

TD3

Implémentation d'une mémoire cache

Côme Allart

Novembre 2025

1 Introduction

L'objectif de ce TP est d'implémenter différents caches afin de manipuler les notions vues en cours. Dans un but pédagogique, le protocole utilisé est très simple. Ce protocole est utilisé pour interagir avec le processeur et la mémoire.

Pour effectuer une transaction, le processeur lève un signal d'activation tout en envoyant l'adresse. Dans le cas d'une écriture, la donnée est également émise. Aucun signal ne doit bouger jusqu'à la réception de la confirmation. Dans le cas d'une lecture, la confirmation sera accompagnée de la donnée lue. Un exemple est montré sur la Figure 1.

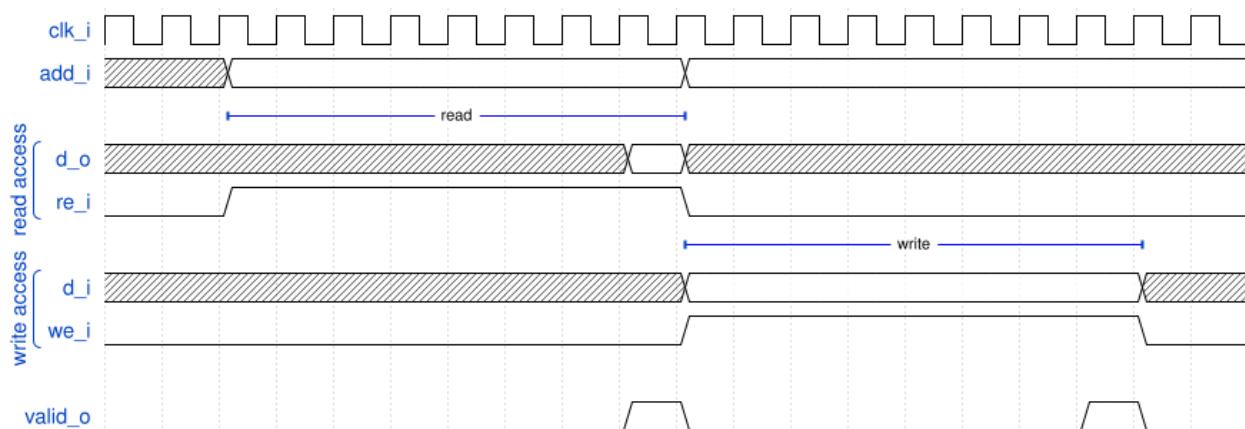


FIGURE 1 – Lecture immédiatement suivie d'une écriture

Ce TP consiste à implémenter un cache instruction. Dans un premier temps nous implémenterons un cache dire puis un cache associatif à deux voies.

2 Cache mémoire "Direct"

Une mémoire cache de type "direct mapped" travaillant sur des adresses 32 bits et construite à l'aide de mémoires ayant un accès par octet a un adressage organisé de la façon suivante :

31 ... 10	9 ... 4	3 ... 0
TAG	INDEX	OFFSET

Quelle est la taille de la ligne de cache, en nombre d'octets, de mots ?

Quelle est le nombre d'entrées de cette mémoire cache ?

3 Performance

Soit le processeur RV32I développé fonctionnant à une période de 100MHz et disposant d'une mémoire cache L1 avec les paramètres suivants :

L1 HIT Time = 1 cycle

MISS penalty = 10 cycles

On constate le taux d'accès suivant :

L1 HIT rate = 90%

Si l'on considère 1000 accès mémoire :

Quel est le taux de MISS ?

Quel est le temps d'accès moyen ?

4 Cache instruction direct

Vous trouverez l'archive du TP directement sur le serveur *tallinn.emse.fr* sous */tmp/ap2_tp3.tgz* Le contenu de l'archive est le suivant :

```
exo3/
|-- firmware
|   |-- build.sh
|   |-- exo3.S
|   '-- firmware_riscv.ld
|-- hdl_src
|   |-- cache_template.sv
|   |-- RV32i_soc.sv
|   '-- wsync_mem_o128.sv
|-- sim
|   '-- build.sh
`-- tb
    '-- RV32i_tb.sv
```

Copier le module vide *cache_template.sv* fourni avec ce TD en *direct_cache.sv*.

Vous allez implémenter votre cache dans ce fichier.

Conservez bien le module vide *cache_template.sv* comme sauvegarde.

L'ensemble de ces fichiers sont à copier dans l'arborescence des td précédents. :

- Les fichiers sources SystemVerilog *.sv sont à placer dans le répertoires *hdl_src*.
- Le fichier *build.sh* dans le répertoire *sim*.

Q0

Pour bien appréhender le sujet établir un schéma du RV32i_soc faisant apparaître :

- RV32i_core
- cache_instruction
- imem
- les interconnexions entre le *cache_instruction*

Quel est la largeur du bus de données entre *imem* et le *cache_instruction* ?

Nous allons créer un cache direct en lecture seule pour les instructions.

Q1

Reprenez les caractéristiques du cache relatifs à la section 2.
Saisissez les caractéristiques du cache dans les '*localparam*'.
La mémoire du cache sera modélisée avec des registres.

Q2

Quelles données est-il nécessaire de stocker ? Précisez la taille de chaque donnée.
Pensez à gérer la réinitialisation : une ligne n'est pas valide avant son premier remplissage.

Q3

Créez les registres pour stocker les données que vous avez listées à la question précédente.

Q4

Implémentez le découpage de l'adresse reçue pour obtenir les trois champs '*tag*', '*index*' et '*offset*'.

Q5

Créez un signal interne qui indique si la requête est '*hit*' ou '*miss*'.
Si la requête est '*hit*', alors le mot lu dans le cache est renvoyé au processeur.

Q6

Assignez les sorties de réponse au processeur pour traiter le cas '*hit*'.
Si la requête est '*miss*', alors il faut envoyer une requête à la mémoire.

Q7

Assignez les sorties de requête à la mémoire.
N'oubliez pas que l'accès à la mémoire doit être aligné sur la taille d'une ligne ; et qu'il ne faut pas répondre au processeur tout de suite.
Lorsque la mémoire répond, il faut transmettre sa réponse au processeur tout en stockant la valeur.

Q8

Ajustez les affectations des sorties de réponse au processeur pour transmettre le résultat de la mémoire.
N'oubliez pas qu'il faut choisir le mot dans la ligne.

Q9

Implémentez le stockage de cette valeur dans la mémoire du cache.
Pensez à affecter la valeur $1'b1$ au bit de validité et à mettre à jour le '*tag*' stocké.

Q10

Démontrez le bon fonctionnement de votre cache en exécutant le programme "Mult.S".

5 Cache instruction associatif à deux voies

Créez une sauvegarde de votre cache précédent *direct_cache.sv*.

Cette nouvelle partie consiste à le rendre associatif, avec une priorité de type LRU. Pour simplifier l'implémentation, le cache sera associatif à deux voies et nous ne chercherons pas à implémenter un cache avec une associativité générique.

Il faut stocker de nouvelles données dans le cache :

- la deuxième voie du cache
- le numéro (indice) de la voie LRU pour chaque ligne du cache.

Q11

Ajoutez ces nouveaux registres.

Pour la deuxième voie du cache, vous pouvez utiliser un tableau à plusieurs dimensions.

Q12

Mettez à jour la condition pour qu'une requête soit un '*hit*'.

Vous pouvez créer un signal de '*hit*' par voie.

Q13

Dans le cas d'un '*hit*', déterminez quelle est la voie à lire.

Q14

Dans le cas d'un '*miss*', déterminez quelle est la voie à évicter.

Q15

Mettez à jour la valeur du LRU correspondant (rappel : LRU = **Least recently used**)

Q16

Démontrez le bon fonctionnement de votre cache.