Validation des applications communicantes embarquées

Télécom

Février 2009

Durée : 3h
Documents : tous documents autorisés

Veuillez respecter les notations introduites dans l'énoncé.
Les parties 1 et 2 sont indépendantes, veuillez les rédiger sur des copies séparées.
Des explications pertinentes en français seront **très** appréciées, et il est indispensable de **justifier** vos réponses.

---

1 - Modélisation transactionnelle et SystemC (10 points)

On considère une application qui utilise notamment deux micro-processeurs. Les deux processeurs n'utilisent pas de mémoire partagée, mais se serviront seulement d'un composant dédié « FIFO » qui permettra d'envoyer des données d'un processeur à l'autre. La spécification de ce composant est fournie en annexe.

Le processeur 1 va envoyer des données à la FIFO jusqu'à ce que celle-ci soit pleine. Lorsque la FIFO est pleine, le processeur se met en attente jusqu'à ce qu'une case se libère dans la FIFO.

De son côté, le processeur 2 va lire des données dans la FIFO jusqu'à ce que celle-ci soit vide, et lorsque c'est le cas, va se mettre en attente jusqu'à ce qu'une nouvelle donnée soit disponible.

Note : chaque processeur dispose d'une unique entrée interruption nommée `irq`, active sur front montant. Les autres entrées/sorties (processeur et mémoire) sont les mêmes que pour les composants vus en TP.

▷ **Question 1 (2 points) :**

En utilisant les conventions graphiques vues en cours, faire un schéma d'ensemble de la plateforme (mettre une légende pour indiquer la signification de chaque symbole). Expliquer comment intégrer le composant FIFO et comment l'utiliser, de façon concise mais précise.

Pour la question suivante, on suppose fournie une classe template FIFO implémentant une file (de taille 1024) en C++:

```
template <typename DATA_TYPE>
class FIFO {
public:
        FIFO();
        /*! Return 0 if put was successful,
          and 1 if the put failed (because of full FIFO) */
        int put(DATA_TYPE d);
        /*! Return 0 if get was successful,
          and 1 if the get failed (because of an empty FIFO) */
        int get(DATA_TYPE & d);
private: // ...
};
```

▷ **Question 2 (5 points) :**

Écrire en SystemC (fichier TLM_FIFO.h et fichier TLM_FIFO.cpp) la classe TLM_FIFO implémentant la spécification donnée en annexe.
Vous utiliserez les mêmes éléments de la bibliothèque TLM que ceux vus en cours et en TPs. On supposera l'existence de deux types `ADDRESS_TYPE` et `DATA_TYPE`, ce dernier étant codé sur 32 bits. Enfin, vous considérerez que les sorties d'interruptions sont à `false` en début de simulation (pas d'initialisation nécessaire).
Lorsqu'une transaction lève une erreur (écriture dans une file pleine, ou lecture dans une file vide), on renverra la valeur tlm::TLM_GENERIC_ERROR_RESPONSE à la place de tlm::TLM_OK_RESPONSE depuis le composant cible, et l'initiateur fera le nécessaire pour continuer la simulation.

On souhaite maintenant écrire en langage C le logiciel embarqué sur le processeur **1**. Pour simplifier, le processeur 1 va simplement envoyer les nombres 0, 1, 2, ..., 4096 au processeur 2 via le composant FIFO (i.e. on va générer les données avec une simple boucle « for »).

On suppose l'existence des fonctions suivantes :

- `void sync_wait()` : attend une interruption (pour simplifier, on suppose ici qu'il n'y a qu'une source d'interruptions)
- `void write_mem(a, d)` : écrit la donnée `d` à l'adresse `a` sur le bus auquel est relié le processeur,
- `int read_mem(a)` : lit une donnée à l'adresse `a` sur le bus auquel est relié le processeur, et la retourne.

Enfin, on supposera que la fonction `int main()` est la fonction principale. Le port esclave (cible) du composant FIFO est à l'adresse `0x1000` sur le bus principal.

▷ **Question 3 (3 points) :**

Expliquez ce que doit faire le processeur 1 pour produire des données en boucle et les communiquer au processeur 2. Ce code doit prendre en compte le cas où la FIFO est vide (et donc l'écriture sur le composant TLM_FIFO renvoie une erreur).
Écrire le logiciel embarqué du processeur 1 : donner le contenu de la fonction principale. Vous introduirez les variables qui vous semblent nécessaires.

Annexe

FIFO
Documentation

Le composant FIFO est un module esclave *Advanced Microcontroller Bus Architecture* (AMBA) destiné à être connecté à un bus haute-performance *Advanced High-performance Bus* (AHB).

Fonctionnalités

Le composant FIFO fournit la logique et les entrées/sorties nécessaires à la synchronisation de deux modules maîtres pour un échange unidirectionnel maître 1 vers maître 2 de données sur 32 bits.

Le composant contient une file (FIFO, pour « First In — First Out ») de taille fixe (1024 cases de 32 bits chacune). Lorsque la file est pleine, une écriture provoque une erreur, et lorsque la file est vide, une lecture provoque une erreur.

Entrées/Sorties

Le composant FIFO possède les entrées/sorties suivantes :

- Interface esclave compatible AMBA
- Sortie d'interruption `int_sender` pour le composant maître 1 (émetteur)
- Sortie d'interruption `int_receiver` pour le composant maître 2 (récepteur)

Fonctionnement interne

Initialement les deux sorties d'interruption du composant FIFO sont à 0 (ou **`false`**). Lors d'une écriture de valeur sur le registre `PUT_GET`, une interruption sur `int_receiver` est émise (passage à 1 ou **`true`**) pour signifier qu'au moins une donnée est disponible dans la FIFO. La sortie d'interruption `int_sender` est réinitialisée (à 0 ou **`false`**) et la valeur est stockée dans la FIFO.

Lors d'une lecture de valeur sur le registre `PUT_GET`, une interruption sur `int_sender` est émise (passage à 1) pour signifier qu'au moins une case libre est disponible dans la FIFO. La sortie

d'interruption `int_receiver` est réinitialisée (à 0) et première valeur contenue dans la FIFO est renvoyée.

Récapitulatif des registres

Adresse relative	Type	Taille	Valeur initiale	Nom	Description
0x00	Lecture/Écriture	32 bits	0x00000000	PUT_GET	Lecture/Écriture dans la file.

Remarque sur le registre PUT_GET

Le registre PUT_GET n'est en réalité pas un registre de données ordinaire. Une écriture déclenche une écriture (action « put ») dans une file FIFO, et une lecture lit (en enlève) la première valeur de la file (action « get »).