
Caches multiprocesseurs

Daniel Etiemble
de@lri.fr

Caches et cohérence des caches

- Rôle clé des caches
 - Réduisent le temps d'accès moyen aux données
 - Réduisent les besoins en bande passante sur l'interconnexion partagée.
- Les caches privés des processeurs posent problème
 - Les copies d'une variable peuvent être présentes dans plusieurs caches
 - Une écriture par un processeur peut ne pas être visible aux autres
 - Problème de la cohérence des caches
 - Actions nécessaires pour assurer la visibilité

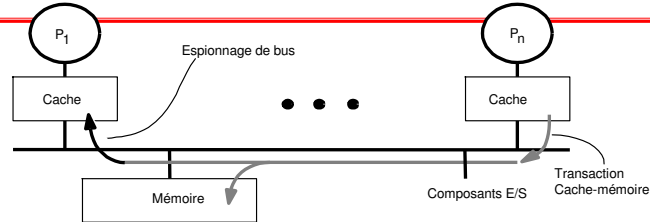
Cohérence des caches avec un bus

- Construite au dessus de deux fondements des systèmes monoprocesseurs
 - Les transactions de bus
 - Le diagramme de transitions d'états des caches
- La transaction de bus monoprocesseur
 - Trois phases : arbitrage, commande/adresse, transfert de données
 - Tous les composants observent les adresses ; un seul maître du bus
- Les états du cache monoprocesseur
 - Ecriture simultanée sans allocation d'écriture
 - Deux états : valide et invalide
 - Réécriture
 - Trois états : valide, invalide, modifié
- Extension aux multiprocesseurs pour implémenter la cohérence

La cohérence par espionnage de bus

- *Idée de base*
 - Les transactions bus sont visibles par tous les processeurs.
 - Les processeurs peuvent observer le bus et effectuer les actions nécessaires sur les événements importants (changer l'état)
- *Implémenter un protocole*
 - Le contrôleur de cache reçoit des entrées de deux côtés :
 - Requêtes venant du processeur, requêtes/réponses bus depuis l'espion
 - Dans chaque cas, effectue les actions nécessaires
 - Mise à jour des états, fourniture des données, génération de nouvelles transactions bus
 - Le protocole est un algorithme distribué : machines d'états coopérantes.
 - Ensemble d'états, diagramme de transitions, actions
 - La granularité de la cohérence est typiquement le bloc de cache

Cohérence avec cache à écriture simultanée

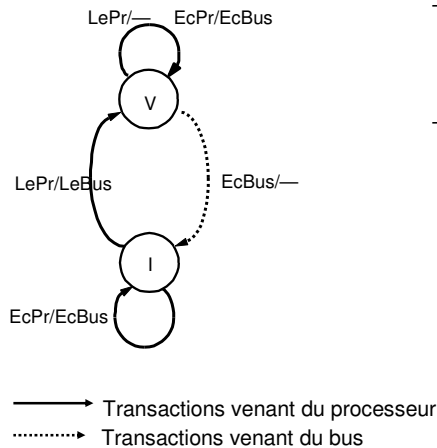


- Extension simple du monoprocesseur : les caches à espionnage avec invalidation ou propagation d'écriture
 - Pas de nouveaux états ou de transactions bus
 - Protocoles à diffusion d'écriture
- Propagation des écritures
 - la mémoire est à jour (écriture simultanée)
- Bande passante élevée nécessaire

Les actions du cache à écriture simultanée (monoprocesseur)

<i>Etat présent</i>	<i>Succès lecture</i>	<i>Succès écriture</i>	<i>Echec lecture</i>	<i>Echec écriture</i>	
Invalide	----	----	Valide	Valide	<i>Etat futur</i>
	----	----	Echec lecture sur Bus	Echec écriture sur Bus	<i>Action</i>
Valide	Valide	Valide	Valide	Valide	<i>Etat futur</i>
	CPU lit le cache	<ul style="list-style-type: none"> • Ecriture bus • CPU écrit dans le cache 	Echec lecture sur Bus	Echec écriture sur Bus	<i>Action</i>

Le diagramme de transition (écriture simultanée non allouée)



- Deux états par bloc de cache
 - L'état d'un bloc peut être considéré comme un vecteur de p éléments.
- Les bits d'état sont associés aux seuls blocs présents dans le cache
 - Les autres blocs sont considérés invalides (non présents)
 - L'écriture va invalider tous les autres caches
 - Il peut y avoir plusieurs blocs lus simultanément, mais l'écriture les invalide

Les actions du cache à réécriture (monoprocesseur)

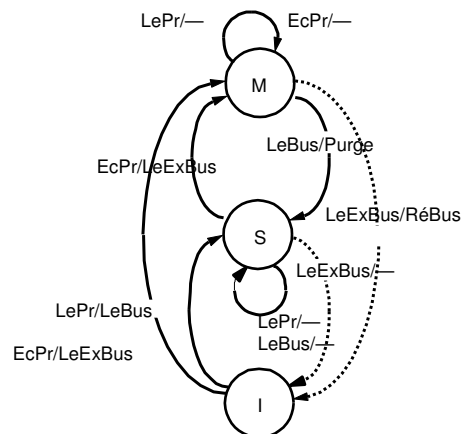
<i>Etat présent</i>	<i>Succès lecture</i>	<i>Succès écriture</i>	<i>Echec lecture</i>	<i>Echec écriture</i>	
Invalide	-----	-----	Valide	Modifié	<i>Etat futur</i>
			Echec lecture sur Bus	Echec écriture sur bus	<i>Action CPU</i>
Valide	Valide	Modifié	Valide	Modifié	<i>Etat futur</i>
	CPU lit le cache	CPU écrit dans le cache	Echec lecture sur Bus	Echec écriture sur bus	<i>Action CPU</i>
Modifié	Modifié	Modifié	Valide	Modifié	<i>Etat futur</i>
	CPU lit le cache	CPU écrit dans le cache	Bloc éjecté et copié en MP Echec lecture sur Bus	Bloc éjecté et copié en MP Echec écriture sur bus	<i>Action CPU</i>

Le protocole de base MSI : réécriture et invalidation

- Etats
 - Invalide (I)
 - Partagé (S): un ou plusieurs
 - Modifié (M): un seul
- Les évènements processeurs :
 - LePr (lecture)
 - EcPr (écriture)
- Les transactions bus
 - LeBus: demande une copie sans vouloir la modifier
 - LeExBus: demande une copie pour la modifier
 - RéBus : Mise à jour de la mémoire
- Actions
 - Modifie l'état, effectue une transaction bus, envoie les données sur le bus

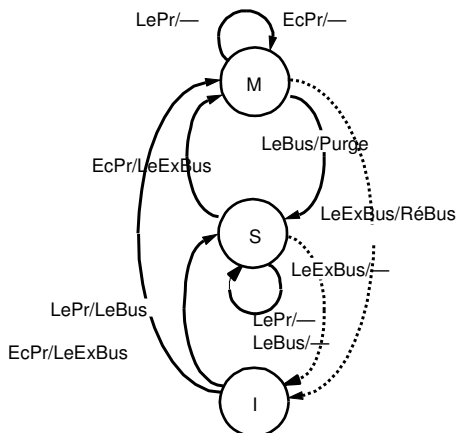
MSI : Succès local lecture

- Le bloc doit être dans l'un des états M ou S
- C'est la valeur correcte (si M, elle a été modifiée par une écriture locale)
- Renvoie la valeur au CPU
- Pas de changement d'état



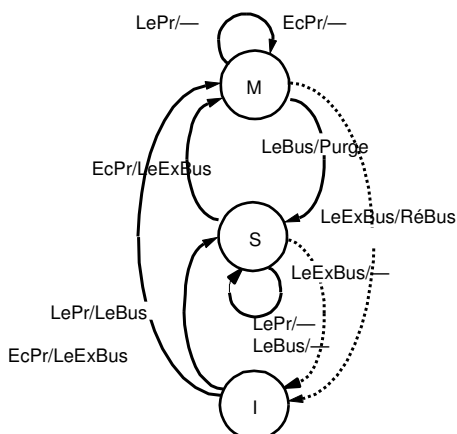
Défaut local en lecture

- Plusieurs caches ont une copie S
 - Le processeur fait une requête bus vers la mémoire
 - Un cache met une copie sur le bus
 - L'accès mémoire est abandonné
 - Le cache local obtient la copie (S). Les autres copies restent (S)
- Un cache a une copie M
 - Le processeur fait une requête bus vers la mémoire
 - Un cache met la copie M sur le bus
 - L'accès mémoire est abandonné
 - Le cache local obtient la copie (S).
 - Le bloc M est écrit en mémoire.
 - Le bloc M passe à S.



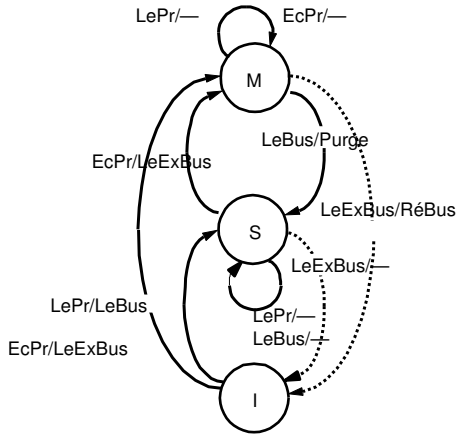
MSI : succès local en écriture

- Les blocs doivent être Mou S
- M
 - le bloc est déjà modifiée
 - mettre à jour la valeur locale
 - Pas de changement d'état
- S
 - Le processeur diffuse sur le bus LeExBus
 - L'état du bloc local passe de S à M
 - Les caches avec une copie S passe à I
 - Le bloc local est mis à jour



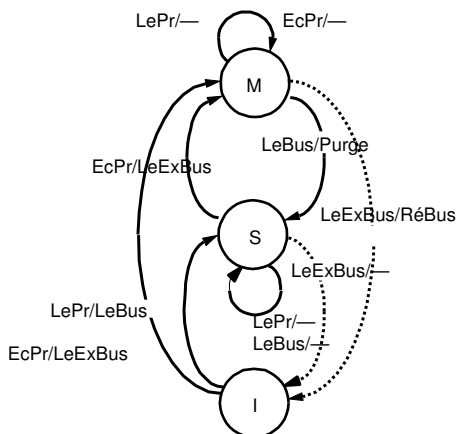
MSI : Echec local en écriture

- Pas d'autres copies
 - Valeur lue depuis la mémoire vers cache local
 - Etat bloc local à M
- D'autres copies à S
 - Valeur lue depuis la mémoire vers cache local (LeExBus)
 - Les autres caches mettent leur copie à I
 - La copie locale est mise à jour et état M
- Une autre copie M
 - Le processeur local envoie LeExBus
 - Le cache avec copie M la voit, bloque LeExBus, prend contrôle du bus et écrit sa copie en mémoire. Le bloc M et les blocs S passent en I.
 - Le processeur local réémet LeExbus : c'est maintenant le cas sans copie
 - Valeur lue depuis la mémoire vers cache local
 - Etat bloc local à M



Le protocole MSI à invalidation : Diagramme de transitions

- Ecriture à un bloc modifié
 - A déjà la valeur la plus récente ; peut utiliser la mise à jour (BusUpgr) au lieu de la LeExBus
- Un remplacement change l'état de deux blocs : le bloc remplacé et le bloc arrivant



Comparaison avec un seul cache à réécriture

- Similarités
 - Succès lecture invisible sur le bus
 - Tous les échecs sont visibles sur le bus
- Différences
 - Avec un seul cache WB, tous les blocs provoquant un échec sont fournis par la MP. Dans le protocole à trois états, les blocs sont fournis soit par la MP ou par le seul bloc de cache contenant la seule copie *Modifié*
 - Avec un seul cache WB, un échec écriture est invisible sur le bus. Dans le protocole à trois états, un succès en écriture sur un bloc *Valide* invalide tous les autres blocs Valide par un Echec écriture Bus (action nécessaire)

Validation du protocole à trois états

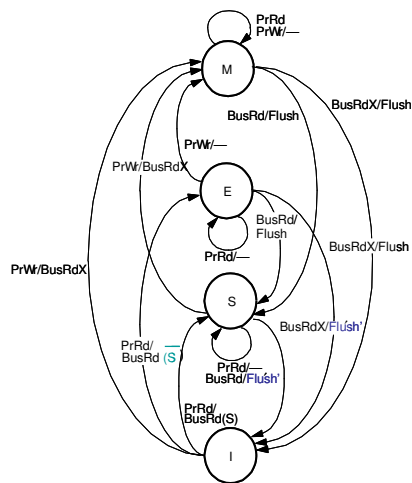
- **Problème** : la transition d'état d'un automate est supposée être atomique, mais ce n'est pas le cas dans ce protocole à cause du bus
- Exemple : Echec lecture CPU sur un bloc Modifié
 - L'accès CPU au cache détecte l'échec
 - Requête bus
 - Acquisition du bus, et changement de l'état du bloc de cache
 - Eviction du bloc du cache et copie en MP
 - Positionnement de l'Echec lecture bus sur le bus
 - Réception du bloc demandé depuis la MP ou un autre cache
 - Libération du bus, et lecture à partir du bloc de cache que l'on vient de recevoir
- L'arbitrage du bus peut provoquer un écart entre les étapes 2 et 3
 - Toute la séquence n'est plus atomique.
 - Le protocole fonctionne correctement si les étapes 3 à 7 sont atomiques, c'est à dire si l'on n'a pas un bus à transactions éclatées

Le protocole MESI

- Quatre états, à invalidation d'écriture
- Version améliorée du protocole à 3 états
 - L'état Valide est séparé en Exclusif et Modifié
 - Exclusif : seule copie – identique au bloc en MP
 - Partagé : plusieurs copies – identique au bloc en MP
- Différentes versions légèrement différentes du protocole MESI
 - Le protocole MESI du PowerPC 601 ne supporte pas les transferts de bloc de cache à cache.

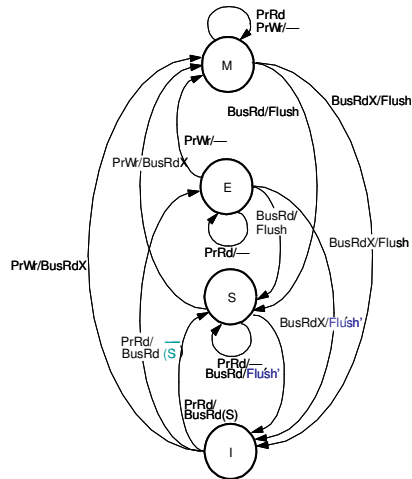
Le protocole MESI

- 4 états
 - Exclusif
 - Seule copie du bloc
 - Non modifié (= Mémoire)
 - Modifié
 - Modifié (!= Mémoire)
 - Partagé
 - Copie dans plusieurs blocs
 - Identique en mémoire
 - Invalide
- Actions
 - Processeurs
 - Bus



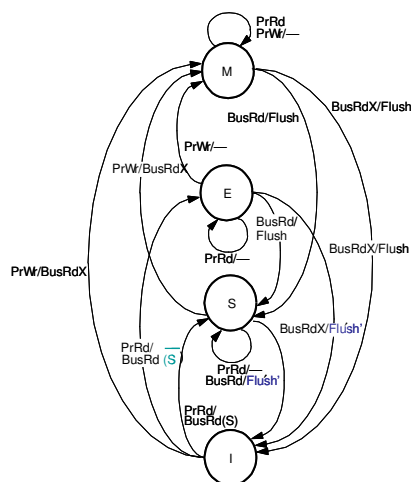
MESI – Succès local lecture

- Le bloc doit être dans l'un des états M, E ou S
- C'est la valeur correcte (si M, elle a été modifiée par une écriture locale)
- Renvoie la valeur
- Pas de changement d'état



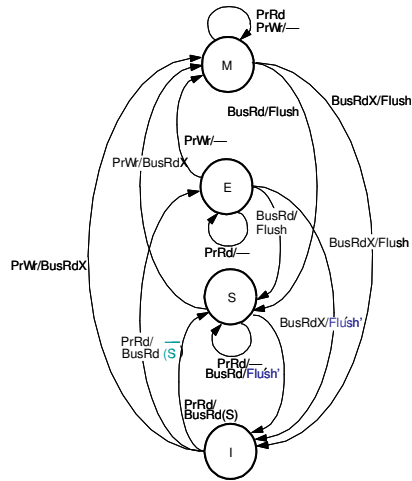
MESI – Défaut local en lecture (1)

- Pas d'autres copies dans un cache
 - Le processeur fait une requête bus vers la mémoire
 - Valeur lue dans le cache local (E)
- Un cache a une copie E
 - Le processeur fait une requête bus vers la mémoire
 - La copie E est mise sur le bus.
 - La requête mémoire est stoppée
 - Le cache local récupère la valeur
 - Les deux blocs passent à S



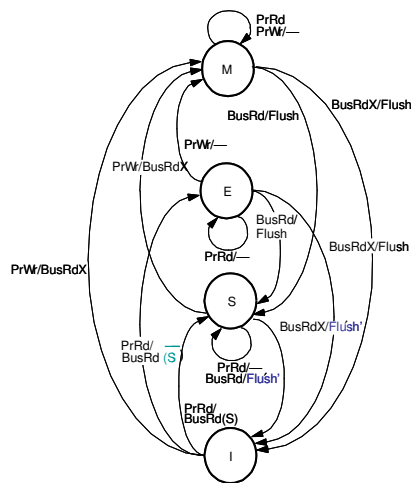
Echec local en écriture

- Pas d'autres copies
 - Valeur lue depuis la mémoire vers cache local
 - Etat bloc local à M
- D'autres copies, soit E (1) soit S (n)
 - Valeur lue depuis la mémoire vers cache local
 - Transaction bus RWITM (lecture avec intention de modifier)
 - Les autres caches mettent leur copie à I
 - La copie locale est mise à jour et état M



Echec local en écriture (2)

- Une autre copie M
 - Le processeur local envoie une Transaction bus RWITM (lecture avec intention de modifier)
 - Le cache avec copie M la voit, bloque RWITM, prend contrôle du bus et écrit sa copie en mémoire. Le bloc passe en I.
 - Le processeur local réémet RWITM : c'est maintenant le cas sans copie
 - Valeur lue depuis la mémoire vers cache local
 - Etat bloc local à M



Le problème du faux partage

- Un processeur écrit dans une partie d'une ligne de cache.
- Un autre processeur écrit dans une autre partie d'une ligne de cache
- Même si chaque processeur ne modifie que sa « partie » de la ligne de cache, toute écriture dans un cache provoque une invalidation de « toute » la ligne de cache dans les autres caches.

Alternatives aux caches

- Approche des processeurs vectoriels
 - Chargement/rangement de vecteurs dans des registres vectoriels
 - Utilisation de mémoires SRAM multi-bancs
- Approche « mémoire scratch pad » (processeurs embarqués)
 - Zone mémoire contrôlée par logiciel
 - Préchargement par logiciel des données nécessaires
 - Instructions de manipulation de blocs