MICRO ARCHITECTURE D'UN CONTRÔLEUR DE CACHE L1

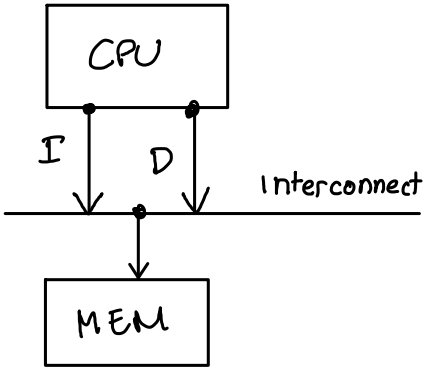
PLAN

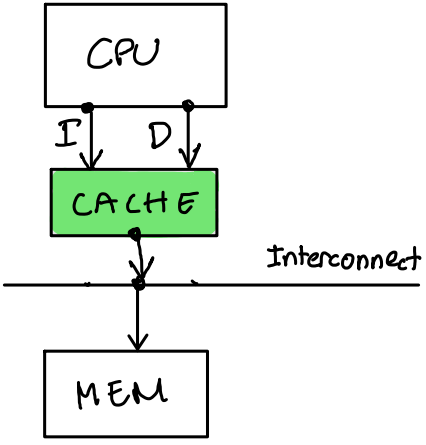
A) PRESENTATION
- RÔLE
- INTERFACE
- CARACTÉRISTIQUES
- TRANSACTIONS

B) AUTOMATES DU CACHE
- ICACHE
- DCACHE
- CMD
- RSP

C) WRITE BUFFER
- RÔLE
- SIMPLE
- MULTI

RÔLE DU CACHE





Le CPU fait
- une lecture I par cycle
- une lecture D ts les 5 cycles
- une écriture D ts les 10 cycles

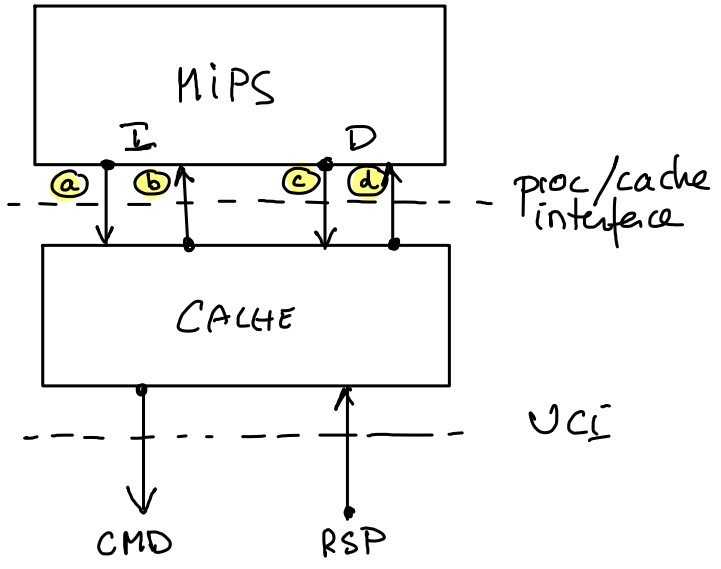
Sans CACHE

☹ utilise beaucoup L'interconnect & la mémoire
Si la latence est grande ⇒ CPI est grand

Avec CACHE

☺ réduction du nombre de transactions
réduction de la latence ⇒ réduit le CPI

INTERFACE DU CACHE



ⓐ
IREQ ADDR	30 bits
IREQ VALID	1 bits
IREQ MODE	1 bits (USER/KERNEL)

ⓑ
IRSP INS	32 bits
IRSP VAL	1 bit
IRSP ERROR	2 bits

transaction OK . BUS ERROR (adr non mappée)
KO . SEG FAULT (segment non mappé)

ⓒ
DREQ ADDR	32 bits
DREQ VALID	1 bits
DREQ MODE	1 bits
DREQ BE	4 bits
DREQ TYPE	4 bits

2 bits R/W
2 bits autres espaces d'adressage

DREQ WDATA	32 bits

ⓓ
DRSP DATA	32 bits
DRSP VAL	1 bits
DRSP ERROR	1 bits

CARACTÉRISTIQUES (1)

- 2 caches séparés instructions et données
- 1 seul port VCI
- bloquant pour les lectures → le risc ne gère que 1 instruction à la fois
- non bloquant pour les écritures
  - → write buffer permet de poster les écritures sauf si le write buffer est plein
- traite plusieurs transactions VCI simultanées
  - → permet d'augmenter le débit puisque latence pipeline
  - → impose 2 automates VCI CMD & RSP

CARACTÉRISTIQUES (2)

- Pas de MMU
  - Si UADDR = adresse des programme (virtuel)
    PADDR = adresse de la mémoire (physique)

  Sans MMU : PADDR = UADDR

  avec MMU : PADDR = TABLE [UADDR, PID]

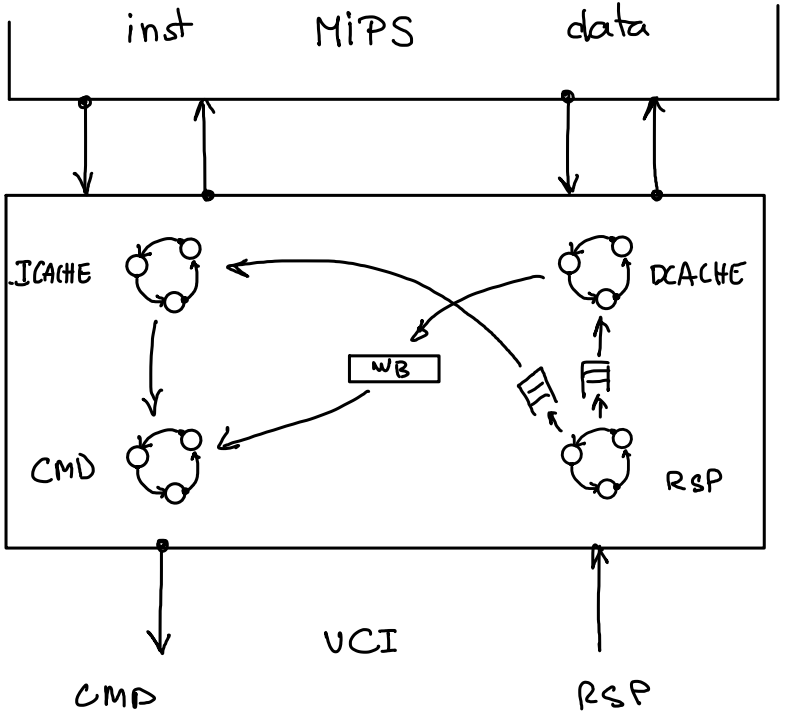
  ↑ traduction ↑ processus

- PAS de cohérence
  - Si 2 caches L1 lisent la même ligne et que l'on la modifie alors l'autre n'est pas mise à jour

TRANSACTIONS VCI

- MISS INSTRUCTION — lecture mémoire
- MISS DATA — lecture mémoire
- READ DATA uncached — lecture périphérique
- READ INSTRUCTION uncached — lecture mémoire (cache désactivé)
- WRITE DATA — ecriture depuis le write buffer

4 AUTOMATES

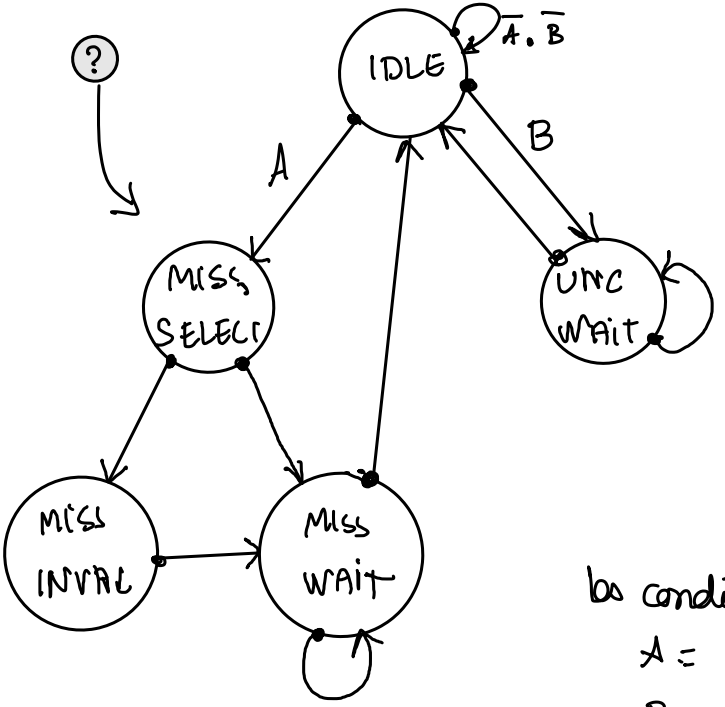


Pourquoi 4 ?

=> pour pouvoir paralléliser
- on a 2 caches => 2 FSM
- CMD + RSP pour pipeliner

ICACHE FSM

Gère les demandes de lecture des MIPS



IDLE : état d'attente d'une lecture avec MIPS
Cet automate est un automate de Mealy.
pour traiter le HIT

MISS_SELECT : choix d'un victime
MISS-INVAL : si la victime est utilisée
MISS_WAIT : Ecriture des mots reçus dans le slot
UNC_WAIT : attente du mot non caché

Les conditions de sortie de l'état IDLE

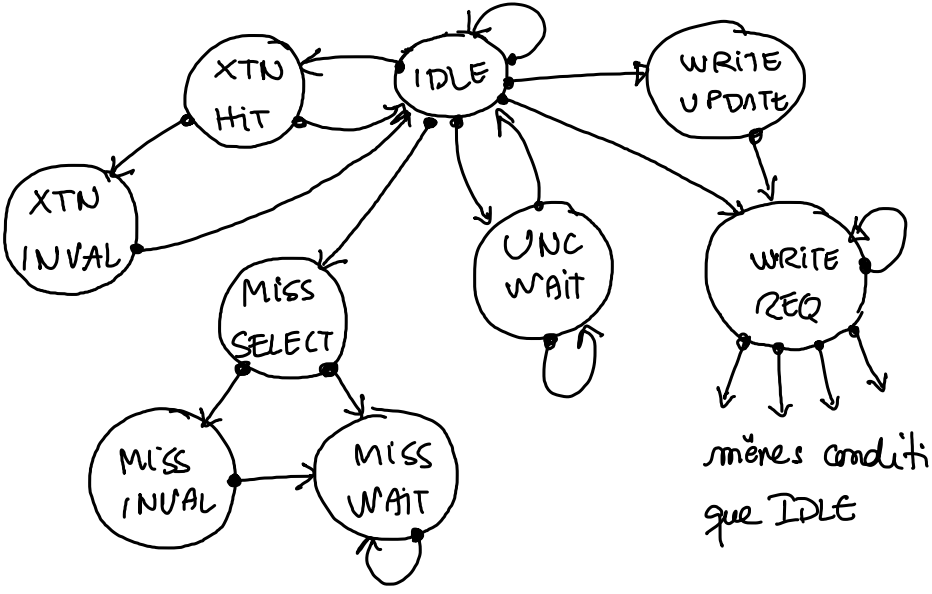
$A = \text{IREQ\_VALID} \cdot \text{ICACHE\_CACHABLE} \cdot \overline{\text{ICACHE\_HIT}}$

$B = \text{IREQ\_VALID} \cdot \overline{\text{ICACHE\_CACHABLE}}$

Vous allez voir les autres transitions dans le code...

DCACHE FSM

gère les demandes de lecture et d'écriture du MIPS + inval



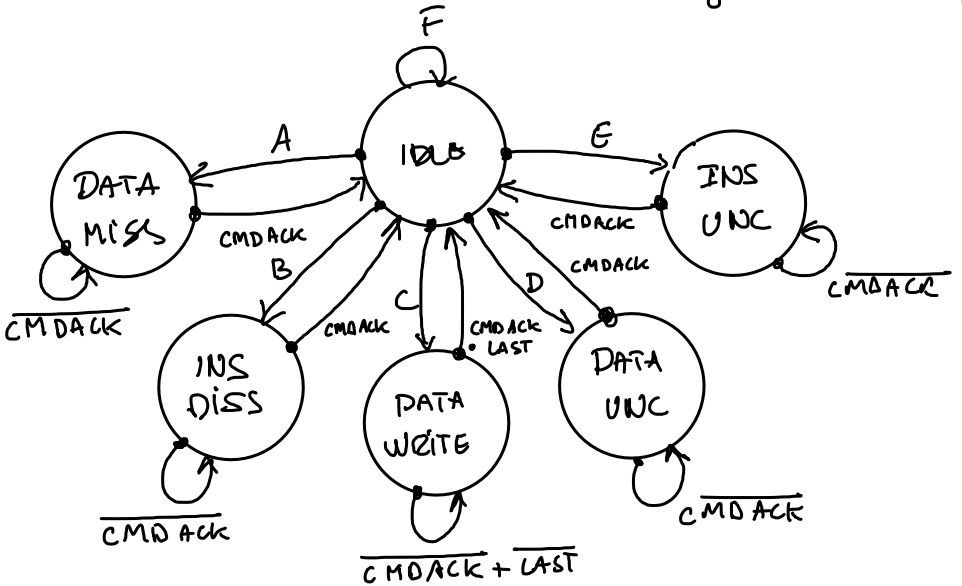
mêmes conditions que IDLE

IDLE
MISS_SELECT
MISS_INVAL
MISS_WAIT
UNC_WAIT
} comme icache_fsm

WRITE_UPDATE =
WRITE_REQ =
XTN_HIT =
XTN_INVAL =
} correspond à l'invalidation d'1 ligne

CMD FSM

gère les commandes VCI
et l'usage de write Buffer



C'est un serveur dont les clients sont les automates du cache qui ont besoin de faire des commandes VCI

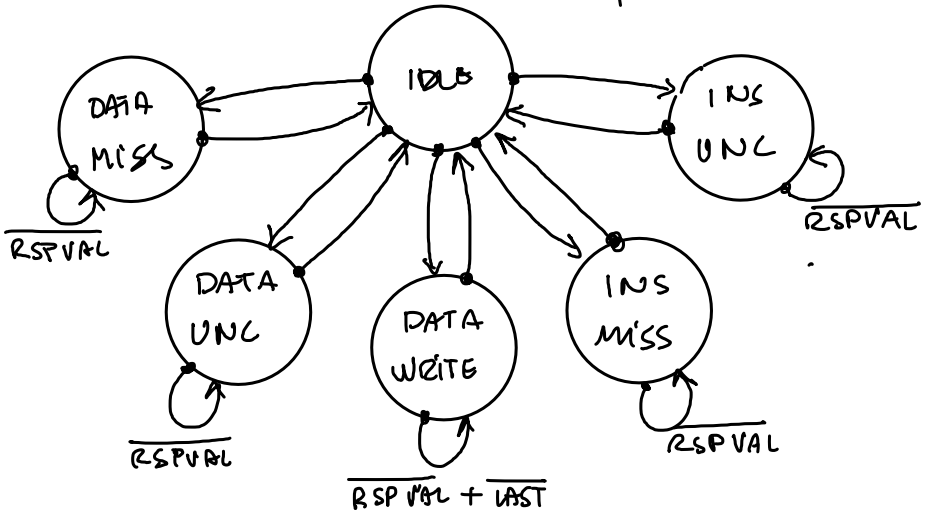
CLIENTS

DMISS INSMISS WRITE DUNC IUNC

les instructions sont prioritaires
et pour les données les écritures sont prioritaires

RSP FSM

gère les réponses VCI
pour le contrôleur ICACHE & DCACHE



Il faut la synchroniser
pour la terminaison & les erreurs
⇒ utilisation de bascules RS.

WRITE BUFFER SIMPLE

- Capacité 1 ligne de cache (+BE)
- 3 états définis par 2 bits :
- Les requêtes sont acceptées dans le write buffer . Si vide ou si même ligne

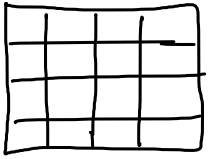
état	r_empty	r_req	
EMPTY	1	Ø	wbuf vide
OPEN	Ø	Ø	wbuf en cours de remplissage
LOCKED	Ø	1	wbuf en cours d'écriture sur VCi

X = CMD_DATA_WRITE . CMD_ACK . LAST
le dernier flit d'un burst a été accepté

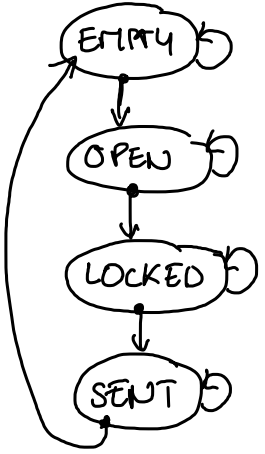
Z = DCACHE_WRITE_REQ . $\overline{WOK}$
Le write buffer prend une écriture

Y = DCACHE_WRITE_REQ . ($\overrightarrow{DREQ\_WRITE}$ + $\overline{WOK}$)
pas d'écriture ou refus du Wbuf

WRITE BUFFER MULTI



- capacité N lignes de buffer
  1 ligne de buffer = 1, 1/2, 1/4 ligne de cache
- requête écriture acceptée si un slot est EMPTY ou OPEN & ∈ buf line
- 4 états pour chaque slot



écriture d'1 mot (DCACHE_FSM)

slot pointed by r_ptr (WBUR_FSM)

commande VCI envoyée (CMD FSM)

Réponse VCI reçue (RSP_FSM)

Il existe un pointeur circulaire incrémenté à chaque cycle rptr qui contrôle la transition OPEN → LOCKED

TME

- analyser le comportement temporel du cache
- analyser la modélisation des automates
- remplacer le write buffer simple par un write buffer MULTI