INF8500 – Vérification et conception des systèmes embarqués

Automne 2019

Laboratoire #1

Modélisation SystemC

Détails concernant le rapport et date de remise:

Voici le barême pour le laboratoire no 1 :

Barème	
Fonctionnalité et qualité du code	/10
Réponses aux questions:	
Question 1	/4
Question 2	/2
Question 3	/2
Question 4	/2
TOTAL	/20

Question 1

En comparant l'interface entre copro1/display, copro2/display et copro3/display, expliquez :

a) Pourquoi copro1/display a deux signaux de contrôle alors que copro3/display en a un seul.
b) Pourquoi copro1/display a deux signaux de contrôle alors que copro2/display n'en a pas.

Question 2

À partir du point 2) de mes suggestions envoyées hier soir par e-mail, c'est-à-dire :

2) Si vous observez bien le comportement de simple_bus.cpp, vous verrez que lorsque vous écrivez un paquet avec burst_write bloquant, la fonction write de l'interface slave (définie dans coprox_adapt_slave.cpp) sera appelée 7 fois. Une première fois pour le wait state (délai du slave à s'activer) et 6 autre fois pour l'écriture de chaque mot du paquet (on aura alors bel et bien un délai de 7 cycles pour le burst complet). Le problème est que c'est à l'intérieur de simple_bus que ce dernier sait qu'il est rendu au 6ième mot du paquet après avoir complété l'appel à write correspondant (c'est à ce moment que la transaction passe de SIMPLE_BUS_WAIT à SIMPLE_BUS_OK, voir la ligne 327 de simple_bus.cpp et qu'on débloque le maître).

Mais du côté de coprox_adapt_slave, il doit savoir que le 6ième mot a été complétée avant la fin de la fonction write. Vous devez alors aussi connaitre l'adresse de début du paquet (adresse du premier mot du paquet dans MEM), pour ensuite reconstruire le paquet. Pour solutionner ces 2 problèmes, je vous suggère de mettre un compteur dans la fonction write et retenir l'adresse du premier mot (juste après le wait state) dans une variable private de coprox_adapt_slave (p.e. m_current_packet_start_address). Ainsi à la fin du dernier write (6ième mot du paquet) vous saurez qu'on est à écrire le dernier mot et vous pourrez reconstruire le paquet avec l'instruction:

packet = new Packet(&MEM[((m_current_packet_start_address - m_start_address) / 4)]); ou est déjà mémorisée à la construction du slave i.e. 0, 96 ou 192 (voir le point 1).

Ce paquet pourra ensuite être passé au copro pour l'affichage par display.

Expliquez le comportement cycle par cycle de simple bus durant l'appel de *bus_port->burst_write()* (voir fichier : *packet_gen_adapt_master.cpp*) en incluant aussi l'utilisation de la fonction *write()* (*simple_bus_slave_if.h*) définie dans l'adapteur des coprocesseurs. Vous devez décrire les différents appels de fonctions et ce qu'ils font, ainsi que le rôle des variables et structures de données utilisées et relatives aux différentes classes (dans *private*) comme par exemple *m_current_request,* etc.

Question 3

Expliquez le rôle des adaptateurs dans la partie 2 du laboratoire. Par exemple, aurait-on pu faire les appels du maitre directement dans *packet_gen.cpp* ou encore aurait-on pu définir directement la fonction *write()* dans les coprocesseurs? Si oui, alors pourquoi avoir recours à des adaptateurs?

Question 4

Classez et comparez le niveau d'abstraction du module *interconnexion* de la partie 1 (Figure 1) à *simple bus* (Figure 2). Indiquez-les avantages/inconvénients de chacun.

Attention : *Sur la figure 2, la connexion entre Packet_Gen_Adapt et Simple Bus doit être pivoter de 180 degré. Autrement dit l'interface est définie dans simple bus.*

Vous devrez me remettre un fichier .zip contenant un répertoire avec le code et un fichier PDF avec le rapport. Veuillez mettre dans le nom du rapport et de l'archive le texte suivant : «Lab1_INF8500_A19_matricule1_matricule2 ».

Je dois vérifier la procédure pour soumettre à partir de Moodle (je vous reviens là-dessus).

Finalement, vous avez jusqu'au **30 septembre (lundi) minuit** pour me remettre le tout.

Guy Bois, responsable du cours INF8500