API pour taches matérielles dans l'architecture OLLAF

Samuel GARCIA & Aravinth GNANASOTHY
CNAM - CEDRIC - LAETITIA
292 rue Saint Martin
Paris - FRANCE
samuel.garcia@cnam.fr

Bertrand GRANADO
UPMC - LIP6 - Syel
4 place Jussieu
Paris - FRANCE
bertrand.granado@lip6.fr

Abstract

Dans cet article nous proposons une méthode pour effectuer des appelle système dans une tache matérielle décrite en VHDL, reposant sur l'architecture OLLAF. Cette méthode consiste à instancier des blocs logiques spéciaux se chargeant de toutes les transactions avec le point d'accès à l'API et offrant un fonctionnement similaire à un bloc classique de fonction similaire. Le développeur d'application peut donc développer son code VHDL comme il en a l'habitude sans se soucier de la complexité du protocole de l'API.

1. Problématique

La problématique est de proposer une façon simple pour effectuer des appels système depuis une tache matérielle décrite en VHDL. Nous nous intéresseront ici plus particulièrement aux accès mémoire.

1.1 Contexte et contraintes

Le contexte de ces travaux est la conception de l'architecture OLLAF. La gestion dynamique des taches matérielle et leur support ont été traité dans de précédents travaux [1]. Nous nous intéressons maintenant au support de fonction systèmes plus liée à l'application. En l'occurrence nous traiterons ici des accès mémoires.

L'architecture en tranche d'OLLAF part du principe que chaque tranche offre le même jeu de services. Ainsi les taches peuvent êtres bougés d'une tranche à l'autre sans aucune modification. Tous les accès au système passent par un port spécial, le point d'accès API [2]. Celui si comporte 32bits de sortie (Xout) 32bits d'entrés (Xins) et deux bits de contrôles (Xack et Xdr). Le fonctionnement de ce port est synchrone avec l'horloge des tranches.

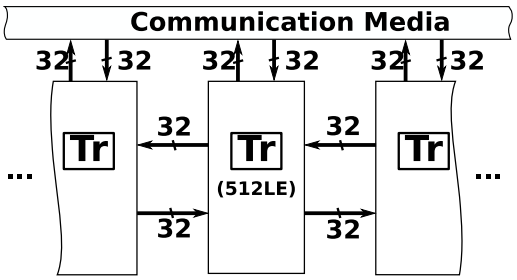

FIGURE **1. External view of a slice in OLLAF**

La dernière contrainte n'est pas des moindre puisqu'elle constitue en soi une grande part de la problématique. Il s'agit de rendre la mise en œuvre d'une application sous forme d'une tache matérielle, au pire, aussi simple qu'une mise en œuvre dans un FPGA classique sans OS. Le véritable but étant au final de rendre la conception plus simple encore en proposant des services de plus haut niveau qu'un simple accès mémoire aléatoire.

2. API Ollaf pour les accès mémoire

La figure 2 montre une vue générale de l'API d'OLLAF. Chaque flèche vers la droite représente un mot binaire à écrire sur le port Xout et chaque fleche vers la gauche représente un mot à lire sur le port Xin.

On peut tel quel utiliser l'API. Mais cela rendrai fastidieux son utilisation. Nous proposons une bibliothèque de blocs logiques (un par fonction) que l'on peut instancier entre le point d'accès de la tranche et le reste de l'architecture de la tâche. Chacun de ces blocs est conçue pour

command block	simple syscall	read std 32 bits access	write std 32 bits access	read burst	write burst
Slice Flip Flops	1	2	2	2	2
4 input LUTs	11	34	34	34	34
Occupied LE	11 (2.1%)	34 (6.7%)	34 (6.7%)	34 (6.7%)	34 (6.7%)

TABLE **1. Implementation results**

avoir un fonctionnement le plus similaire possible à un bloc logique classique de fonction comparable. Le listing si dessous montre l'exemple d'un bloc d'accès mémoire en lecture seul, en comparaison de l'instanciation d'un bloc de mémoire ROM tel qu'on le ferait classiquement lors d'un développement sur FPGA. On voie qu'hormis la connexion au point d'accès de la tranche (qui est toujours identique), la mise en œuvre est tout à fait similaire.

3. Résultats

Le tableau ci-dessous présente des résultats d'implémentation de ces blocs sur un FPGA spartan3. Ces mesures donnent une bonne idée du surcoût induit.

Mentionnons tout de même que le résultat principal de ces travaux doit être vu dans la simplicité de mise en œuvre.

4. Conclusion et Perspectives

Nous avons montrés une façon de mettre en œuvre une API pour tâche matérielle accessible directement en VHDL et utilisant des méthodes déjà connus de tout développeur matériel.

Les perspective sont d'étendre la bibliothèque de bloc API a d'autres services système jusqu'à obtenir un jeu complet de services systèmes courants. Nous espérons prochainement pouvoir fournir un banc de test spécial permettant de simuler une tache matériel sous OLLAF sans avoir besoin de la plateforme elle-même.

(a)

```
in_mem : entity
  OLLAF.sinROM port map(
     data_out => data_sinus,
     address => adr_ROM,
     CE => CE,
     Ack => fir_ce,
  );
```

(b)

```
Commande_lecture_32bits : entity
  OLLAF.read32bit_command port map (
     clock => clock,
     rst  => rst,
     Xack => Xack1,
     Xdr => Xdr1,
     Xin  => Xin1,
     Xout => Xout1,
     --
     data_out => data_sinus,
     address => adr_ROM,
     CE  => CE,
     Ack => fir_ce
  );
```

Listing 1: a) instanciation d'une ROM dans un design FPGA
b) instanciation d'un bloc de lecture mémoire dans une tache OLLAF

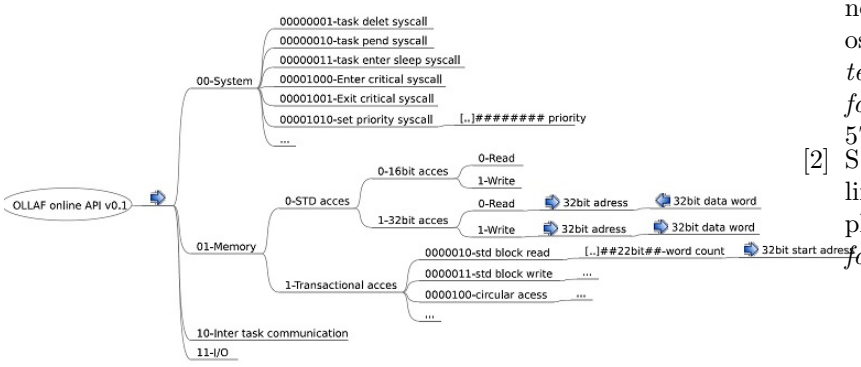

FIGURE **2. Tree view of OLLAF'S API**

Références

[1] S. Garcia and B. Granado. Ollaf : A fine grained dynamically reconfigurable architecture for os support. *EURASIP Jounral on Embedded Systems - Special Issue on design and architectures for signal and image processing*, 2009 :Article ID 574716,2009, 2009.

[2] S. Garcia and B. Granado. Task model and online operating system for hardware tasks in ollaf platform. *Workshop on Design and Architectures for Signal & Image Processing*, 2011.