HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Hes·so
Haute Ecole Spécialisée
de Suisse occidentale

CONCEPTION DE SYSTÈME NUMÉRIQUE SUR FPGA
CSF

MÉMOIRE CACHE

João Miguel Domingues Pedrosa
Rick Wertenbroek

23 juin 2016

Table des matières

1 Introduction

Pour ce laboratoire, nous avons dû implémenter une mémoire cache interne à une FPGA. La mémoire cache sert à accélérer les accès en mémoire en allant chercher les données dans une zone mémoire plus petit mais plus proche donc plus rapide. Il faut donc interface, avec un simple bus, les accès entre l'utilisateur (agent) et la mémoire externe (DDR).

2 Mémoire Cache

2.1 Mémoire

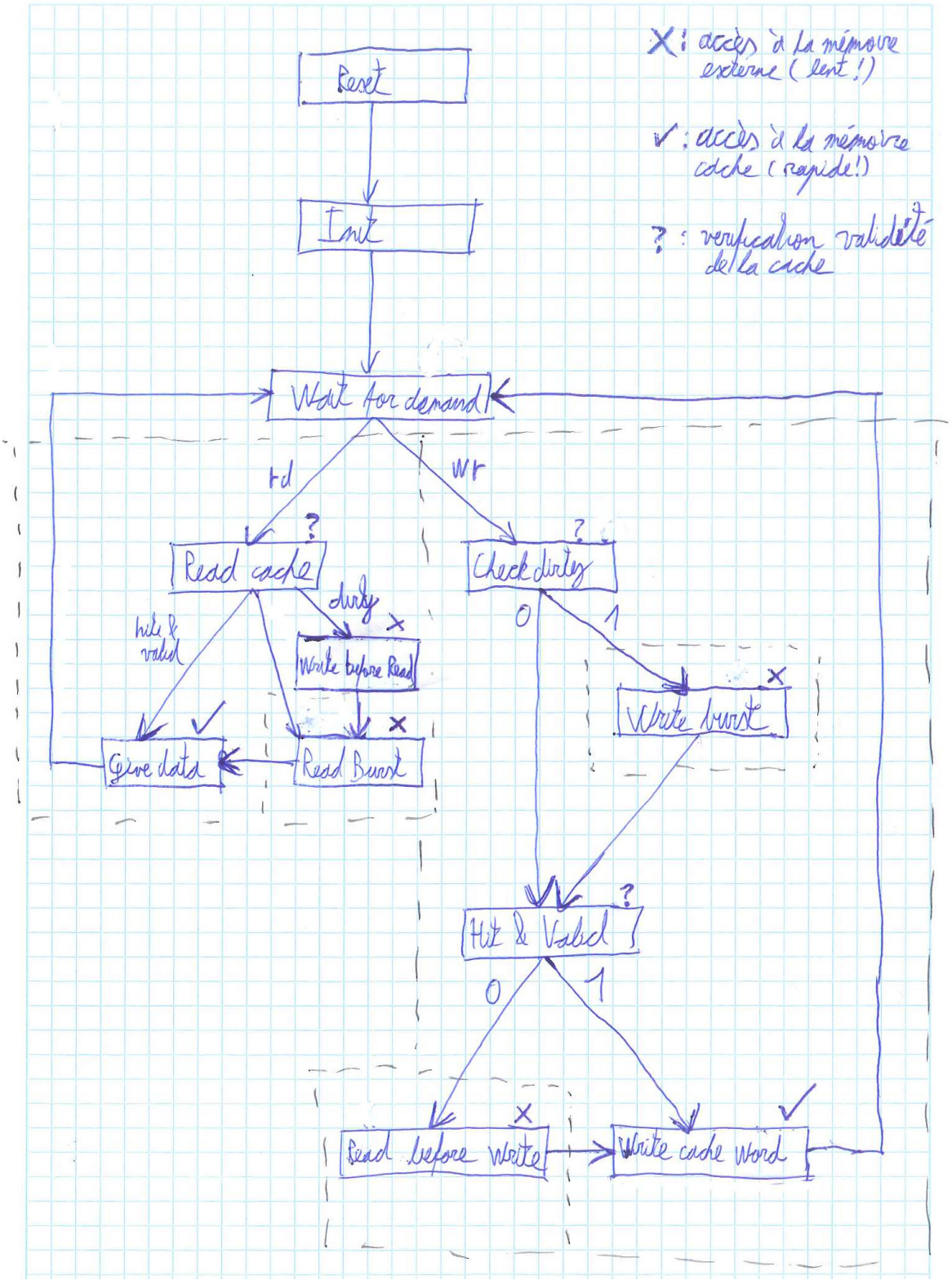
La mémoire se sépare en plusieurs attributs qui sont les suivants :

- Data (vecteur) : il s'agit des données
- Tag (vecteur) : permet de vérifier si le bloque mémoire rechercher est juste
- Valid (bit) : indique si la ligne en cache est initialisé
- Dirty (bit) : indique si il y a eu un accès écriture dans la ligne

Pour chaque attributs, nous avons un tableau de la taille du nombre de ligne en cache que l'on accédera grâce à l'index retrouvé par l'adresse rechercher par l'agent.

2.2 Contrôleur

Pour représenter le fonctionnement de la cache, nous sommes parties d'une machine d'état.

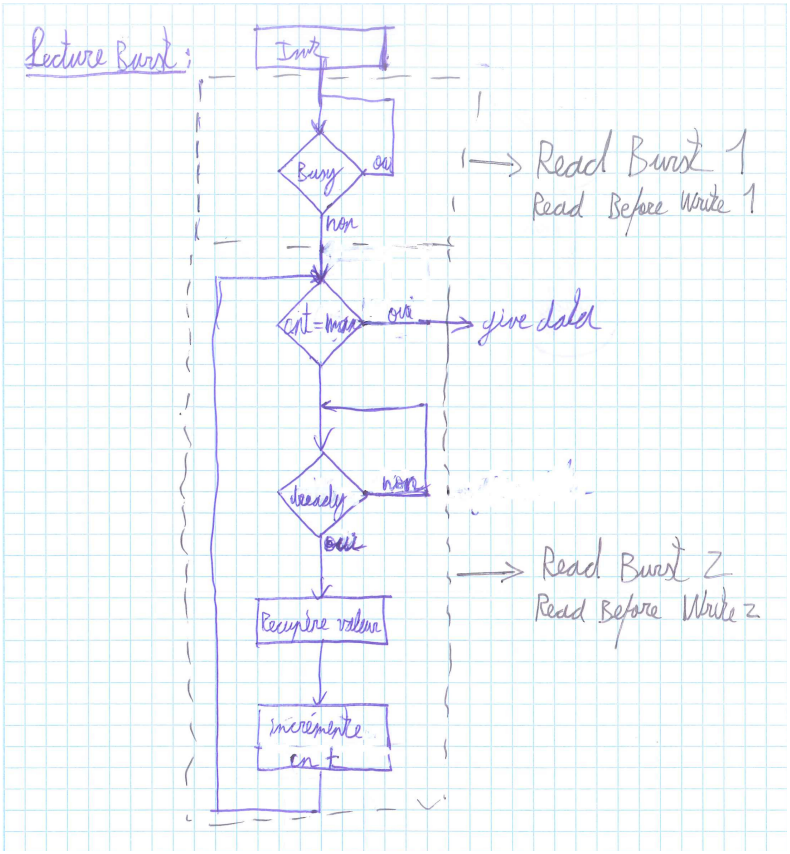


On commence tout d'abord par une initialisation des différents attributs de la cache. Il s'agit notamment de mettre les bits `valid` et `dirty` à 0 pour indiquer que la cache est vide et qu'elle doit donc être rempli.

Ensuite, dans `wait for demand`, on attend un signal de lecture ou écriture (`rd` ou `wr`). On est après redirigé dans la partie concerner.

2.2.1 Lecture

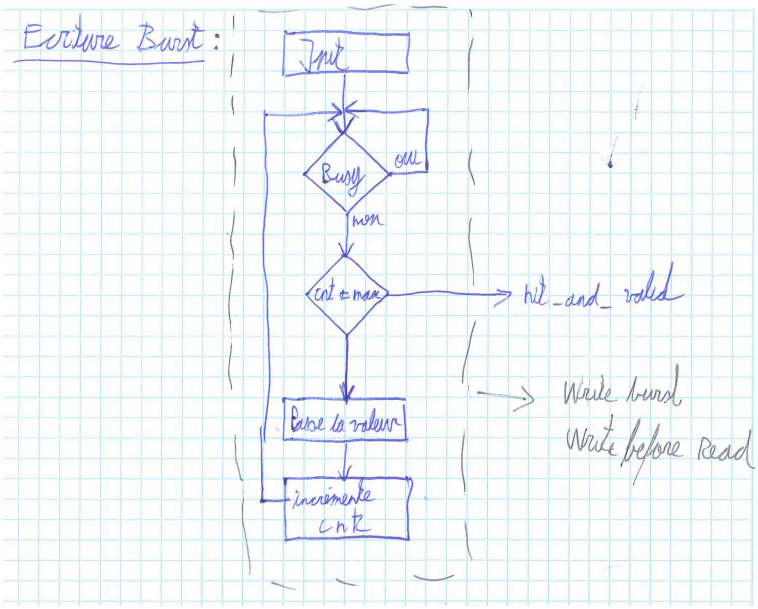
Pour la lecture, on vérifie d'abord s'il y a un hit. Pour cela, on regarde si le tag trouvé via l'adresse correspond à celui en cache et que la ligne est valide. Si la réponse est positive, on récupère directement la valeur dans la mémoire cache sinon, on doit chercher en mémoire d'abord. Avant de mettre à jour la ligne de la cache, on vérifie que la ligne est pas `dirty` si le tag est différent. Cela veut dire qu'il faut d'abord faire une écriture en mémoire avant de lire afin d'éviter les incohérences dans le futur. L'accès en mémoire se fait à chaque fois en burst et de la manière suivante :



On attend tout d'abord sur le busy car il se peut que la mémoire soit occupé lorsqu'on veut l'accéder. Une fois la mémoire libre, on lui demande les données, on vérifie qu'elles soit prêtent et les enregistres. On vérifie que la mémoire à finit le transfère via un compteur que l'on incrémente à chaque fois qu'une donnée a été récupérer avec succès.

2.2.2 Écriture

Pour l'écriture, on doit d'abord vérifier que la ligne à écrire contient le bon tag et dans le cas contraire qu'elle n'est pas `dirty`. Dans le cas où la réponse est négative, il faut écrire dans la mémoire pour la mettre à jour avant de changer le contenu de la cache car le bloque mémoire accédé sera différent. L'écriture se fait en burst et se fait de la manière suivante :



Par rapport à la lecture, on n'attend toujours sur busy avant de passer la valeur.

Avant d'écrire en cache, on doit s'assurer que la ligne est valide. On évite ainsi d'écrire dans une ligne qui n'a pas de valeur cohérente par rapport à la mémoire, ce qui la fausserai plus tard dans l'exécution. Si la ligne est invalide, on fait une lecture en burst dans la mémoire. Une fois ceci fais, on modifie la valeur en cache et on met le bit `dirty` de la ligne à 1.

3 Mémoire simulée

4 Conclusion