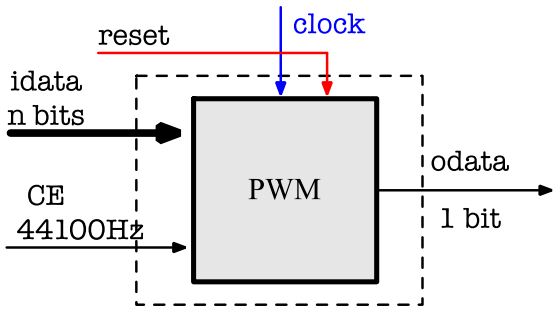


Figure 2 : Représentation des E/S du module PWM_mod.

Tache n°1

- Créez un nouveau fichier VHDL nommé PWM_mod.vhd. Décrivez le comportement du module pour qu'il corresponde au chronogrammes donnés en **Figure 1**.

Tache n°2

- Simulez le comportement de votre nouveau module VHDL à l'aide d'un testbench

3. CONCEPTION DU SYSTEME

Pour pouvoir jouer un son sur la sortie Jack de la carte, il est nécessaire d'alimenter votre module PWM avec des échantillons audio… Vous allez donc devoir concevoir un système légèrement plus complexe. Le schéma du système est présenté dans la figure suivante.

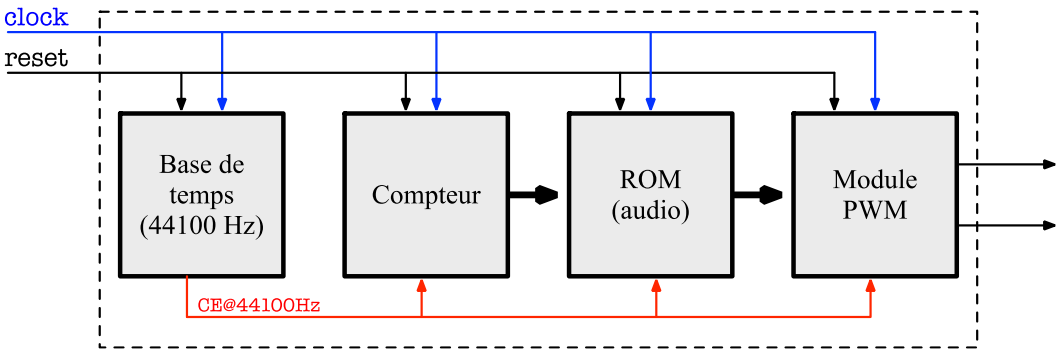


Figure 3 : système à concevoir afin de générer un signal audio.

Les séquences audio que vous manipulerez ont été échantillonnées à 44100 Hz, cela implique que les échantillons audio doivent être envoyés au module PWM à cette cadence. En conséquence, dans ce système la « base de temps » correspond à un compteur de période égale à 1s/44100 Hz.

Les échantillons audio seront stockés dans une mémoire de type ROM. Sa profondeur sera de 44100 données et chaque donnée sera codée sur 11 bits. Le compteur situé en amont de la mémoire ROM permettra de parcourir l'ensemble des données disponibles dans cette dernière au rythme de la base de temps.

Tache n°1

- Créez en VHDL le compteur d'adresses qui permettra de parcourir le contenu de la mémoire ROM.
- Téléchargez le module VHDL décrivant une mémoire de type ROM sur le site internet de votre encadrant (ROM_Sinus.vhd).

Tache n°2

- Associez l'ensemble des éléments du système.

Tache n°3

- Simulez le comportement du système pour valider son fonctionnement. Afin de vous aider dans cette tâche, vous prendrez soin de visualiser les valeurs transitant entre le module ROM et le module PWM (utilisez le mode d'affichage analogique dans ISIM).

4. IMPLANTATION SUR LA CARTE ET VALIDATION EXPERIMENTALE

Maintenant que votre système est complètement décrit et fonctionnel, il ne vous reste plus qu'à le tester sur carte afin d'écouter ce mélodieux sinus…

Tache n°1

- Écrivez le fichier xcd qui va bien,
- Lancez la synthèse et le placement routage,

- Chargez le bitstream sur la carte, écoutez et vérifiez que le signal est bien celui attendu ;-)

5. IMPLANTATION SUR LA CARTE ET VALIDATION EXPERIMENTALE

Afin de rendre ce système un peu plus flexible, il est nécessaire de remplacer cette mémoire de type ROM par une mémoire de type RAM. Toutefois pour permettre le chargement de séquences audio dans la mémoire RAM une fois le FPGA configuré, il est nécessaire de charger les échantillons audio depuis un PC à l'aide de la liaison série.

Le développement de la partie du système assurant cette fonctionnalité a été développée par vos encadrants. Vous devrez donc intégrer le module développé dans votre système. Le système que vous obtiendrez est schématisé dans la **Figure 4**.

Note : il ne vous est pas demandé de modifier ou de comprendre les modules fournis par vos enseignants (partie grisé dans la **Figure 4**).

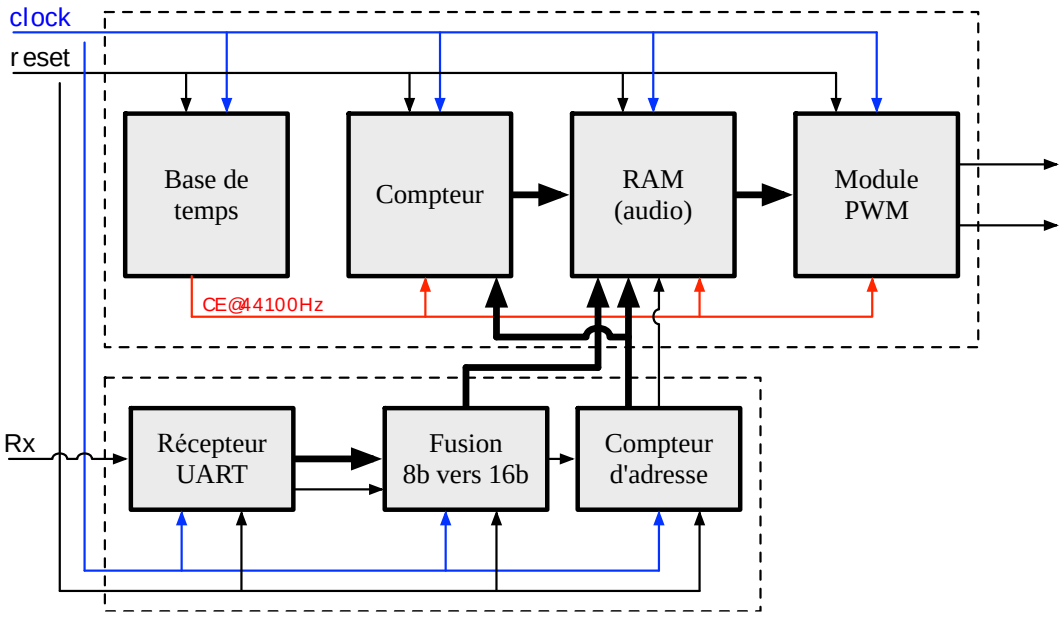


Figure 4 : système à concevoir afin de générer un signal audio.

Récupérez les fichiers en charge de la gestion UART sur le site internet de votre enseignant ([TP] Les fichiers réalisant le lien…) puis d'adaptez le système développé précédemment :

- Il faut prendre en considération l'écriture des données dans la mémoire (adresse d'écriture, signal de validation).
- Le compteur permettant de lire les données dans la mémoire doit être borné par le nombre de données actuellement disponible dans la mémoire.

Tache n°1

- Téléchargez les fichiers VHDL en charge de la communication avec la liaison série.
- Créez un module VHDL interconnectant votre développement avec les modules récupérés.

Tache n°2

- Créer ou modifier le fichier XCD nécessaire au processus d'implantation.

Tache n°3

- Générez le fichier permettant de configurer le FPGA avec votre système.

Tache n°4

- Configurez le circuit FPGA avec le bitstream généré.

Tache n°5

- Branchez **délicatement** un casque audio sur la sortie Jack de la carte pour entendre la douce mélodie produite.

Ajout d'une interface homme machine (IHM)

6. INTRODUCTION

Dans la première partie du TP vous avez développé un système permettant de contrôler la lecture de séquence audio. Pour cela vous avez géré les différents boutons présents sur la carte afin de contrôler des compteurs, piloter l'affichage sur les 7 segments et enfin générer des signaux de contrôle.

En l'état actuel de vos développements, les 2 parties ne sont pas complètement compatibles. Par exemple, vous ne pouvez pour le moment pas arrêter ou changer le sens de lecture de votre player audio. La gestion du volume n'a pas non plus été intégrée[1].

Tache n°1

- Dessinez le schéma du nouveau système et faites valider votre design par votre encadrant.

Tache n°2

- Adaptez votre player audio afin de supporter les signaux de contrôle issus de l'interface.
- Créez un module VHDL interconnectant les 2 sous modules.

Tache n°3

- Créer ou modifier le fichier XDC nécessaire au processus d'implantation.

Tache n°4

- Générez le fichier permettant de configurer le FPGA avec votre système.

Tache n°5

- Configurez le circuit FPGA avec le bitstream généré.

Tache n°6

- Testez votre système sur carte.

[1] Pour cette dernière, vous privilégierez une approche simple : nous diviserez la valeur de l'échantillon audio par une puissance de 2 ($2^{9\text{-volume}}$).

Ajout et réduction du bruit (filtrage)

Résumé : Dans cette partie nous allons nous intéresser à l'implantation matérielle d'un filtre passe-bas ainsi qu'à la génération de « bruit aléatoire ».

7. AJOUT DE BRUIT ALEATOIRE AU SIGNAL AUDIO

Avant de parler de filtrage, vous allez commencer par ajouter un « bruit aléatoire » à votre signal audio. Etant donné que la fonction rand n'existe pas en VHDL, vous allez devoir utiliser un générateur pseudo-aléatoire (comme rand), mais décrit au niveau RTL.

Pour vous faire gagner un peu de temps, un générateur pseudo-aléatoire à base de LFSR a déjà été sélectionné et modifié par votre enseignant.

Tache n°1

- Récupérer ce module VHDL (rand_mod.vhd) et ouvrez-le pour analyse.

Tache n°2

- Développez votre module qui utilise le générateur pseudo-aléatoire pour ajouter du bruit au signal d'entrée. Faites attention aux problèmes de saturation… Une entrée switch permettra d'activer ou de désactiver le module.

Tache n°3

- Simulez le module. Modifiez le système et implantez sur la carte pour valider fonctionnellement votre design.

8. FILTRAGE DU SIGNAL BRUITE

L'approche plus simple pour réduire le bruit dans un signal consiste à faire la moyenne des « n » dernier échantillons. Ce calcul agit sur le signal d'entrée comme un filtre passe bas.

Afin d'annuler tout ou partie du bruit dans notre signal audio, développer 2 modules VHDL : le premier calcule la moyenne des 4 derniers échantillons reçus, tandis que le second à une fenêtre d'observation 2 fois plus importante.

Les interfaces du module seront identiques aux interfaces précédemment utilisées, ainsi vous pourrez placer ce module ou vous le souhaitez dans la chaine de traitements numériques. Faites cependant attention à le positionner après la source de bruit…

Tache n°1

- Développez vos modules de filtrage. Faites attention aux problèmes de saturation… Deux switchs distincts permettront comme d'habitude d'activer ou de désactiver les modules.

Tache n°2

- Simulez le comportement de vos 2 modules.

Tache n°3

- Modifiez le système et implantez-le sur la carte pour valider fonctionnellement l'effet correcteur des filtres.

Implantation d'un effet « d'écho »

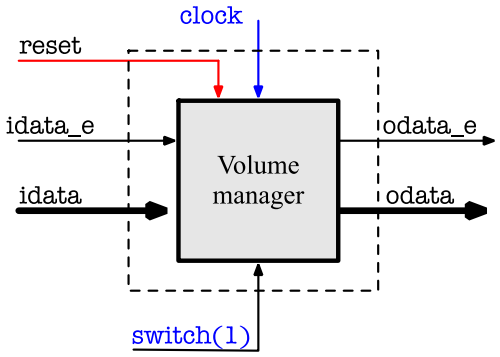
Résumé : Maintenant que votre système est fonctionnel, on va pouvoir lui adjoindre quels modules permettant de transformer le signal audio.

9. INTRODUCTION

Ajouter un effet d'écho à un signal audio revient à sommer à l'échantillon de l'instant t, la valeur de l'échantillon t-δ ; avec δ une valeur suffisamment grande (> 300ms). Pour pouvoir mémoriser un nombre aussi important de données vous êtes dans l'obligation d'utiliser des blocs RAM.

10. ARCHITECTURE DU MODULE A DEVELOPPER

Le module d'écho a des E/S identiques à celles du module de gestion du volume. Chaque nouvelle donnée est accompagnée d'un signal de validation (idata_e). Le comportement est analogue au niveau de la sortie comme cela est présenté dans la figure suivante.



Les interfaces adoptées par ce module permettent de le positionner soit en amont soit en aval du module de gestion du volume. Il est à noter que le signal provenant du switch 1 permet d'activer ou de désactiver l'ajout de l'écho au signal d'entrée.

Tache n°1

- Imaginez l'architecture interne du module. Vous avez à votre disposition soit une mémoire RAM, soit un registre à décalage. Ces 2 éléments de base peuvent être implantés sous la forme d'un bloc RAM.

Tache n°2

- Développez votre module, modifiez le système, simulez…

Tache n°3

- Implantez sur la carte et validez votre architecture.