HEIG-VD | REDS Institut Reconfigurable and Embedded Digital Systems

Alexandre Corbaz, Catel Torres Arzur Gabriel

Laboratoire de Programmation Temps Réel

semestre printemps 2023 - 2024

Laboratoire 4 : Acquisition

Temps à disposition : 4 périodes

Objectifs

Le but de ce laboratoire est de réaliser un système capable d'acquérir différents types de données. Vous aurez des données audio ainsi que des données vidéos à traiter.

- Les données audios proviennent de la FPGA. C'est l'entrée "MIC IN" qui est utilisée. Vous devez connecter un micro (ou autre) pour produire des samples.
- Les données vidéos proviennent d'un fichier, cela permet d'émuler le comportement d'un DMA qui produit des données dans un buffer.

Le son sera lu puis enregistré par le système que vous allez concevoir.

Tout ceci doit bien sûr s'effectuer dans un environnement temps-réel avec l'aide de Xenomai. Le son que vous allez lire est un flux d'échantillons continu qui ne s'interrompt jamais. Même si ce flux d'échantillons est de fait bufferisé en hardware, leur lecture doit se faire sous la contrainte d'une échéance ferme afin d'éviter que certaines données soient perdues en chemin.

⚠ Ce labo est le premier d'une série sur le même thème. Vous aurez donc certainement des éléments à y reprendre pour les prochains labos.

Mise en route de la carte DE1-SoC

Lisez attentivement ce qui suit ! Le laboratoire demande un peu de préparation pour pouvoir mener correctement vos expériences.

Le laboratoire comprend plusieurs fichiers :

1. `/usr/custom_modules/de1_ioctl_module.ko` : un driver Linux sous forme de module permettant de mapper les GPIO de la FPGA connectés aux boutons et aux leds de la DE1-SoC.
2. `/usr/custom_modules/de1_audio_module.ko` : un driver Linux sous forme de module permettant de mapper les zones de la FPGA permettant de contrôler le son.
3. `/usr/custom_modules/de1_video_module.ko` : un driver Linux sous forme de module permettant de mapper le buffer contenant les données RGB pour l'affichage de la vidéo sur la sortie VGA.
4. `main.c` : le programme à modifier.

Pour tester votre programme, lorsque la board vient d'être démarrée, vous devez effectuer un insmod des différents modules à disposition :

- `insmod /usr/custom_modules/de1_ioctl_module.ko`
- `insmod /usr/custom_modules/de1_audio_module.ko`
- `insmod /usr/custom_modules/de1_video_module.ko`

Lorsque vous récupérez le code (git pull), il est nécessaire de récupérer aussi les sources de la librairie Xenomai :

— `git submodule init`

— `git submodule update`

Sur la machine hôte, compilez le programme à l'aide du Makefile fourni puis transférez le binaire sur la De1-SoC. Vous pouvez ensuite l'exécuter pour faire vos tests.

Vue du système

L'acquisition du son et le pilotage des entrées/sorties sont déjà en place. Nous vous mettons à disposition plusieurs interfaces :

1. Un ensemble de méthodes pour accéder au matériel de la carte
   (a) Etat des boutons
   (b) Etat des switches
   (c) Affichage 7 segments
   (d) Leds

2. Une méthode pour accéder aux échantillons sonores récupérés par le driver S'agissant du contrôleur de son, voici ses charactéristiques :
   (a) Echantillonnage stéréo (deux canaux)
   (b) Echantillonnage à 48 KHz
   (c) Buffer de 128 échantillons par canal
   (d) Un échantillon est une donnée sur 16 bits.

   Veuillez noter que les données audio dans le buffer retourné par le driver sont intercalées, alternant entre les canaux gauche ('left') et droit ('right') pour chaque échantillon.

3. Enfin, pour la partie vidéo, vous avez à disposition un fichier contenant les données raw (RGB0). Il n'y a donc aucune compression appliquée. Le fichier s'appelle `output_video.raw`. C'est ce fichier que nous allons utiliser pour simuler un flux de données qui devrait normalement nous être fourni par un DMA, qui serait en mesure de récupérer des données d'un device externe et de les mettre à disposition à l'utilisateur périodiquement.

⚠ Rappel : avant de lancer le code d'exemple, lisez la mise en route de la carte DE1-SoC.

Etape 1 : Audio

Le code qui vous est fourni ne met rien en place, c'est à vous d'utiliser les librairies à disposition pour réaliser le comportement décrit dans cette section.

Pour commencer, nous allons mettre en place le nécessaire pour gérer la partie audio :

✓ 1. Analyse théorique .
   (a) Pour commencer utilisez la description de l'audio fournie précédemment pour déterminer la fréquence de la tâche minimum nécessaire pour effectuer l'acquisition des données sans que celle-ci ne perde de données.
   (b) Prenez connaissance des méthodes à disposition dans les fichiers `audio_utils.c/h`. Les entêtes sont détaillées et vous donnent toutes les informations nécessaires pour la lecture et l'écriture des données audios.

✠ 2. Mettez ensuite en place une tâche Xenomai qui va récupérer les données en entrée, et qui va réécrire ces données sur la sortie. Cette première approche vous permettra de valider votre implémentation et votre analyse théorique.

✠ 3. Une fois cette base en place, ajouter un système qui permet d'écrire les données dans un fichier audio (wav). Le comportement souhaité est le suivant :
   — SW0 DOWN : Le son n'est pas enregistré.
   — SW0 UP : Un fichier *.wav est créé, et les données sont sauvegardées dans ce fichier jusqu'à ce que SW0 passe à DOWN à nouveau.
   — Indépendamment du fichier, le son doit toujours être écrit sur la la sortie audio.
   — Pas de gestion des fichiers demandée. À chaque nouvel enregistrement demandé, le fichier précédent est écrasé.

Etape 2 : Vidéo

Maintenant que l'audio est en place, nous allons récupérer des données RAW RGB0 d'un fichier et remplir un buffer qui se trouve dans la RAM de la FPGA. Une IP permet d'envoyer à 60FPS les données qui se trouvent dans ce buffer sur la sortie VGA, ce qui vous permet de visualiser les données qui se trouvent dans le fichier.

Description du fichier :

— Le format de la vidéo est 320*240
— Chaque pixel est sur 4 bytes avec la configuration suivante (RGB0)
— La vidéo est originale a un framerate de 15 images par secondes.

Travail à faire :

✓ 1. Commencez par analyser les données :
   — Combien de frames contient le fichier ?
   — Quelle est la taille d'une image ?

2. Tout comme pour la partie audio, prenez connaissance de la librairie et des fonctions à disposition pour gérer le buffer (et donc l'affichage).

✠ 3. Mettez ensuite en place une tâche Xenomai avec une période adaptée qui va lire image par image le fichier, puis remplir le buffer pour afficher le contenu sur la sortie VGA.

Etape 3 : Caractérisation de vos tâches

Comme nous avons exploré dans le laboratoire précédent, différentes méthodes permettent de caractériser les performances, notamment le temps d'exécution d'une tâche.

✓ Pour ce laboratoire, je vous demande de mesurer et analyser les performances de vos tâches de traitement audio et vidéo. Veuillez documenter à la fois le temps d'exécution et la distribution du temps pour chaque tâche. Cela nous aidera à évaluer leur comportement (déterministe) sous différentes charges par la suite.

✓ Libre à vous d'utiliser les méthodes de mesure que nous avons utilisées et discutées. Merci aussi d'indiquer si vous voyer de futures optimisations de vos tâches.

Travail à effectuer

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

Un rapport au format PDF vous est également demandé. Il devra contenir les réponses aux questions directes (indiquées par le symbole ✓), vos choix architecturaux ainsi que les tests effectués.

- Rendez le tout sur cyberlearn, dans un fichier compressé nommé `rendu.tar.gz`
  - Ce fichier doit être généré en lançant le script `ptr_rendu.sh`
  - Le script vérifie qu'un fichier `rapport.pdf` est présent à son niveau, ainsi qu'un dossier `code` également présent au même niveau