HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Reconfigurable & embedded Digital Systems

Prof. Yann Thoma

Laboratoire de Programmation Temps Réel

semestre automne 2019 - 2020

Laboratoire 5 : Audio player

Temps à disposition : 6 périodes

Le laboratoire peut être effectué par groupe de 2 personnes.

Objectifs

Le but de ce laboratoire est de réaliser un lecteur audio capable de jouer des fichiers wav, et ce sur une carte DE1-SoC.

Vue du système

La DE1-SoC possède une interface audio accessible depuis la partie logique programmable. Le système qui vous est fourni contient la partie logique ainsi qu'un driver spécialement développé pour ce laboratoire. Ce driver vous permet d'envoyer des données sur les deux canaux audios. En interne il dispose d'un buffer de 128 mots par canal. De plus, il permet de piloter et scruter les différentes E/S de la carte (afficheurs 7seg, boutons poussoirs, leds et switchs).

Etape 1 : Mise en place d'une tâche périodique

Le code qui vous est fourni est fonctionnel. L'application lance une tâche qui joue le fichier wav donné en argument. Vous observerez que cette tâche tente d'écrire dans le buffer de manière continue. Elle va donc consommer le 100% du temps processeur.

En tant qu'expert de Temps Réel, vous savez qu'il est possible de faire mieux. Commencez donc par en faire une tâche périodique, afin de ne pas consommer inutilement (du temps et de l'énergie). En prenant en compte la taille du buffer du driver audio, à vous de calculer une fréquence pertinente pour cette tâche.

Après cette étape, le fichier devrait être lu correctement du début à la fin.

Etape 2 : Monitoring

Nous sommes maintenant intéressés à observer l'avancée du morceau. Pour ce faire vous disposez de 6 affichages 7 segments.

La fonction `display_time()` vous permet d'afficher une valeur sur cet affichage.

Il vous est demandé que l'affichage présente le temps écoulé depuis le début du morceau, à la cen-

tième de seconde près. Vous aurez donc 2 digits pour les minutes, 2 pour les secondes, et 2 pour les centièmes de seconde. Comme contrainte d'implémentation nous imposons le fait que ce monitoring se déroule dans une tâche dédiée, afin de ne pas perturber le rendu sonore.

Etape 3 : Contrôle

Une application qui lance directement la lecture d'un fichier audio et qui se termine ensuite n'est pas forcément d'une utilité incroyable. Nous désirons donc ajouter un peu de contrôle à notre lecteur. Ceci ce fera via les boutons présents sur la carte. Nous allons développer les fonctionnalités suivantes :

1. Play/Pause (key 0) : Lance la lecture du fichier. Si la lecture est cours, celle-ci se met en pause, et si elle est en pause, elle doit reprendre.
2. Stop (key 1) : Stoppe la lecture du fichier. Réappuyer sur Play relancera alors la lecture depuis le début.
3. Fwd (Forward) (key 2, switch0 off) : Avance la lecture de 10 secondes.
4. Rew (Rewind) (key 3, switch0 off) : Recule de 10 secondes dans le fichier.
5. Vol + (key 2, switch0 on) : Augmente le volume de sortie.
6. Vol - (key 3, switch0 on) : Réduit le volume de sortie.

Etant donné que la carte ne dispose que de 4 boutons, nous utiliserons un switch (celui le plus à droite) pour sélectionner entre l'avance/recule et la modification du volume.

Pour accéder aux boutons il est nécessaire d'aller vérifier l'état des boutons de manière active (pas d'interruptions générées). Une tâche périodique devrait faire l'affaire pour aller vérifier si les boutons sont pressés ou non ainsi que l'état des switchs. Les fonction `keys()` et `switches()` vous permet alors de récupérer ces informations.

Volume

Le volume sera compris entre 0 et 10. Les touches correspondantes doivent permettre d'augmenter ou diminuer celui-ci de 1. Les 10 leds de la carte vous permettent de représenter le volume actuel. La fonction `set_volume_leds()` permet d'appliquer la valeur de chacune des Leds à disposition.

L'intensité du signal audio est codé sur 16 bits. A vous d'appliquer la bonne opération en fonction du volume choisi. Considérez que le volume maximal correspond à jouer le fichier tel quel.

Avance/recule rapide

Cette fonctionnalité nécessite de synchroniser correctement la tâche lisant les capteurs et les autres. A vous de réfléchir au meilleur moyen de gérer cette synchronisation.

Play/pause/stop

Cette fonctionnalité nécessite de synchroniser correctement la tâche lisant les capteurs et les autres. A vous de réfléchir au meilleur moyen de gérer cette synchronisation.

Travail à effectuer

A vous de développer le code correspondant aux différentes étapes. Le code final devra être correctement commenté.

Un petit rapport présentant vos choix architecturaux vous est également demandé.

Le tout sera à rendre sur cyberlearn.

Mise en route de la carte DE1-SoC

Le laboratoire comprend plusieurs fichiers :

1. `De1-SoC.sof` : un bistream contenant un driver audio (hardware) s'interfaçant avec le codec audio de la De1-SoC et des PIO pour piloter/scruter les différentes entrées/sorties de la carte.
2. `rtsnd.ko` : un driver Linux sous forme de module permettant à la fois de maper les PIO et envoyer un flux audio vers le driver audio en hardware.
3. `*.wav` : des fichiers audio dont les données sont codées en format PCM (format brut).
4. `snd_player.c` : programme de départ lisant les fichiers wav et pilotant le driver audio. Un Makefile vous est fourni ainsi que la bibliothèque Xenomai.

Après la fin du boot, programmez la FPGA au travers du programmeur de Quartus. Une fois cela fait, transférez et chargez le module Linux sur la De1-SoC très simplement avec insmod : `insmod rtsnd.ko`. Sur la mahcine hôte, compilez le player à l'aide de Makefile fourni puis transférez le binaire sur la De1-SoC. A ce stade, vous pouvez lancer le player en lançant par exemple :

```
LD_LIBRARY_PATH=/usr/xenomai/lib/ ./snd_player bensound-dance.wav.
```

Caractéristiques du codec audio

Le codec audio de la De1-SoC est préconfiguré par le driver hardware sur la FPGA. Celui-ce ne peut être modifié après coup. Le codec fonctionne à une fréquence d'échantillonage de 48 Khz, sur 2 canaux et à une résolution de 16 bits. Si vous souhaitez jouer vos morceaux préférés sur la De1-SoC, convertissez les en wav format PCM à la même résolution et même fréquence d'échantillonage que le codec audio.