BOUSQUET Bruno (1798492)
BOLLARD Guillaume *(2039062)*

INF8500 – Septembre 2019

Laboratoire #1 – Modélisation SystemC

SOMMAIRE

I. Question 1

"copro1/display" utilise 2 signaux de contrôle pour :

- Indiquer à l'interconnexion qu'il est prêt à recevoir le paquet (signal **ack**)
- Savoir lorsque le paquet fourni par interconnexion est valide (signal **ready**)

Ces 2 signaux sont utilisés pour effectuer le protocole "poignée de main" (handshaking).

a) Différence entre copro1/display et copro3/display

Dans la partie "copro3/display", le signal **ready** est remplacé par une fonction **wait** faisant appel à la fonction **value_changed_event** de la classe **buffer**. Elle remplit le même rôle car cela permet de savoir lorsque le buffer est modifié en écriture par interconnexion (c'est à dire lorsqu'un nouveau paquet est envoyé).

Seul le signal de contrôle **ack** est donc encore utilisé dans "copro3/display".

b) Différence entre copro1/display et copro2/display

Dans la partie "copro2/display" :

- Le signal **ack** est supprimé car interconnexion envoie les paquets sur la fifo. Ainsi l'interconnexion peut envoyer plusieurs paquets avant qu'ils soient traités par copro2 et interconnexion n'a pas besoin d'attendre que copro2 soit prêt.
- Le signal **ready** est remplacé par la lecture bloquante dans la fifo. Copro2 lit donc un paquet dès qu'il est valide.

Il n'y a donc pas besoin d'implémenter de signal de contrôle pour "copro2/display".

II. Question 2

Lorsqu'un appel à burst_write() est fait, simple_bus va créer une requête d'écriture. Lors du prochain front descendant de l'horloge, la fonction main_action() sera appelée. S'il n'y a pas de requête en cours, on en sélectionnera une nouvelle, puis handle_request() sera appelée. L'esclave sera sélectionné en fonction de l'adresse, puis un write (ou read) sera effectué, selon la nature de la requête. Au retour de la fonction, on obtient l'état de l'esclave. Dans le cas d'un bus error, on arrête la communication tout de suite. Dans le cas d'un bus wait, l'esclave est occupé, on tentera d'envoyer la même requête ultérieurement. Dans le cas d'un bus ok, on marquera la requête comme terminée, laissant la place à une prochaine requête en attente, s'il y a lieu.

Différents retour possible de la fonction write() par le slave :

- SIMPLE_BUS_ERROR : le traitement des requêtes sera arrêté, même s'il survient au milieu de plusieurs requêtes.

- SIMPLE_BUS_OK : Indication au master que la donnée a été bien reçue via le bus. Le master enverra donc le prochain mot au cycle suivant sauf si on est arrivé à la dernière adresse.
- SIMPLE_BUS_WAIT : La requête n'a pas pu être traitée, le master doit ré-essayer de renvoyer les donnée.

N° cycle	1	2	3	4	5	6	7
Contenu bus (envoyé par le Master)	MOT #1	MOT #1	MOT #2	MOT #3	MOT #4	MOT #5	MOT #6
Réponse Slave	WAIT	OK	OK	OK	OK	OK	OK
Action Slave	Récupération adresse	Ecriture MOT#1	Ecriture MOT#2	Ecriture MOT#3	Ecriture MOT#4	Ecriture MOT#5	Ecriture MOT#6

Echange Master/Slave via SImpleBus durant les 7 cycles de communication

III. Question 3

Les adaptateurs, bien que facultatifs, font partie d'une couche d'abstraction du matériel qui rend sont utilisation plus aisé.
Dans notre laboratoire, nous aurions pu modifier le code du générateur et des copro pour qu'ils puissent interfacer directement avec simple_bus. Par contre, lorsqu'on travaille avec des vrais périphériques matériels, il est souvent impossible de pouvoir changer leur interface. Alors, pour faire l'usage d'un bus, il faut implémenter des interfaces.

IV. Question 4

Dans la partie 2, l'utilisation de SimpleBus permet une communication sur le bus identique entre le master et tous les slaves (copro_adaptx), tandis que dans la partie 1, le master doit implémenter chacun des protocoles utilisés par les slaves lui-même.

	Partie 1	Partie 2
Clock	non	oui
Communication standard	non	oui
Abstraction	faible	haute

Du fait de l'utilisation de la clock, l'implémentation 2 a l'avantage d'être synchronisé, ce qui ressemble plus à une implémentation réelle d'un bus matériel. Cependant son implémentation est plus complexe (augmentation du nombre d'intermédiaires entre le master et chaque copro).