TP2: Introduction d'un coprocesseur matériel spécialisé

1. 0. Objectif
2. 1. Tâche matérielle
3. 2. Tâche matérielle virtuelle
    1. 2.1. Déploiement
    2. 2.2. Précision temporelle
    3. 2.3. Comparaison des implémentations
4. 3. Compte-Rendu

TP Précédent: MjpegCourse/Multipipe

0. Objectif

L'objectif de ce TP est de vous montrer comment introduire un coprocesseur matériel spécialisé dans une architecture matérielle comportant principalement des processeurs programmables.

Nous allons nous concentrer sur la traduction en matériel de la tâche la plus gourmande en ressources du TCG: la tâche `Idct`.

Nous repartirons de la plateforme du TP3: VgmnNoirqMulti. Sur la base de cette plateforme à deux processeurs Mips, nous ajouterons un composant dédié au traitement de l'`Idct`.

Reprenez les fichiers du TP2:

- La description de la plateforme
- La description de l'application
- Le code des tâches (`Libu` ne gère qu'un seul pipeline et `Split` n'existe pas)

? Pour un déploiement entièrement logiciel, rappelez le temps nécessaire pour décoder 25 images.

1. Tâche matérielle

Maintenant nous allons utiliser un coprocesseur dédié au traitement de l'`Idct`.

Ajoutez une déclaration à la tâche `idct`:

```
idct = TaskModel(
                'idct',
                infifos = [ 'input' ],
                outfifos = [ 'output' ],
                impl = [ SwTask( 'idct',
                                                    stack_size = 1024,
                                                    sources = [ 'src/idct.c' ],
                                                    defines = [ 'WIDTH', 'HEIGHT' ] ),
                                 HwTask( IdctCoproc )
                                 ] )
```

? Quel est le temps de simulation nécessaire pour décoder 25 images ?

2. Tâche matérielle virtuelle

2.1. Déploiement

Ici, la tâche sera virtuellement matérielle dans le sens où nous utiliserons le code C de notre tâche pour implémenter le comportement matériel de l'`Idct`.

Remplacez la déclaration `HwTask( IdctCoproc )` par une déclaration de coprocesseur matériel virtuel `Synthetic()`.

L'implémentation `Synthetic()` doit être accompagnée d'une déclaration de tâche logicielle (`SwTask`) et permet la synthèse virtuelle de la tâche. On peut alors déployer la tâche comme si elle était matérielle, son comportement est simulé.

? Avec cette tâche déployée en coprocesseur virtuel, quel est le temps nécessaire pour décoder 25 images ? Quel rapport à la simulation avec un vrai coprocesseur matériel ?

2.2. Précision temporelle

Pour rendre la tâche matérielle virtuelle plus exacte en temps de simulation, on peut ajouter des directives dans le code source C des tâches pour préciser le temps qu'il faudrait pour réaliser la même action en matériel: `srl_busy_cycles` (voir SrlApi).

En lisant le code de l'implémentation matérielle du coprocesseur `Idct`, déduisez les temps nécessaires aux différentes parties du traitement.

2.3. Comparaison des implémentations

Ajoutez les directives temporelles dans `src/idct.c`.

? Quel est maintenant le temps de simulation nécessaire pour 25 images ?

? Qu'en déduisez-vous sur la différence entre les deux possibilités pour tester une implémentation matérielle ?

? Quel intérêt y a-t-il à pouvoir caractériser précisément le temps de traitement d'une tâche matérielle à partir d'un code en C ?

3. Compte-Rendu

Comme pour les TP précédents, vous rendrez une archive contenant:

```
$ tar tzf binome0_binome1.tar.gz
tp5/
tp5/rapport.pdf
tp5/vgmn_noirq_multi.py
tp5/mjpeg/
tp5/mjpeg/mjpeg.py
tp5/mjpeg/src/
tp5/mjpeg/src/iqzz.c
tp5/mjpeg/src/idct.c
tp5/mjpeg/src/libu.c
```

Cette archive devra être livrée avant le mardi 13 mars 2007, 18h00 à [MailAsim:nipo Nicolas Pouillon]