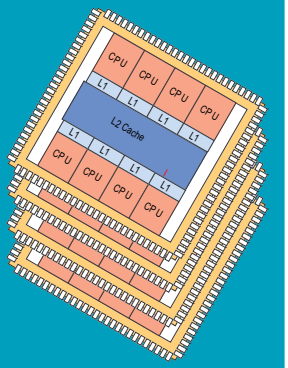
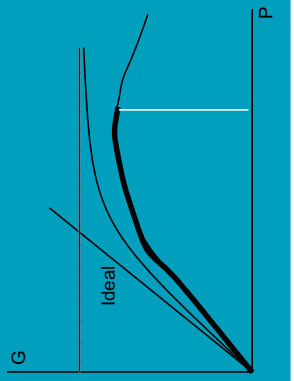
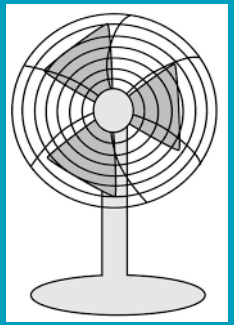






Multiprocesadores

$$P = C_e \cdot v^2 \cdot f + v \cdot I_f$$

J.M. Llabería

Contenido

Capítulo 5
Protocolo de observación en un bus atómico

En el Capítulo 4, cuando un procesador requiere acceder a memoria y no obtiene el acceso al bus suspende la interpretación de instrucciones[1]. En este capítulo, un procesador, que requiere acceder a memoria y no obtiene el bus, sigue interpretando instrucciones mientras: a) no se produzca un riesgo estructural, b) no se produzca un riesgos de datos o c) inicie la interpretación de una instrucción de acceso a memoria[2].

Por otro lado, en el Capítulo 4 se producen riesgos estructurales entre el agente procesador y el agente observador de un CC. Se ha supuesto que los recursos del camino de datos de acceso a la cache sólo permiten un acceso. La decisión que se ha tomado en el Capítulo 4, para gestionar el riesgo, ha sido que el acceso del agente observador es prioritario y se suspende la interpretación de instrucciones. En este capítulo se añaden recursos para reducir los ciclos perdidos por riesgos estructurales. Un CC puede estar sirviendo un acierto en cache, en la mayoría de las situaciones, mientras el agente observador del CC está efectuando una acción de observación.

En el Capítulo 4 no existen peticiones pendiente de obtener acceso al bus, ya que un acceso a memoria es una acción atómica. En este capítulo pueden existir peticiones pendientes de obtener el acceso al bus. Esta característica da lugar a que se cree una ventana temporal entre la petición (necesidad) de acceder al bus y la obtención del acceso al bus (Figura 5.1). Por tanto, a diferencia del Capítulo 4, un acceso a memoria no puede considerarse una petición atómica (acceso a cache y si es necesario acceso al bus y a memoria).

1. En concreto, reinterpreta la instrucción de acceso a memoria e instrucciones más jovenes hasta obtener el acceso al bus. Este mecanismo se utiliza para que todas las acciones en un acceso a memoria se efectúen de forma atómica: a) acceso a la cache y b) utilización del bus y acceso al módulo de memoria.
2. Este funcionamiento se relaja en algunos ejercicios. Por ejemplo, en un procesador se soportan aciertos en cache, cuando se dispone de los derechos de acceso necesarios, existiendo un acceso a memoria pendiente.

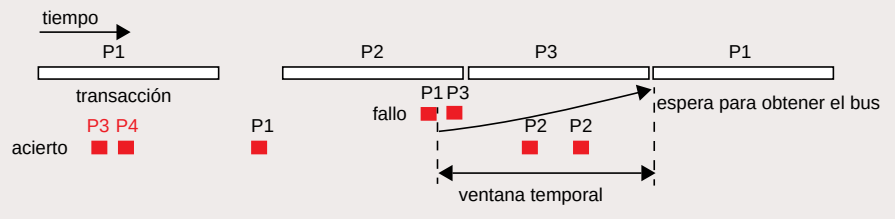


Figura 5.1 Ventana temporal entre petición de acceso al bus y concesión del bus para iniciar la transacción.

Por ejemplo, dos controladores de coherencia (CC1 y CC2), en un protocolo de invalidación con escritura retardada, solicitan el bus en el mismo ciclo. Uno de ellos obtiene el bus (CC1) y el otro debe esperarse (CC2). Mientras el CC2 está pendiente de obtener acceso al bus, debe servir el bloque de datos solicitado en la transacción que ocupa el bus (debida a CC1). Esto es, el agente observador del CC2 debe servir peticiones, mientras el agente procesador del CC2 espera obtener el bus para iniciar una transacción. Además, si las peticiones de los dos controladores de coherencia (CC1 y CC2) acceden al mismo bloque, es posible que las acciones previstas, para la petición pendiente de obtener el bus (CC2), deban modificarse en función de la transacción en curso. Esta última posibilidad da lugar a que, usualmente, la ventana temporal se denomine ventana de vulnerabilidad.

En este Capítulo una transacción de bus es atómica. Esto es, una transacción ocupa el bus desde que se le concede a un CC hasta que finaliza la transacción[3].

Ejercicio

En un protocolo de invalidación con escritura inmediata, describa una situación donde exista una ventana de vulnerabilidad, cuando se considera la posibilidad de peticiones pendientes de obtener el bus.

Respuesta

Un bloque está en estado válido en dos procesadores y los dos procesadores ejecutan concurrentemente una instrucción de escritura a una palabra del mismo bloque. Los dos controladores de coherencia solicitan el bus en el mismo ciclo. El controlador de coherencia del procesador P1 (CC1) obtiene el bus y el controlador de coherencia del procesador P2 (CC2) debe esperarse. La ventana de vulnerabilidad está en CC2. El bus ordena en primer lugar la petición de CC1. Por tanto, la copia del bloque en la cache del procesador P2

3. En los ejercicios se relaja la hipótesis. El bus se utiliza de forma segmentada y existen transacciones concurrentes. Ahora bien, en un primer paso, se garantiza que dos transacciones concurrentes no acceden al mismo bloque. Posteriormente, se analiza el caso de transacciones concurrentes al mismo bloque.

debe invalidarse. En consecuencia, al obtener el CC2 el bus, el estado del bloque es distinto del estado que tenía cuando se efectuó la petición de bus. En ese instante de tiempo se detectó acierto en cache.

En la descripción de las deficiencias en los apartados genéricos no se tiene en cuenta un único protocolo (VI, MLI). Los comentarios y mecanismo descritos pueden ser aplicables a uno de los protocolos descritos en el Capítulo 4 o a ambos. La aplicación concreta a cada protocolo se efectúa en el apartado dedicado al protocolo.

RECURSOS NECESARIOS EN UNA TRANSACCIÓN DE BUS

En un multiprocesador, en una transacción de bus distinguimos varias fases (Figura 5.2): arbitraje (ARB), difusión de la petición (@), observación (Obs), respuesta de observación (ROb) y transmisión de los datos (bloque).

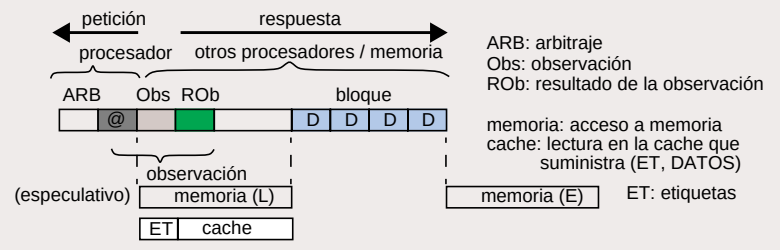


Figura 5.2 Fases en una transacción de bus.

En la Figura 5.3 se muestra un esquema de un multiprocesador donde se identifican las señales en el bus: a) señales de arbitraje, b) dirección (DIR), c) datos (DAT) y c) respuesta de observación (ROb, parada). Notemos que la mayoría de elementos del bus se han dibujado en ambos lados de los nodos, para que la mayoría de las señales fluyan de izquierda a derecha.

Arbitraje (ARB). Los CC solicitan acceso al bus y el árbitro concede el acceso a uno de ellos. Son necesarias señales de petición de bus, el árbitro y señales de concesión de bus.

Difusión de la petición (DIR). Transmisión por el bus de la dirección del bloque accedido y del tipo de acceso además de otras señales de control[4]. Para transmitir por el bus las señales descritas se utilizan conjuntos de cables.

Observación (Obs). Los CC comprueban si el bloque, cuya dirección se transmite por el bus, está almacenado en su cache. Es necesario acceder a los campos etiqueta y estado de la cache. El acceso del agente observador a estos recursos entra en competencia con un posible acceso del agente procesador.

4. No mostradas en la figura.

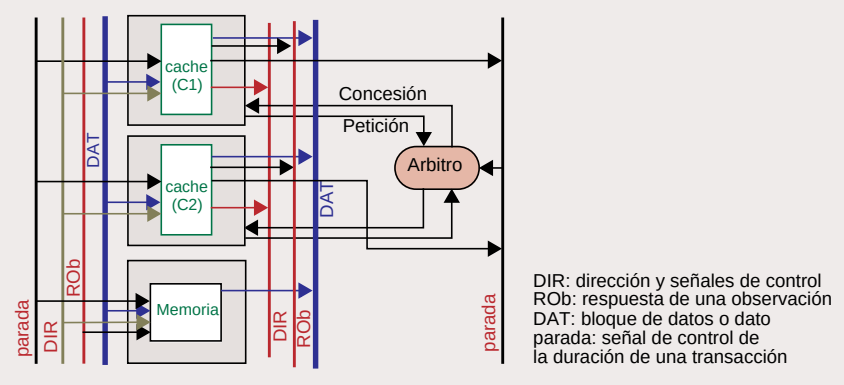


Figura 5.3 Esquema de los elementos utilizados en una transacción de bus.

Respuesta de la observación (ROb, parada). En función de si el bloque está almacenado en cache y de su estado, los controladores de cache emiten una respuesta en las señales del bus dedicadas para este menester (Figura 5.4). En esta fase un CC también puede activar la señal de parada de la transacción cuando debe suministrar el bloque y determina que no puede hacerlo en los ciclos prefijados de la transacción.

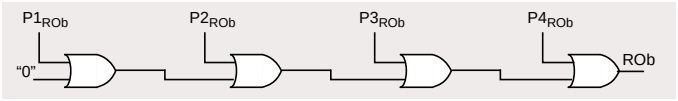


Figura 5.4 Señal de respuesta que es la función OR de las señales activadas por cada procesador.

En la fase de respuesta de cada CC se considera implícita la confirmación de la invalidación al finalizar la transacción, si es el caso. Por otro lado, la memoria u otro CC suministra el bloque solicitado en función del protocolo de coherencia que se utiliza. Cuando el agente observador necesita acceder al campo de datos de la cache, puede entrar en competencia con un posible acceso del agente procesador.

Transmisión de datos (DAT). Por las señales dedicadas en el bus, para este propósito, se transmite: a) el bloque solicitado, b) el dato o c) bloque que se almacenará en memoria. Suponemos que la ocupación del bus es la misma tanto en acciones de lectura de bloque como de actualización de memoria.

Cuando se expulsa de una cache un bloque actualizado se actualiza memoria y la confirmación de memoria se considera implícita al finalizar la transacción de bus.

El bus es síncrono y todas las transacciones tienen la misma duración a menos que se especifique lo contrario.

Riesgos estructurales en el camino de datos de acceso a la cache

El análisis se efectúa para la fase de observación y la fase de respuesta.

Fase de observación

En la fase de observación el agente observador puede tener un riesgo estructural con el agente procesador. El agente observador necesita leer el campo de etiqueta y el campo de estado y efectuar una comparación de direcciones (Figura 5.2)[5]. Para eliminar este riesgo estructural, el cual determina ciclos perdidos, se duplica el campo etiqueta, el campo estado[6] y la lógica de comparación necesaria (Figura 5.5). La copia es accedida para lecturas por el agente observador. Notemos que, mediante el duplicado, hay dos caminos de lectura, uno para cada agente, pero sólo uno de escritura.

En un fallo de cache hay que actualizar las dos copias del campo de etiqueta y el campo de estado. También, en una acción de observación, que determine modificar el estado, hay que modificar las dos copias del campo estado.

El duplicado de los campos etiqueta y estado filtra la mayoría de las observaciones. Esto es, transacciones de otros procesadores que referencian bloques que no están almacenados en la cache o bloques cuyo estado no necesita modificarse.

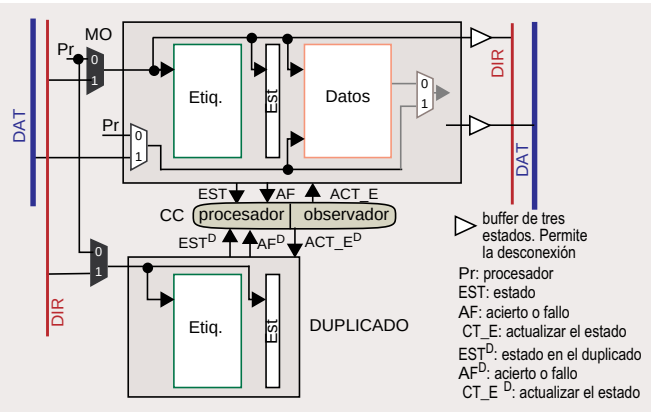


Figura 5.5 Reducción de riesgos estructurales: duplicación del campo etiqueta y estado de la cache.

5. El agente procesador puede estar interpretando una instrucción de acceso a memoria.
6. El campo estado usualmente se implementa utilizando una única estructura con dos puertos de acceso.

Fase de respuesta

En la fase de respuesta a una observación puede ser necesario, dependiendo del protocolo de coherencia, que una cache suministre el bloque. Por tanto, hay que leer en el campo de datos de la cache. Esta circunstancia puede producir un riesgo estructural entre el agente observador y el agente procesador. Cuando no es posible suspender la interpretación de instrucciones se utiliza una señal de "parada", disponible en el bus. Esta señal mantiene la transacción suspendida hasta que el agente observador puede acceder al campo de datos de la cache y suministrar el bloque (Figura 5.2)[7]. En otras palabras, se alarga o extiende la duración de una transacción (Figura 5.6)[8].

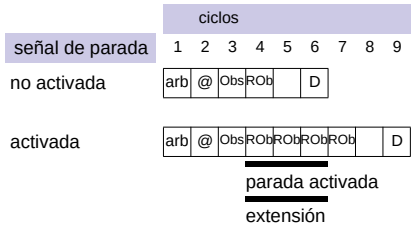


Figura 5.6 Señal de parada para extender la duración de una transacción de bus. Activación y observación en la fase ROb.

VENTANA TEMPORAL ENTRE PETICIÓN E INICIO DE LA TRANSACCIÓN

En este capítulo se permite que existan peticiones pendientes de utilizar el bus mientras el bus está ocupado. Esta característica da lugar a la existencia de una ventana temporal entre la necesidad de acceder al bus (petición) y la obtención del bus. Por tanto, un acceso a memoria, que utiliza el bus, no puede considerarse atómico.

Durante la ventana de tiempo las transacciones de bus que se observan pueden referenciar:

- un bloque distinto que el bloque que referencia la petición que está esperando obtener el bus.
- el mismo bloque que el bloque que referencia la petición que está esperando obtener el bus (ventana de vulnerabilidad).

7. La señal de parada se activa cada ciclo. También puede indicar que se mantiene la transacción suspendida un número determinado de ciclo. Cuando no es suficiente una activación se puede volver a activar al finalizar el periodo de suspensión.
8. Si el bus está segmentado de la forma descrita en el Capítulo 4, todas las transacciones concurrentes se suspenden. Notemos que cualquiera de los CC pueden activar la señal de parada y todos la observan. Memoria también observa la señal parada.

En el primer caso, si la cache debe de responder a la transacción en curso, se produce un riesgo estructural en el autómata del CC. El CC está gestionando un acceso a memoria, que está esperando obtener el bus, y es necesario que atienda la petición de observación. Para eliminar el riesgo estructural el autómata debe ser reentrante. Esto es, debe ser capaz de almacenar el estado de la petición pendiente, gestionar la petición de observación y posteriormente reanudar la gestión de la petición pendiente en el punto donde estaba.

Los agentes en el CC que gestionan la petición pendiente y la observación de la transacción son distintos. Por tanto, es suficiente que el agente procesador no utilice los recursos que necesita el agente observador, durante el lapso de tiempo que dura la transacción. En el peor caso son todos los campos de la cache. Notemos que la respuesta a una observación puede necesitar actualizar el campo estado y leer el campo de datos. En la Figura 5.7 se muestra de forma esquemática el camino de datos de un multiprocesador y las fases: a) arbitraje y b) observación. La transacción de bus corresponde al CC1 y existe una transacción pendiente del CC2 (Figura 5.7 a). Este último CC debe efectuar una observación (Figura 5.7 b).

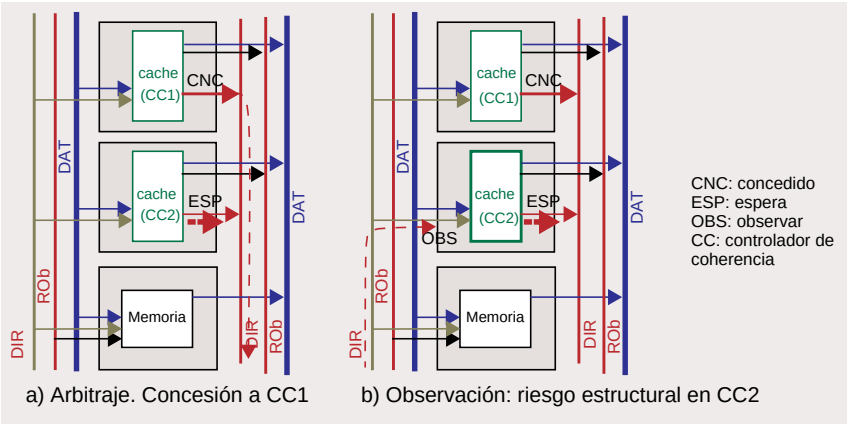


Figura 5.7 Riesgo estructural en el controlador de coherencia: a) arbitraje, b) riesgo estructural.

En el segundo caso, la existencia de la ventana temporal o de vulnerabilidad permite que, mientras se está esperando la concesión de acceso al bus, se observe una transacción al mismo bloque. Esta transacción, en el orden de bus, es previa a la que está esperando. Por tanto, hay que tenerla en cuenta, ya que puede determinar modificaciones del estado del bloque. Por otro lado, al efectuar la petición de bus se han previsto unas acciones cuando se conceda el bus, sin tener en cuenta la transacción que hay en el bus. En consecuencia, en la petición pendiente hay que tener en cuenta la transacción

en curso y es posible que haya que modificar las acciones previstas. Si este es el caso diremos que se produce un riesgo de datos.

En estas condiciones, la detección de la necesidad de acceder al bus (petición de bus) y la realización de la transacción (concesión de bus) no puede considerarse una acción atómica (Figura 5.1). Entonces, para gestionar esta característica se introducen los denominados estados transitorios. Estos estados se utilizan para separar la acción de solicitar acceso al bus de la acción de ocupar el bus, para iniciar y finalizar una transacción, la cual es atómica. Cada una de estas dos acciones es individualmente atómica, pero no lo son conjuntamente como en el Capítulo 4. El estado transitorio se utilizará para gestionar una espera entre acciones atómicas.

En la Figura 5.8 se muestra un diagrama de transiciones entre estados con dos estados estables y dos estados transitorios. La forma de etiquetar un estado transitorio es utilizar como prefijo el estado fuente y como sufijo el estado destino. Las acciones de petición de bus (PtBus) y concesión de bus (CnBus) se utilizan para identificar: a) el inicio de la ventana de vulnerabilidad y b) el final de la ventana de vulnerabilidad. Cuando se efectúa la petición de bus se realiza una transición desde el estado A al estado AB. En el estado AB puede que se observen transacciones que no requieran modificar las acciones previstas. Entonces, cuando se recibe la señal de concesión de bus se inicia la transacción prevista y se efectúa la transición al estado estable B. Si en el estado AB se observa una transacción, que requiere modificar las acciones previstas, se efectúa una transición a un estado transitorio que la identifique (ABB). En el estado transitorio ABB, cuando se recibe la señal de concesión de bus, se actúa en consecuencia.

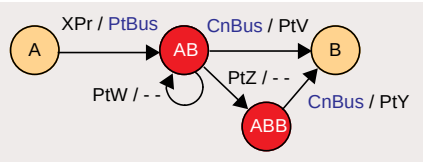


Figura 5.8 Estados transitorios para distinguir entre petición e inicio de la transacción de bus.

PROTOCOLO DE INVALIDACION CON ESCRITURA INMEDIATA

La descripción funcional del protocolo se ha efectuado en el Capítulo 4. En cuanto al camino de datos es el descrito en el mismo capítulo, teniendo en cuenta la replicación de los campos etiqueta y estado de la cache (Figura 5.5).

Eventos y transacciones

Además de los eventos descritos en el Capítulo 4 hay que tener en cuenta los eventos relacionados con el árbitro del bus: a) petición de bus y b) concesión de bus. En la Tabla 5.1 se describen estos eventos.

Bus		Comentario
Peticiones del CC	**Respuestas al CC**	
PtBus: petición de acceso al bus	CnBus: concesión de acceso al bus	Cada CC tiene este par de señales. Cuando un CC determina que necesita utilizar el bus (escritura o fallo de lectura) efectúa una petición al árbitro. Cuando el árbitro concede el acceso a un CC este utiliza el bus para iniciar la transacción.

Tabla 5.1 Peticiones al árbitro y respuestas.

Cada CC está conectado al árbitro mediante señales dedicadas de petición y concesión. El árbitro concede una de las peticiones activadas en cada fase de arbitraje. Una fase de arbitraje se realiza después de que finalice la transacción en curso[9] o cuando no hay transacción en curso.

Por razones de completitud de este capítulo, en la Tabla 5.2 y en la Tabla 5.3 se vuelven a describir los eventos descritos en el Capítulo 4. En las figuras incluidas en las tablas se muestra de forma explícita el árbitro de bus (Arb) y las señales de petición (PtBus) y concesión (CnBus) de bus.

En la Tabla 5.2 se describen las peticiones del procesador al CC. El procesador efectúa dos operaciones: a) lectura (load) y b) escritura (store). Cuando es acierto de lectura sólo se accede a la cache. En un fallo de lectura y en una escritura se accede a memoria.

Procesador	Controlador de coherencia (CC)	Comentario
Peticiones al CC	**Respuestas del CC**	
LPr (load): lectura de un dato	dato	Si es un acierto en cache se lee el dato de cache. En caso contrario, antes de suministrar el dato, se inicia una transacción de lectura del bloque (Pt) y se asigna un contenedor para almacenar el bloque.
EPr (store): escritura de un dato	confirmación de la escritura (implícita al finalizar la transacción)	En cualquier caso se actualiza la memoria (PtE). Si es un acierto y el estado del bloque es válido se escribe el dato en cache al finalizar la transacción. No se asigna un contenedor en caso de fallo.

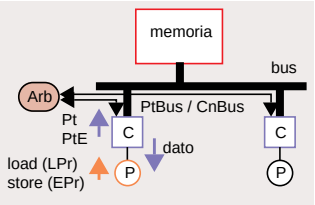


Tabla 5.2 Protocolo VI. Peticiones del procesador y respuestas.

9. Cuando hay una transacción en curso, la fase de arbitraje se suele solapar con la finalización de la transacción.

En la Tabla 5.3 se describen las peticiones del CC a memoria (Pt y PtE) y acciones del CC para gestionar la ubicación de bloques en la cache (conflictos, CcRe). Se distinguen las acciones de lectura de bloque (Pt) y escritura de dato (PtE). También se describe la acción de expulsión de un bloque debido a un conflicto.

Controlador de coherencia (CC)	Memoria / otros CC	Comentario
Peticiones a memoria y acciones	**Respuestas**	
Pt: petición de lectura de un bloque	bloque de datos	Se lee el bloque de datos de memoria y se suministra.
PtE: petición de escritura de un dato	confirmación de la consolidación de la escritura de forma implícita (un CC invalida el bloque si tiene copia)	Se actualiza la memoria con el dato
CcRe: expulsión de un bloque.		Se invalida la información del contenedor. No se notifica a memoria, ya que está actualizada

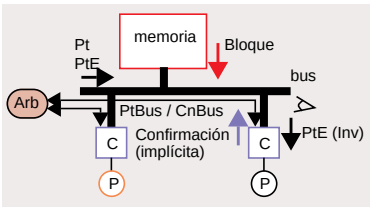


Tabla 5.3 Protocolo VI. Peticiones del CC a memoria, respuestas de memoria y de los otros CC y acciones del CC.

Estados y transiciones

En este protocolo hay dos estados estables: inválido (I) y válido (V). El número de estados transitorios es cuatro, que denominamos: IV, VV, II y VVI.

La descripción de las transiciones entre estados la efectuaremos en tres partes. En primer lugar tendremos en cuenta las transiciones debidas a las peticiones del procesador entre estados estables. En segundo lugar las transiciones inducidas por el observador entre estados estables[10]. Seguidamente analizaremos las transiciones inducidas por el observador en estados transitorios. Finalmente, como existe relación entre estas últimas y las primeras transiciones descritas se presentan el diagrama completo de estados y transiciones.

Eventos del procesador y reemplazo

El procesador se bloquea cuando hay que acceder a memoria y hay un acceso a memoria pendiente. Por otro lado, no hay buffer de escrituras[11]. Por tanto, un CC no gestiona peticiones del procesador cuando un bloque está en un estado transitorio.

10. Estos dos conjuntos de transiciones son los descritos en el Capítulo 4 utilizando los estados transitorios para identificar las acciones de petición y concesión de bus.
11. Posteriormente se relaja esta hipótesis.

En un estado estable (I, V), después de determinar que hay que acceder a memoria y a la vez que efectúa la petición de bus, el agente procesador cambia el estado estable por un estado transitorio. Al obtener la concesión de bus, el agente procesador emite la petición por el bus. En la fase de datos de la transacción, correspondiente a la respuesta, se efectúa la transición del estado transitorio al estado estable final (Figura 5.9). En una acción de reemplazo se invalida el bloque.

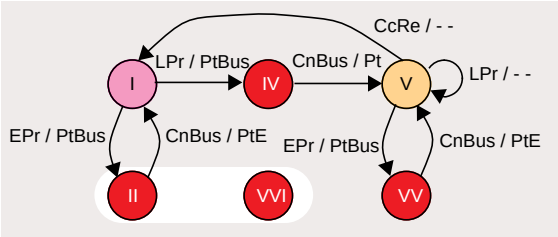


Figura 5.9 Protocolo VI. Transiciones desde estados estables iniciadas por el agente procesador y acción de reemplazo.

Eventos externos: acciones inducidas por observaciones

En la Figura 5.10 se muestra el diagrama de transiciones entre estados. Sólo hay una transición entre dos estados estables distintos. El agente observador al observar una transacción de escritura invalida el bloque, si está almacenado en cache.

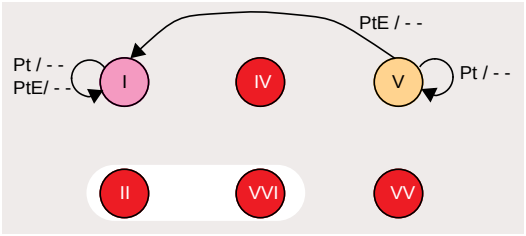


Figura 5.10 Protocolo VI. Transiciones desde estados estables iniciadas por el agente observador.

Eventos externos en estados transitorios

Mientras se está esperando obtener el bus (ventana de vulnerabilidad) se pueden observar transacciones al mismo bloque. En estos casos usualmente se dice que se produce una condición de carrera (“race condition”) que hay que resolver para que el estado final sea consistente[12].

En los estados transitorios IV e II estas observaciones no determinan ningún cambio de estado. Se llega a ellos después de detectar un fallo de cache. Por tanto, no se tiene copia del bloque en cache.

12. Dos transacciones referencian el mismo bloque y hay que serializarlas.

Sin embargo, en el estado VV se tiene copia del bloque en cache. Notemos que si no hubiera la petición pendiente, el bloque estaría en estado V y se invalidaría cuando el agente observador observara una transacción PtE en el bus (Figura 5.10). En otras palabras, la petición de escritura que ocupa el bus ha sido ordenada por el árbitro, en el orden global, antes que la petición pendiente.

En el estado VV, las acciones pendientes, después de obtener el bus son: a) actualizar memoria y b) actualizar la cache (Figura 5.11). Ahora bien, si se observa una escritura en la ventana de vulnerabilidad el bloque no contiene información válida. Por tanto, cuando se obtenga la concesión del bus sólo hay que actualizar memoria[13]. En consecuencia, al observar una transacción PtE en el estado transitorio VV, se cambia a otro estado transitorio denominado VVI, el cual se utiliza para recordar que previamente el árbitro ha ordenado una escritura emitida desde otro CC. Desde este estado transitorio, cuando se obtenga la concesión de bus, se actualizará memoria y se pasará al estado estable I.

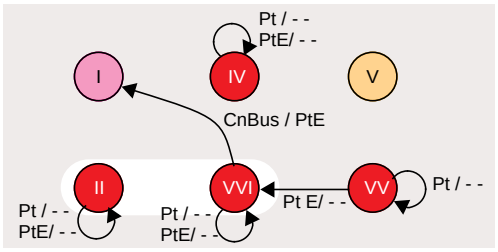


Figura 5.11 Protocolo VI. Transiciones desde estados transitorios iniciadas por el agente observador y la concesión del bus.

Diagrama completo de estados y transiciones

En la Figura 5.12 se muestran los dos estados estables y los cuatro estados transitorios con todas las posibles transiciones debidas a eventos del agente procesador, eventos del agente observador y a acciones de reemplazo.

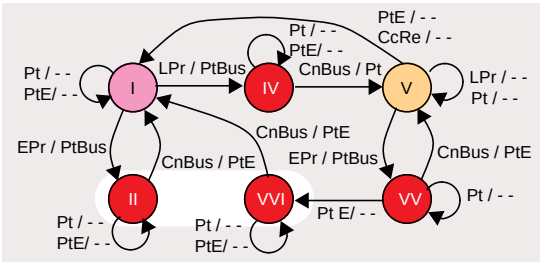


Figura 5.12 Bus atómico. Protocolo VI. Diagrama de estado y transiciones.

13. Recordemos que en una escritura, si no se tiene copia del bloque no se asigna contenedor de cache.

Notemos que todas las acciones y respuestas de salida en los estados II y VVI son las mismas (Pt / - -, PtE / - -, CnBus / PtE, Figura 5.12). Por tanto, los dos estados transitorios pueden agruparse en un único estado transitorio, lo cual no realizaremos por razones de claridad en la descripción.

Tabla de estados y transiciones

En la Tabla 5.4 se muestra la descripción de los estados, estables y transitorios, eventos y transiciones entre estados en formato tabla. Las casillas que no contienen información indican un error: en un estado determinado no puede llegar el evento que determina la casilla correspondiente en el cruce.

			Eventos del procesador y reemplazo			Bus	Eventos externos (transacciones de bus)	
			LPr	EPr	CcRe	CnBus	Pt	PtE
Estados	Estables	I	PtBus; IV	PtBus; II			- - ; I	- - ; I
		V	- - ; V	PtBus; VV	- - ; I		- - ; V	- - ; I
	transitorios	IV				Pt; V	- - ; IV	- - ; IV
		VV				PtE; V	- - ; VV	- - ; VVI
		VVI				PtE; I	- - ; VVI	- - ; VVI
		II				PtE; I	- - ; II	- - ; II

Tabla 5.4 Bus atómico. Protocolo VI. Tabla de estados y transiciones.

Representación de transacciones y transiciones entre estados

En este apartado se muestran dos formas de representar las transacciones de bus y transiciones entre estados al ejecutar una secuencia de accesos a memoria: a) en formato tabla y b) mediante un diagrama temporal. Finalmente se muestra, mediante varios gráficos, una animación.

Representación en formato tabla

Utilizaremos una tabla, donde cada fila representa un acceso a memoria y no se gestiona el siguiente acceso hasta que ha finalizado el anterior. En una fila, de izquierda a derecha, después de la instrucción de acceso a memoria, se especifica:

1. El estado transitorio, si es el caso.

2. La transacción de bus, si es el caso.
3. El nombre de la variable y el valor en memoria.
4. Quién suministra el dato o bloque (cache o memoria).
5. Para las cache donde se modifica la información almacenada, el nombre de la variable, el valor y el estado estable del bloque.

Ejemplo. En la Tabla 5.5 se muestra una secuencia de accesos a memoria realizada por tres procesadores. La variable t no está almacenada en ninguna cache y el valor en memoria es dos.

Los dos primeros accesos son instrucciones load y producen fallos en cache. El estado transitorio del bloque en las caches, que gestionan los controladores de coherencia CC1 y CC3, es IV y el estado final es V. El siguiente acceso, efectuado por el procesador P3, es un acierto de escritura, se invalida la copia en la cache del procesador P2 y se actualiza la copia de cache (C3) y memoria. Notemos que la actualización de memoria queda reflejada en la fila. El estado transitorio del bloque en la cache C3 es VV y el estado final es V.

	C 1	C 2	C 3	Bus	mem.			C 1			C 2			C 3		
acceso	est.	est.	est.	trans.	var.	val.	sum.	var.	val.	est.	var.	val.	est.	var.	val.	est.
1. P1 load t	IV			Pt	t	2	mem.	t	2	V						
2. P3 load t			IV	Pt	t	2	mem							t	2	V
3. P3 store t (21)			VV	PtE	t	21	C3	t	2	I				t	21	V
4. P1 load t	IV			Pt	t	21	mem.	t	21	V						
5. P2 store t (8)		II		PtE	t	8	C2	t	21	I				t	21	I

Tabla 5.5 Bus atómico. Protocolo VI. Formato tabla: transacciones y cambios de estado del bloque en las caches en una secuencia de accesos a memoria.

Seguidamente en el procesador P1 se produce un fallo de cache debido a una instrucción load. Finalmente el procesador P2, que no tiene almacenado el bloque en cache, ejecuta una instrucción store. Ello da lugar a que se invaliden las copias en las caches de los otros procesadores y se actualice memoria. El estado transitorio del bloque en la cache C2 es II y el estado final es I. Recordemos que no se asigna contenedor.

Diagrama temporal

EL diagrama temporal y la representación de una transacción son como en el Capítulo 4.

En el segundo grupo de filas del final de la tabla también se representa el estado transitorio al solicitar acceso al bus. En la fase arb de una transacción se especifica el estado transitorio del bloque. Al finalizar la transacción se especifica el estado estable del bloque.

Ejemplo. Para la secuencia de accesos a memoria de la Tabla 5.5, en la Figura 5.13 se muestra un diagrama donde se observa la secuencia de transacciones de bus con las fases.

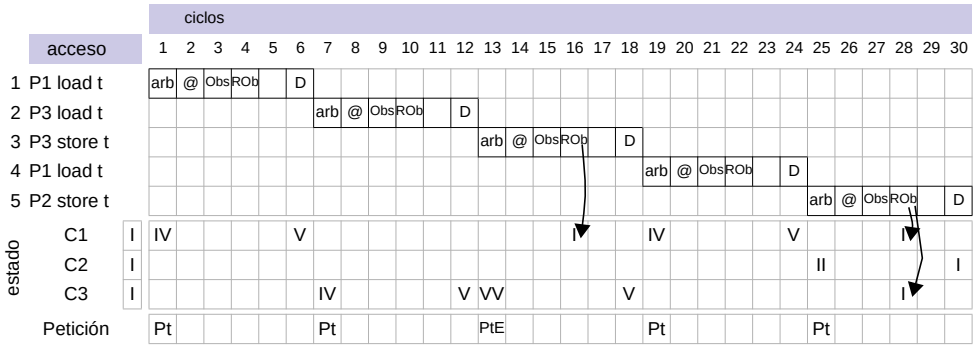


Figura 5.13 Bus atómico. Protocolo VI. Diagrama temporal y cambios de estado de una secuencia de accesos.

Animación

En la Figura 5.14 se muestra, mediante fotogramas, una animación de la secuencia de accesos a memoria de la Tabla 5.5. Además de las peticiones y respuestas, se muestran los cambios de estado utilizando diagramas de transición entre estados. Cuando un bloque no está almacenado en un contenedor de cache se supone que está en estado inválido. Para reducir la información representada en la figura, no se muestran los casos en que un agente observador no modifica el estado de un bloque al observar una transacción.

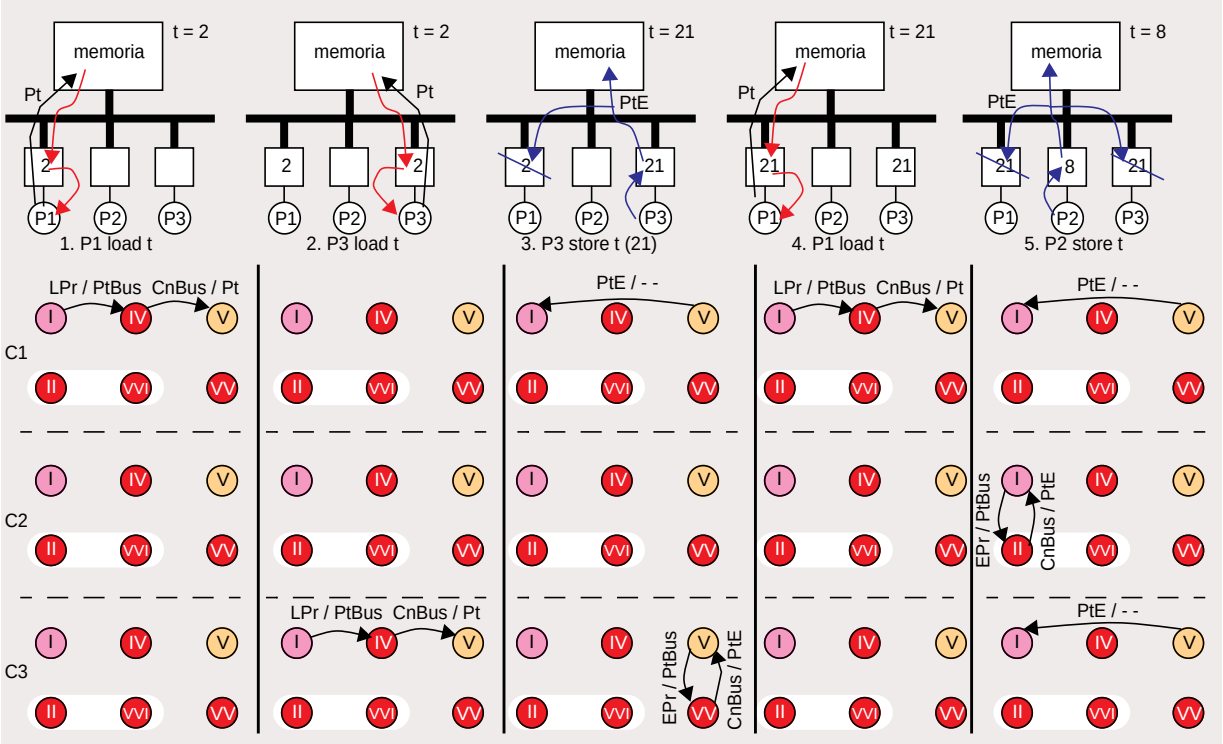


Figura 5.14 Bus atómico. Protocolo VI. Secuencia de accesos a memoria y fotogramas de las peticiones y respuestas y cambios de estado. En una observación, cuando el estado es inválido no se muestra la transacción.

Representación en formato tabla de peticiones concurrentes

En una secuencia se agrupan los accesos a memoria y la agrupación se muestra mediante líneas horizontales continuas. Usualmente se agrupan de dos en dos.

Cada grupo de peticiones se gestiona independientemente. No se empieza a gestionar el siguiente grupo de peticiones hasta que ha finalizado el anterior. Cuando los accesos a memoria de un grupo necesitan acceder al bus se supone que las acciones de petición se realizan concurrentemente en el mismo ciclo.

En una fila de la tabla, de izquierda a derecha, después de la instrucción de acceso a memoria se especifica:

1. El orden de concesión del bus a la petición (1ª, 2ª, . . .). Supondremos usualmente que en un grupo de peticiones, el bus se concede en el orden en el cual se especifican los accesos a memoria en la secuencia de accesos.

2. El estado transitorio al efectuar la petición de bus.

3. La petición que ocupa el bus en la columna correspondiente al orden de concesión.

4. El nombre de la variable y el valor en memoria.

5. Quién suministra el dato o bloque (cache o memoria).

6. Para las caches donde se modifica la información almacenada, el nombre de la variable, el valor y el estado estable o transitorio del bloque.

Ejemplo. Una secuencia de accesos a memoria referencia las variables u y t que están contenidas en bloques distintos. Estos bloques al almacenarse en cache se ubican en contenedores distintos. Los bloques no están inicialmente almacenadas en cache y los valores de las variables en memoria son u = 7 y t = 2.

Los dos primeros accesos concurrentes son fallos de lectura. El CC1 obtiene la concesión del bus en primer lugar. El estado transitorio de los bloques en los dos casos es IV y el estado estable final es V.

	arb.	C 1	C 2	C 3	Bus (trans.)		mem.			C 1			C 2		
acceso	ord.	est.	est.	est.	1º	2º	var.	val.	sum.	var.	val.	est.	var.	val.	est.
1. P1 load t	1º	IV			Pt		t	2	mem	t	2	V			
1. P2 load u	2º		IV			Pt	u	7	mem				u	7	V
2. P1 store u (9)	1º	II			PtE		u	9	C1				u	7	I
2. P2 store t (1)	2º		II			PtE	t	1	C2	t	2	I			
3. P2 load u	1º	IV			Pt		u	9	mem				u	9	V
3. P1 load u	2º		IV			Pt	u	9	mem	u	9	V			
4. P1 store u (12)	1º	VV			Pt		u	12	C1	u	12	V	u	9	VVI
4. P2 store u (3)	2º		VV			Pt	u	3	C2	u	12	I	u	9	I

Tabla 5.6 Bus atómico. Protocolo VI. Formato tabla. transacciones y cambios de estado en una secuencia de accesos a memoria concurrentes.

El segundo grupo de accesos concurrentes son fallos de escritura. El CC1 obtiene la concesión del bus en primer lugar. El estado transitorio de los bloques en los dos casos es II. Al ser fallo en escritura no se asigna contenedor en cache. Cada acceso actualiza una variable distinta. El acceso del procesador P1 invalida la copia del bloque en la cache del procesador P2 y el

acceso del procesador P2 invalida la copia del bloque en la cache del procesador P1. El tercer grupo de accesos concurrentes son fallos de lectura. Este caso es semejante al primer grupo de accesos concurrentes.

El cuarto grupo de accesos concurrentes son aciertos de escritura cuando se efectúa la petición de bus. Por tanto, el estado transitorio en los dos casos es VV. Notemos que los dos accesos son a la misma variable. La concesión de bus la obtiene en primer lugar el CC1. El estado estable del bloque en la cache C1 al finalizar la transacción es V y el estado transitorio del bloque en la cache C2 es VVI. Notemos que la transacción en curso y la petición pendiente de CC2 referencian la misma variable. Por tanto, el CC2 debe tener en cuenta la ordenación determinada por el bus. El estado transitorio en la cache C2 se muestra en la columna correspondiente a la cache C2.

Al finalizar la transacción iniciada por el CC2 el estado estable del bloque en las caches C1 y C2 es I. La transacción del CC2 invalida la copia en la cache de C1 y previamente la transacción del CC1 había indicado que debía invalidarse la copia en la cache C2 (estado transitorio VVI).

Diagrama temporal de peticiones concurrentes

En un grupo de peticiones suponemos que se solapa la fase de arbitraje de la próxima transacción con la finalización de la transacción previa.

En las Figura 5.15 se muestra un diagrama temporal de la ocupación del bus y de los cambios de estado en una transacción.

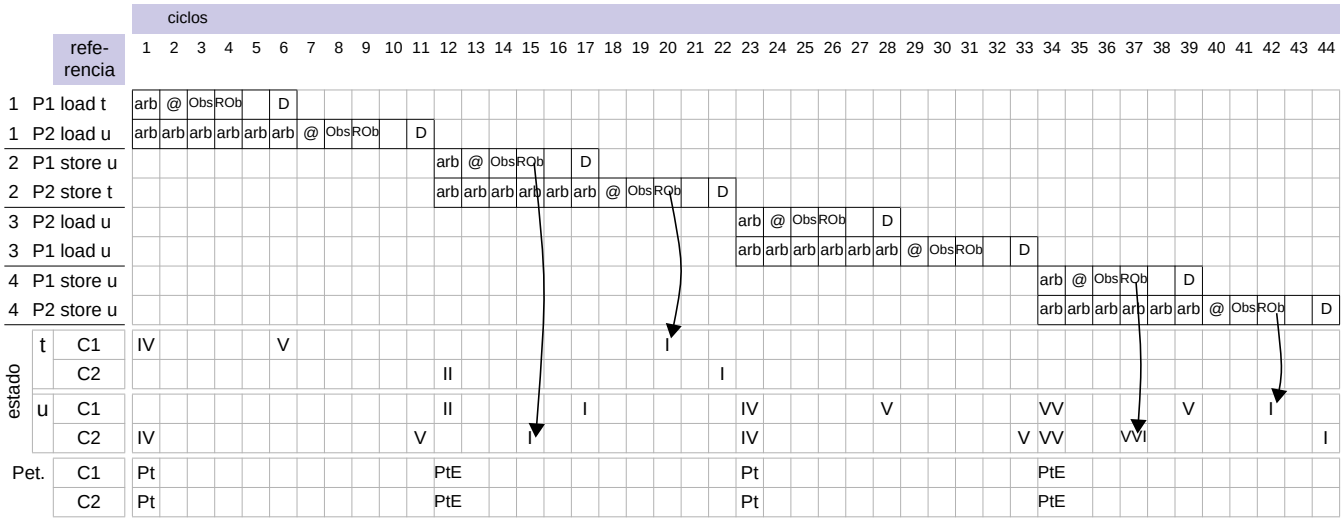


Figura 5.15 Bus atómico. Protocolo VI. Diagrama temporal de una secuencia de accesos concurrentes.

Verificación no formal de coherencia y consistencia

La base es la descripción efectuada en el Capítulo 4. La diferencia se centra en los estados transitorios de un bloque en cache. Durante la ventana temporal se memoriza localmente y por bloque en el CC, la observación de una escritura que haya ordenado previamente el árbitro del bus.

Coherencia de cache

Orden de programa en accesos a la misma posición de memoria. El procesador efectúa los accesos en el orden determinado por el L.M. El procesador se bloquea en un acceso a memoria, hasta que finaliza la transacción pendiente[14].

Propagación de escrituras. Al utilizar escritura inmediata todas las escrituras se transmiten por el bus. Una escritura se propaga mediante una transacción y es observada por todos los CC. Las invalidaciones se efectúan durante la transacción y la respuesta de cada invalidación está implícita en la transacción de bus.

Serialización de escrituras (atomicidad de una escritura).

- Punto de serialización: El bus es el punto de serialización de las escrituras a la misma posición de memoria[15]. En el bus sólo hay una transacción en curso y es observada por todos los CC. El arbitraje del bus determina el orden de serialización de las transacciones de escritura a una posición de memoria. Si una cache tiene el bloque en un estado estable la acción de invalidación es inmediata. En caso contrario se memoriza mediante un estado transitorio del bloque. Este estado transitorio indica que previamente ha sido serializada por el árbitro una escritura iniciada por otro CC. Entonces, las escrituras a una posición de memoria son serializadas por el árbitro en coordinación, o con ayuda, de los CC de forma local durante la ventana de vulnerabilidad de una petición de acceso al bus.
- Consolidación de una escritura: Una escritura está consolidada al finalizar la transacción[16]. Las acciones que se toman localmente en un CC al obtener el bus tienen en cuenta si previamente ha sido serializada la escritura de otro CC al mismo bloque. Por otro lado, una escritura es atómica, ya que, una vez finalizada la transacción correspondiente, una instrucción load posterior lee el valor establecido por la escritura previa en el orden de bus. Las copias del bloque son invalidadas en cada

14. Además, hay que garantizar las dependencias de datos en el hilo que se ejecuta.
15. Aún más, es el punto de serialización de las escrituras y los fallos de lectura a cualquier posición de memoria.
16. Esta decisión es conservadora. La escritura puede considerarse consolidada cuando ocupa el bus.

transacción de escritura. Un fallo de lectura lee el valor establecido por la escritura previa en el orden de bus. Un acierto de lectura lee el valor obtenido en el fallo de lectura previo por la escritura previa, que acierta en cache, del mismo procesador.

Consistencia secuencial de memoria

Se utilizan los mismo razonamientos que en el Capítulo 4, ya que una escritura puede considerarse consolidada al finalizar la transacción.

PROTOCOLO DE INVALIDACION CON ESCRITURA RETARDADA

La descripción funcional del protocolo se ha efectuado en el Capítulo 4. El camino de datos es el descrito en el mismo capítulo teniendo en cuenta la replicación de los campos etiqueta y estado de la cache (Figura 5.5).

Eventos y transacciones

Además de los eventos descritos en el Capítulo 4 hay que tener en cuenta los eventos relacionados con el árbitro del bus: a) petición de bus y b) concesión de bus. En la Tabla 5.1 se describen estos eventos.

Por razones de completitud de este capítulo se vuelven a describir los eventos descritos en el Capítulo 4 en la Tabla 5.7 y en la Tabla 5.8. En las figuras incluidas en las tablas se muestra de forma explícita el árbitro de bus (Arb) y las señales de petición (PtBus) y concesión (CnBus) de bus.

En la Tabla 5.7 se describen las peticiones del procesador al CC. El procesador efectúa dos operaciones: a) lectura (load) y b) escritura (store). En una lectura, si es acierto en cache sólo se accede a cache. En una escritura si es acierto y el estado es M se actualiza el bloque almacenado en cache. En los otros casos se solicita el bloque a memoria y si se pretende actualizar el bloque, la solicitud es con intención de modificación.

Procesador	Controlador de cache (CC)	Comentario
Peticiones al CC	**Respuestas del CC**	
LPr (load): lectura de un dato	dato.	Si es un acierto en cache se lee el dato de cache. En caso contrario, antes de suministrar el dato, se inicia una transacción de lectura de bloque (Pt) y se asigna un contenedor para almacenar el bloque.
EPr (store): escritura de un dato	confirmación de la escritura.	Si es un acierto y el estado del bloque es M se escribe el dato en cache. En caso contrario, en primer lugar, se solicita el bloque con intención de modificación (PtIm).

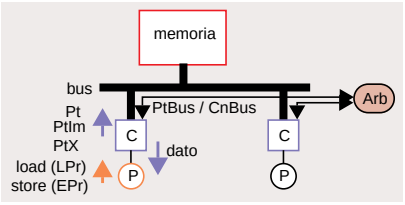


Tabla 5.7 Protocolo MLI. Peticiones del procesador y respuestas.

En la Tabla 5.8 se describen las peticiones y respuestas del CC y acciones del mismo para gestionar la ubicación de bloques en cache.

Controlador de coherencia	Memoria / otros CC	Comentario
Peticiones a memoria y acciones del CC y memoria	**Respuestas**	
Pt: petición de lectura de un bloque	bloque de datos.	Se obtiene el bloque de datos.
PtIm: petición de lectura de un bloque con intención de modificación	bloque de datos y confirmación de las invalidaciones (implícito al finalizar la transacción).	Se obtiene el bloque de datos en exclusividad.
	CaC: suministro del bloque.	Bloque en estado M. La cache suministra el bloque solicitado en una transacción.
	MOD: bloque en estado M.	Bloque en estado M. Inhibe el suministro desde memoria.
CcRe: expulsión de un bloque		Se invalida la información del contenedor.
PtX: petición de actualización de memoria		Actualización de memoria con un bloque en estado M.
Dev: actualización de memoria		Actualización de memoria en CaC, si es el caso.

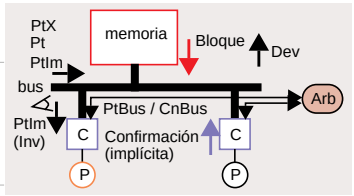


Tabla 5.8 Protocolo MLI. Peticiones del CC a memoria, respuestas de memoria y de los otros CC y acciones del CC.

En la Tabla 5.8 se distinguen las peticiones: a) lectura de bloque y b) lectura de bloque con intención de modificación. Las respuestas son: a) suministro del bloque y b) señal para inhibir el suministro de memoria. Finalmente se describe la petición asociada con la expulsión de un bloque modificado y la acción inducida de actualización de memoria, si es el caso.

Estados y transiciones

La memoria puede no estar actualizada y los posibles estados estables de un bloque en un contenedor de cache son: a) Inválido (I), b) Lectura (L) y c) Modificado (M). El número de estados transitorios es cuatro, que denominamos: IL, LM, IM y MI.

La descripción de las transiciones entre estados la efectuaremos en tres partes. En primer lugar tendremos en cuenta las transiciones entre estados estables debidas a las peticiones del procesador. En segundo lugar las transiciones entre estados estables inducidas por el observador[17]. Seguidamente analizaremos las transiciones inducidas por el observador en estados transitorios. Finalmente, como existe relación entre estas últimas y las primeras transiciones descritas se presenta el diagrama completo de estados y transiciones.

Eventos del procesador y reemplazo

El procesador se bloquea cuando hay que acceder a memoria y hay un acceso a memoria pendiente. Por otro lado, no hay buffer de escrituras. Por tanto, un CC no gestiona peticiones del procesador cuando un bloque está en un estado transitorio.

En un estado estable, después de determinar que hay que acceder a memoria (I, L) y a la vez que efectúa la petición de bus (PtBus), el agente procesador cambia el estado estable por un estado transitorio (IM, IL, LM, Figura 5.16).

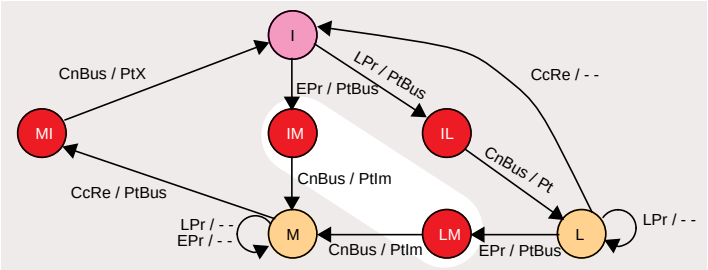


Figura 5.16 Protocolo MLI. Transiciones desde estados estables iniciadas por el agente procesador y acción de reemplazo.

17. Estos dos conjuntos de transiciones son los descritos en el Capítulo 4 utilizando los estados transitorios para identificar las acciones de petición y concesión de bus.

Al obtener la concesión de bus (CnBus), y por tanto emitir la petición por el bus, se efectúa la transición del estado transitorio al estado estable final. En una acción de reemplazo se invalida el bloque. Si el bloque está en estado M previamente hay que actualizar memoria. Para ello, se solicita acceso al bus (estado MI) y una vez concedido, se solicita a memoria que proceda a la actualización (PtX).

Eventos externos: acciones inducidas por observaciones

En la Figura 5.17 se muestra el diagrama de transiciones entre estados. Hay tres transiciones entre estados estables distintos.

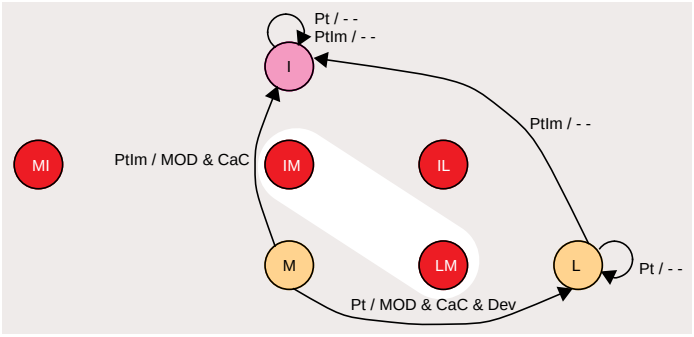


Figura 5.17 Protocolo MLI. Transiciones desde estados estables iniciadas por el agente observador.

Las transiciones desde el estado M requieren suministrar el bloque (CaC), además de indicar que la memoria debe inhibirse del suministro (señal MOD activada). En la transición al estado L se realizan las mismas acciones y además, se actualiza memoria (Dev)[18].

Eventos externos en estados transitorios

Mientras se está esperando obtener el bus (ventana de vulnerabilidad) se pueden observar transacciones al mismo bloque (Figura 5.18).

En los estados transitorios IL, IM y LM estas observaciones no determinan ningún cambio de estado. A los dos primeros estados se llega después de detectar un fallo de cache. Por tanto, no se tiene copia del bloque en cache. Al estado transitorio LM se llega debido a que se necesita la exclusividad para actualizar una palabra del bloque. Aunque el estado inicial era L y se tiene copia del bloque, se solicita el bloque con intención de modificarlo. Por tanto, la petición de exclusividad se gestiona igual que un fallo de cache.

18. En una transición al estado I también se puede actualizar memoria, aunque la actualización es superfula. La transición ente estados está inducida por una petición de lectura con intención de modificación.

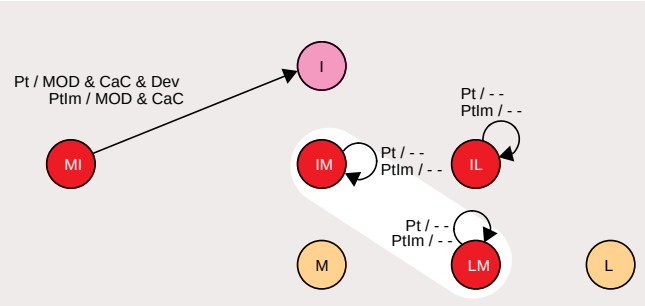


Figura 5.18 Protocolo MLI. Transiciones desde estados transitorios iniciadas por el agente observador.

En el estado MI se tiene copia del bloque en cache y la memoria no está actualizada. De hecho, se está esperando la concesión del bus para actualizar memoria, ya que el CC expulsa el bloque. La cache que tiene el bloque en el estado MI es la encargada de suministrar el bloque, cuando observa una petición que accede al bloque. Entonces, al observar una transacción, donde se referencia el bloque que se quiere expulsar, se suministra (CaC) y se indica a memoria que inhiba el suministro (MOD). Cuando la petición es Pt adicionalmente se actualiza memoria, aprovechando el suministro de la cache (operación Dev).

Conceptualmente la acción es la invalidación del bloque en cache. Entonces, el CC invalida el bloque y desactiva la solicitud de petición de acceso al bus[19].

Diagrama completo de estados y transiciones

En la Figura 5.19 se muestran los tres estados estables y los cuatro estados transitorios con todas las transiciones debidas a eventos del agente procesador, eventos del agente observador y a acciones de reemplazo.

Notemos que todas las acciones y respuestas de salida en los estados IM y LM son las mismas (Pt / - -, PtIm / - -, CnBus / PtIm, Figura 5.19). Por tanto, los dos estados transitorios pueden agruparse en un único estado transitorio, lo cual no realizaremos por razones de claridad en la descripción.

19. Esta alternativa no es válida en algunas redes ordenadas donde, por ejemplo, se encolan las peticiones de arbitraje.

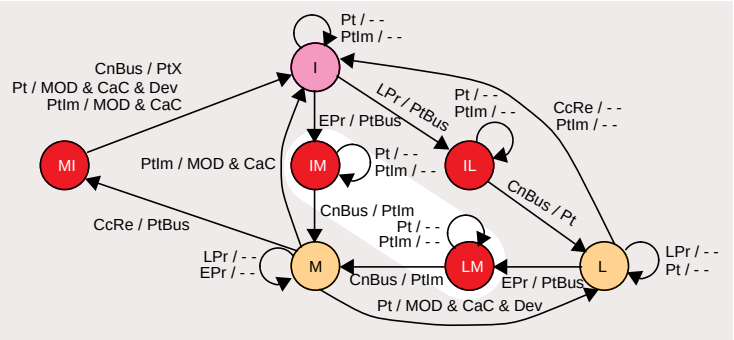


Figura 5.19 Bus atómico. Protocolo MLI. Diagrama de estado y transiciones.

Tabla de estados y transiciones

En la Tabla 5.9 se muestra la descripción de los estados estables y transitorios, eventos y transiciones entre estados en formato tabla. Las casillas que no contienen información indican un error: en un estado determinado no puede llegar el evento que determina la casilla correspondiente en el cruce.

			Eventos del procesador y reemplazo			Arbitro	Eventos externos (transacciones de bus)	
			LPr	EPr	CcRe	CnBus	Pt	PtIm
Estado	Estables	I	PtBus ; IL	PtBus; IM			- - ; I	- - ; I
		L	- - ; L	- - , PtBus; LM	- -; I		- - ; L	- - ; I
		M	- - ; M	- - ; M	PtBus; MI		MOD&CaC&Dev; L	MOD&CaC; I
	Ttransitorios	IL				Pt; L	- - ; IL	- -; IL
		LM				PtIm; M	- - ; LM	- -; LM
		IM				PtIm; M	- - ; IM	- - ; IM
		MI				PtX; I	MOD&CaC&Dev; I	MOD&CaC; I

Tabla 5.9 Bus atómico y protocolo MLI: Tabla de estados y transiciones entre estados.

Representación de transacciones y transiciones entre estados

En este apartado se muestran dos formas de representar las transacciones de bus y transiciones entre estados al ejecutar una secuencia de accesos a memoria: a) en formato tabla y b) mediante un diagrama temporal. Finalmente se muestra, mediante varios gráficos, una animación.

Representación en formato tabla

Utilizaremos una tabla, donde cada fila representa un acceso a memoria y no se gestiona el siguiente acceso hasta que ha finalizado el anterior. En una fila, de izquierda a derecha, después de la instrucción de acceso a memoria, se especifica:

1. El estado transitorio, si es el caso.
2. La transacción de bus y la señal MOD, si es el caso.
3. El nombre de la variable y el valor en memoria.
4. Quién suministra el bloque (cache o memoria).
5. Para las cache donde se modifica la información almacenada, el nombre de la variable, el valor y el estado estable del bloque.

Cuando es necesario mostrar un reemplazo se utilizan dos filas consecutivas. En la primera fila se indica el bloque que se expulsa y en la segunda fila el acceso a memoria que determina la expulsión del bloque.

Ejemplo. En la Tabla 5.10 se muestra una secuencia de accesos a memoria realizada por tres procesadores. Para rellenar la tabla suponemos que la variable t no está almacenada en ninguna cache y el valor en memoria es dos.

	C 1	C 2	C 3	Bus		mem.			C 1			C 2			C 3		
acceso	est.	est.	est.	trans.	señal	var.	val.	sum.	var.	val.	est.	var.	val.	est.	var.	val.	est.
1. P1 load t	IL			Pt		t	2	mem.	t	2	L						
2. P3 load t			IL	Pt		t	2	mem							t	2	L
3. P3 store t (21)			LM	PtIm		t	2	mem	t	2	I				t	21	M
4. P1 load t	IL			Pt	MOD	t	21	C3	t	21	L				t	21	L
5. P2 store t (8)		IM		PtIm		t	21	mem	t	21	I	t	8	M	t	21	I

Tabla 5.10 Bus atómico. Protocolo MLI. Secuencia de accesos a memoria: transacciones y cambios de estado en las caches.

Los dos primeros accesos son instrucciones load y producen fallos en cache. El estado transitorio del bloque en las caches C1 y C3 es IL y el estado final es L. El siguiente acceso es un acierto, pero es una escritura y hay que obtener el acceso exclusivo al bloque. Se solicita el bloque con intención de modificación y la copia en la cache C1 será invalidada. El estado transitorio del bloque en la cache del procesador P3 es LM y el estado final es M.

Seguidamente, en el procesador P1 se produce un fallo de cache debido a una instrucción load. El bloque lo tiene en exclusividad la cache del procesador P3. Por tanto, la cache C3 es la encargada de suministrar el bloque en la transacción de bus iniciada por el CC1. Por tanto, el CC3 activa la señal MOD. La memoria se actualiza utilizando el bloque suministrado en la transacción. El estado transitorio del bloque en la cache del procesador P1 es IL y el estado final L. El estado final del bloque en la cache del procesador P3 al finalizar la transacción es L.

Finalmente el procesador P2, que no tiene almacenado el bloque en cache, ejecuta una instrucción store. Ello da lugar a que se invaliden las copias en las caches de los otros procesadores. En este caso el bloque lo suministra memoria. El estado transitorio del bloque en la cache del procesador P2 es IM y el final estado final es M.

Diagrama temporal

El diagrama temporal y la representación de una transacción son como en el Capítulo 4.

En el segundo grupo de filas del final de la tabla también se representa el estado transitorio al solicitar acceso al bus. En la fase arb de una transacción se especifica el estado transitorio del bloque. Al finalizar la transacción se especifica el estado estable del bloque.

Cuando en una transacción se produce una transferencia entre caches (CaC), se indica en la fase posterior a ROb mediante el acrónimo CX, donde X es el indentificador numérico del CC que suministra el bloque.

El cambio de estado de un bloque que se expulsa y que no determina una transacción se indica en la fase arb de la petición que produce el conflicto.

Una expulsión, que requiere una transacción, se especifica en la columna etiquetada como accesos. Para ello, se utilizan dos filas contiguas. En la primera fila se especifica la expulsión (PtX) y en la segunda fila la petición que determina la expulsión.

Ejemplo. Para la secuencia de accesos a memoria de la Tabla 5.10, en la Figura 5.20 se muestra un diagrama donde se observa la secuencia de transacciones de bus con las fases.

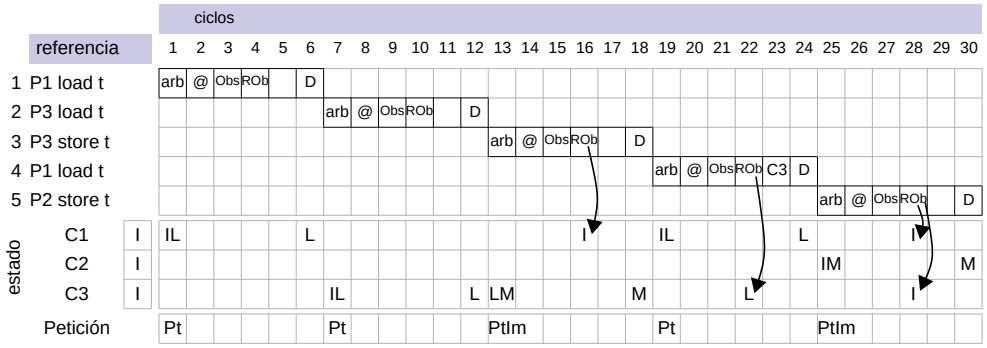


Figura 5.20 Bus atómico. Protocolo MLI. Secuencia de accesos a memoria y diagrama temporal y cambios de estado.

Animación

En la Figura 5.21 se muestra, mediante fotogramas, una animación de la secuencia de accesos a memoria de la Tabla 5.10. Además de las peticiones y respuestas se muestran los cambios de estado utilizando diagramas de transición entre estados. Cuando un bloque no está almacenado en un contenedor de cache se supone que está en estado inválido. Para reducir la información representada en la figura, no se muestran los casos en que un agente observador no modifica el estado al observar una transacción. Tampoco se muestra el estado transitorio MI, ya que no se utiliza en esta secuencia de accesos a memoria.

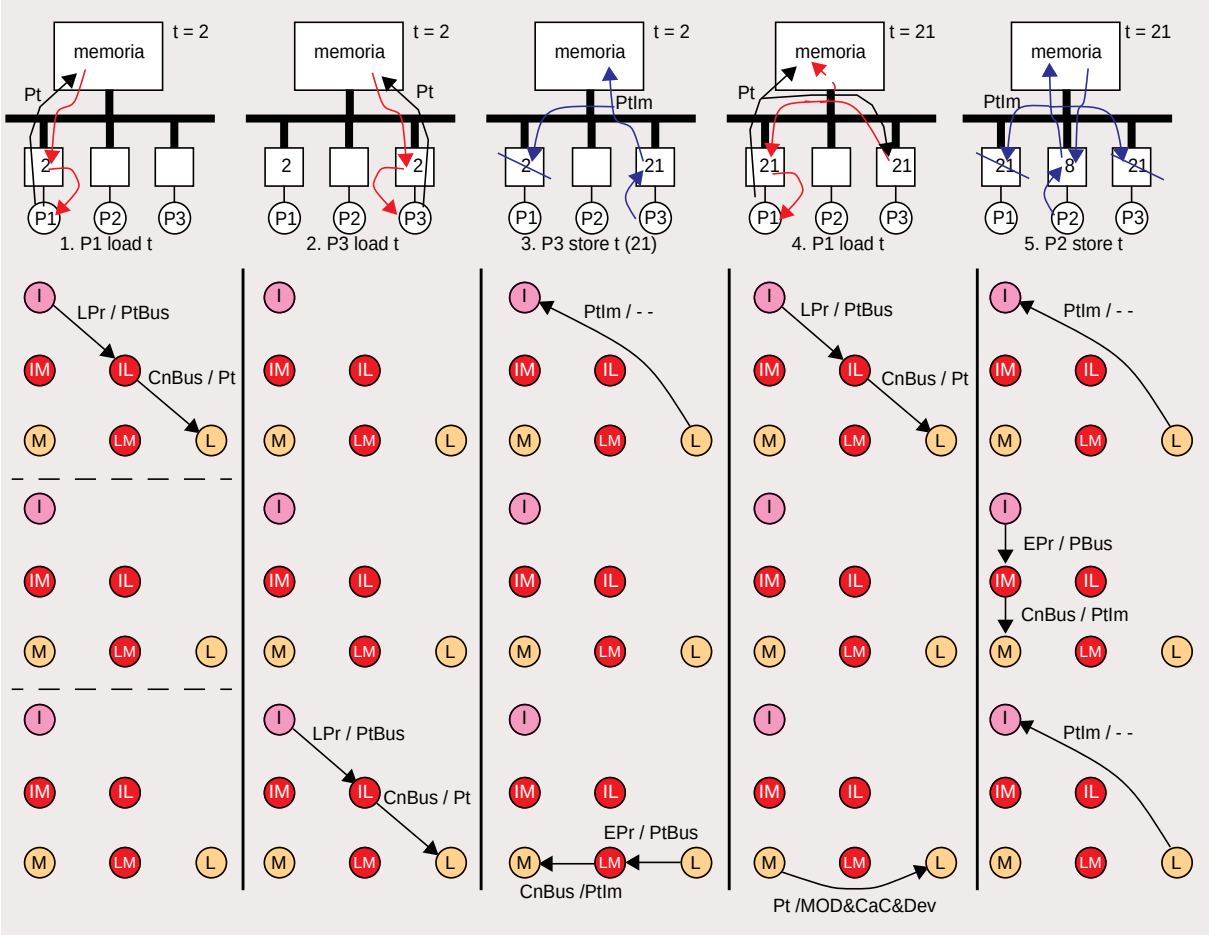


Figura 5.21 Bus atómico. Protocolo MLI. Secuencia de accesos a memoria. Fotogramas de las peticiones y respuestas y cambios de estado.

Representación en formato tabla de peticiones concurrentes

En una secuencia se agrupan los accesos a memoria y la agrupación se muestra mediante líneas horizontales continuas. Usualmente se agrupan de dos en dos.

Cada grupo de peticiones se gestiona independientemente. No se empieza a gestionar el siguiente grupo de peticiones hasta que ha finalizado el anterior. Cuando los accesos a memoria de un grupo necesitan acceder al bus se supone que las acciones de petición se realizan concurrentemente en el mismo ciclo.

En una fila de la tabla, de izquierda a derecha, después de la instrucción de acceso a memoria se especifica:

1. El orden de concesión del bus a la petición (1ª, 2ª, . . .). Supondremos usualmente que en un grupo de peticiones, el bus se concede en el orden en el cual se especifican los accesos a memoria en la secuencia de accesos.
2. El estado transitorio al efectuar la petición de bus.
3. La petición que ocupa el bus en la columna correspondiente al orden de concesión.
4. El nombre de la variable y el valor en memoria.
5. Quién suministra el dato o bloque (cache o memoria).
6. Para las caches donde se modifica la información almacenada, el nombre de la variable, el valor y el estado estable o transitorio del bloque.

Cuando es necesario mostrar un reemplazo se utilizan dos filas consecutivas. En la primera fila se indica el bloque que se expulsa y en la segunda fila el acceso a memoria que determina la expulsión del bloque.

Una expulsión que no se efectúa, ya que se ha suministrado el bloque, se representa indicando el estado I en la columna correspondiente al estado transitorio. Esta representación es similar a la expulsión de un bloque en el estado L.

Ejemplo. Una secuencia de accesos a memoria referencia las variables u y t que están contenidas en bloques distintos. Estos bloques al almacenar en cache se ubican en el mismo contenedor. Los bloques no están inicialmente almacenadas en cache y los valores de las variables en memoria son u = 7 y t = 2.

Los dos primeros accesos concurrentes son fallos de lectura. El CC1 obtiene la concesión del bus en primer lugar. El estado transitorio de los bloques en los dos casos es IL y el estado estable final es L.

El segundo grupo de accesos concurrentes son fallos de escritura y existe un conflictos en las dos caches. Como los bloques que se expulsan están en estado L no es necesario efectuar ninguna transacción, memoria tiene una copia válida de los bloques. La acción de expulsión la realizan los dos CC antes de efectuar la petición de bus. En la tabla se observa una fila dedicada a este hecho en los dos accesos a memoria (primera fila en cada acceso a memoria, CcRe). El CC1 obtiene la concesión del bus en primer lugar. El estado transitorio del bloque en los dos accesos a memoria (P1 y P2) es IM. Cada acceso actualiza una variable distinta y la petición es con intención de modificación. Ninguna de las dos transacciones invalida ningún bloque.

	arb.	C 1	C 2	C 3	Bus (trans.)			mem.			C 1			C 2		
acceso	ord.	est.	est.	est.	1º	2º	señal	var.	val.	sum.	var.	val.	est.	var.	val.	est.
1. P1 load t	1º	IL			Pt			t	2	mem.	t	2	L			
1. P2 load u	2º		IL			Pt		u	7	mem				u	7	L
2. P1 CcRe t		I									t	2	I			
store u (9)	1º	IM			PtIm			u	7	mem	u	9	M			
2. P2 CcRe u		I												u	7	I
store t (1)	2º		IM			PtIm		t	2	mem				t	1	M
3. P2 PtX t	1º	MI				PtX		t	1	C2				t	1	I
load u	2º	IL			Pt		MOD	u	9	C1	u	9	L	u	9	L
3. P1 load u																
4. P1 store u (12)	1º	LM			PtIm			u	9	mem	u	12	M	u	9	LM
4. P2 store u (3)	2º		LM			PtIm	MOD	u	12	C1	u	12	I	u	3	M

Tabla 5.11 Bus atómico. Protocolo MLI. Formato tabla: transacciones y cambios de estado del bloque en las caches en una secuencia de accesos a memoria concurrentes.

En el tercer grupo de accesos concurrentes hay un fallo de lectura y un acierto de lectura. El CC2 debe expulsar un bloque actualizado, debido a un conflicto en cache. Suponemos que las dos peticiones de bus (PtX y Pt) obtienen de forma consecutiva el bus. La expulsión de bloque y la petición de bloque por parte del CC2 se muestran en dos filas consecutivas. En la transacción iniciada por el CC2, donde se solicita un bloque, la memoria no está actualizada y el bloque lo suministra la cache C1. Al efectuar el suministro, el estado del bloque en la cache C1 pasa del estado M al estado L. Respecto al acierto de lectura no se indica ninguna información en la fila ya que no se efectúa ninguna transacción.

El cuarto grupo de accesos concurrentes son aciertos de escritura, siendo el estado del bloque L en las dos caches. La petición de los dos controladores de coherencia (CC1 y CC2) es lectura de bloque con intención de modificación y el estado transitorio es LM. La concesión de bus la obtiene en primer lugar CC1. El estado estable en la cache C1 al finalizar la transacción es M y el estado transitorio en la cache C2 es LM. Notemos que la transacción y la petición pendiente referencian la misma variable. Por tanto, el CC2 debe tener en cuenta la ordenación determinada por el bus. Como la petición pendiente obtendrá como respuesta el bloque, no se modifica el estado transitorio. En la transacción iniciada por CC2 la cache C1 suministra el bloque y activa la señal

correspondiente para que la memoria se inhiba (MOD). La transacción del CC2 invalida la copia en la cache de C1. Al finalizar la transacción iniciada por el CC2 el estado estable del bloque en la cache C1 es I y en la cache C2 es M.

Diagrama temporal de peticiones concurrentes

En un grupo de peticiones suponemos que se solapa la fase de arbitraje de la próxima transacción con la finalización de la transacción previa.

En un diagrama temporal, en la fase arb se especifica el estado transitorio y en la fase D se especifica el estado final en una transacción que ocupa el bus.

El cambio de estado de un bloque que se expulsa y que no determina una transacción se indica en la fase arb de la petición que produce el conflicto.

Una expulsión que requiere una transacción se especifica en la columna etiquetada como accesos. Para ello, se utilizan dos filas contiguas. En la primera fila se especifica la expulsión (PtX) y en la segunda fila la petición que determina la expulsión.

En el diagrama temporal, la petición de expulsión es la primera que solicita el bus. Una vez ha obtenido el bus se indica la solicitud de la petición que ha determinado la expulsión. Una expulsión que no se efectúa, ya que se ha suministrado el bloque, deja de solicitar el arbitraje. La desactivación de la señal se efectúa en el ciclo siguiente a la fase ROb de la transacción que ha requerido el suministro. En este mismo ciclo se empieza a indicar el arbitraje de la petición que ha determinado la expulsión.

En un diagrama temporal un acierto en cache se indica con la letra A.

Ejemplo. En las Figura 5.15 se muestra un diagrama temporal de la ocupación del bus y de los cambios de estado en una transacción. Las expulsiones que determinan el segundo grupo de acceso a memoria no requieren efectuar una transacción. Por tanto, no se utilizan filas para mostrarlo. El cambio de estado del bloque expulsado se indica en la fase arb de la petición que determina la expulsión. En cambio, en el tercer grupo de accesos a memoria se efectúa una expulsión de un bloque que ha sido actualizado en cache. La petición que determina el reemplazo empieza a solicitar el arbitraje después de que la petición de expulsión haya obtenido el bus. El acceso del procesador P1 es un acierto en cache.

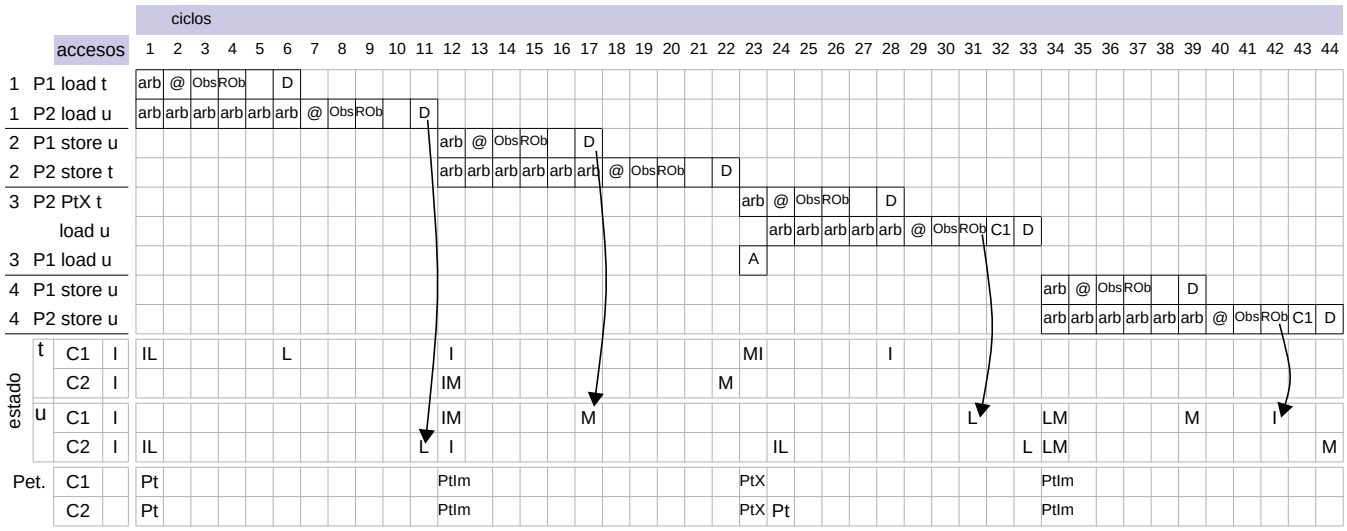


Figura 5.22 Bus atómico. Protocolo MLI. Diagrama temporal de una secuencia de accesos concurrentes.

Verificación no formal de coherencia y consistencia

La base es la descripción efectuada en el Capítulo 4. La diferencia se centra en los estados transitorios de un bloque en cache. Durante la ventana temporal en el CC se memoriza, localmente y por bloque, la observación de una escritura que haya ordenado previamente el árbitro del bus. En particular, en el protocolo MLI no es necesario memorizar ninguna ordenación de peticiones, ya que la petición PtX, correspondiente a una expulsión de bloque no requiere ser ordenada. Es para actualizar memoria. Esto es, mantener coherencia en la jerarquía observada por el procesador.

Por otro lado, durante la ventana de vulnerabilidad de una expulsión es necesario suministrar el bloque, para mantener la coherencia de cache. Una lectura obtiene el valor de la última escritura consolidada (atomicidad de las escrituras). Las escrituras a un bloque en el estado M están consolidas. Además, el bloque puede haber sido leído por el procesador cuya cache expulsa el bloque. Hay que garantizar que el orden de programa del hilo que ejecuta el procesador es observado por otro hilos.

APENDICE A: REDUCCIÓN DE LA LATENCIA DE UN FALLO

Cuando se produce una acción de reemplazo y el bloque que se expulsa está en el estado M hay que actualizar memoria.

Hasta ahora hemos supuesto que la actualización de memoria se efectúa antes de iniciar el servicio del fallo de cache (Figura 5.23). Esta forma de actuar determina que la latencia de servir el fallo de cache sea, como mínimo, el doble.

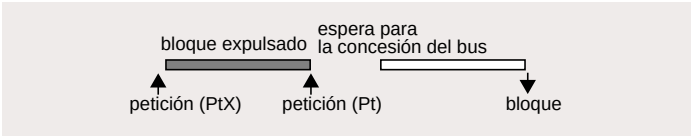


Figura 5.23 Reemplazo: actualización de memoria antes de servir el fallo.

Una forma de reducir la latencia de servicio de un fallo de cache es almacenar en un buffer el bloque expulsado e iniciar inmediatamente el servicio del fallo (Figura 5.24). Posteriormente se actualiza memoria con el bloque reemplazado.

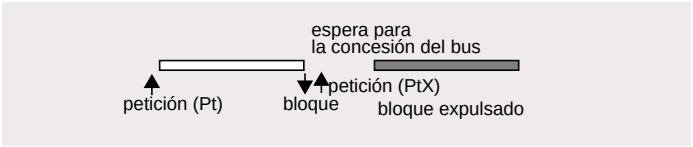


Figura 5.24 Reemplazo: actualización de memoria posterior al servicio del fallo.

El bloque expulsado se almacena en un buffer, que se denomina buffer de expulsiones (BEX). En el buffer se distinguen los mismos campos que en la cache: dirección, estado y datos (Figura 5.25).

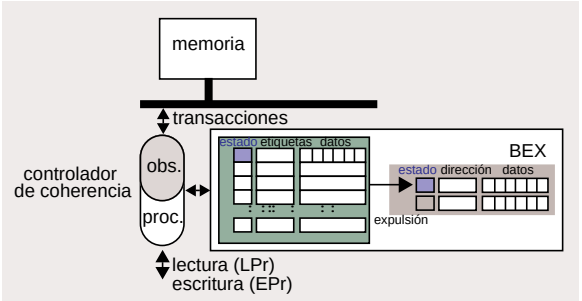


Figura 5.25 Cache con buffer de expulsiones (BEX).

Notemos que se almacena la dirección del bloque, la cual se obtiene concatenando los bits leídos del campo etiqueta con los bits utilizados para determinar el conjunto de cache. El BEX, desde el punto de vista de coherencia, debe observarse como una extensión de la cache. Esto es, los dos agentes, procesador y observador, tienen que comprobar si el bloque al que están accediendo está almacenado en el BEX.

El agente observador debe suministrar el bloque en una transacción de bus que acceda al bloque. En una acción de suministro del bloque, si no se ha solicitado acceso al bus (PtBus) no es necesario actualizar memoria explícitamente y la entrada del BEX queda libre. Si se ha solicitado acceso al bus, después de efectuar el suministro, el CC desactiva la solicitud de acceso al bus. En la Figura 5.26 se muestra el diagrama de transición entre estados de un bloque almacenado en el BEX[20].

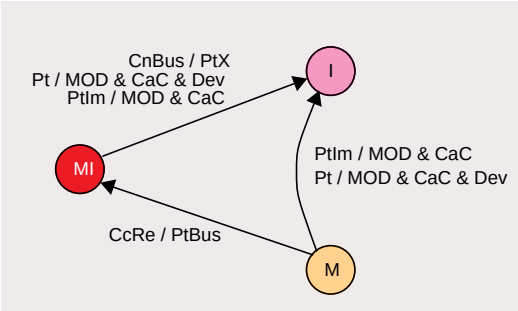


Figura 5.26 Diagrama de transiciones entre estados del buffer de expulsiones.

Usualmente el tamaño del BEX permite almacenar más de un bloque y la organización es totalmente asociativa. El BEX se intenta vaciar cuando no hay actividad de peticiones de bus en el procesador, cuando está lleno y es necesario otro reemplazo o cuando el número de entradas ocupadas llega a una marca, que usualmente se denomina marca de agua.

La posición de memoria de un acceso del procesador puede estar contenida en un bloque almacenado en el BEX[21]. Una alternativa es bloquear el procesador y actualizar memoria con el bloque referenciado. Después de actualizar memoria, se gestionará un fallo de cache[22].

Otra alternativa es disponer de circuitería en el BEX para suministrar el valor al procesador (load) y circuitería para poder actualizar una entrada en el BEX (store).

20. En los estados M y MI la copia del bloque en el BEX es válida. Además, es la única copia válida en el multiprocesador.
21. Hay que respetar las dependencias de datos del hilo que se está ejecutando en el procesador.
22. Esta serialización la gestiona el controlador de coherencia.

EJEMPLOS

Protocolo MLI

Una secuencia de accesos a memoria referencia las variables u y t que están contenidas en bloques distintos. Estos bloques al almacenar en cache se ubican en el mismo contenedor. Los bloques no están inicialmente almacenadas en cache y los valores de las variables en memoria son u = 9 y t = 12.

Pregunta 1: *Utilice una tabla similar a la Tabla 5.11 para mostrar las transacciones de bus y estados de los bloques en las caches. La secuencia de accesos a memoria se indica en la tabla mostrada en la respuesta. En esta secuencia se agrupan los accesos a memoria de dos en dos y la agrupación se muestra mediante líneas horizontales continuas.*

Respuesta: Los dos primeros accesos concurrentes son fallos de escritura. El CC1 obtiene la concesión del bus en primer lugar. El estado transitorio de los bloques en los dos casos es IM y el estado estable final es M.

	arb.	C 1	C 2	C 3	Bus (trans.)					mem.			C 1			C 2		
accesos	ord.	est.	est.	est.	1º	2º	3º	4º	señal	var.	val.	sum.	var.	val.	est.	var.	val.	est.
1. P1 store t (2)	1º	IM			PtIm					t	9	mem.	t	2	M			
1. P2 store u (7)	2º		IM			PtIm				u	12	mem				u	7	M
2. P1 PtX t	1º	MI			PtX					t	2	C1	t	2	I			
load u	2º	IL				Pt			MOD	u	7	C2	u	7	L	u	7	I
2. P2 PtX u			I													u	7	I
load t	3º		IL					Pt	MOD	t	2	mem				t	2	L
3. P2 CcRe t			I													t	2	I
load u	1º		IL		Pt					u	7	mem				u	7	L
3. P1 load u	2º																	
4. P1 store u (12)	1º	LM			PtIm					u	7	mem	u	12	M	u	7	LM
4. P2 store u (3)	2º		LM			PtIm			MOD	u	12	C1	u	12	I	u	3	M

El segundo grupo de accesos concurrentes son fallos de lectura y existe un conflictos en las dos caches. Como los bloques que se expulsan están en el estado M, es necesario efectuar una transacción para actualizar memoria. En la tabla se observa una fila dedicada a este hecho en los dos accesos a memoria (primera fila en cada acceso a memoria). El CC1 obtiene la concesión del bus en primer lugar. Al finalizar la transacción de expulsión del bloque en la cache C1 el estado es I. Suponemos que la segunda petición en obtener el bus

también es del procesador P1. Durante la transacción el CC2 observa que el bloque referenciado es el que quiere expulsar, lo suministra, inhibe el suministro de memoria y cambia el estado del bloque al estado I (cuarta fila en la tabla, columna est.). Además, memoria se actualiza con el bloque que se transfiere por el bus. También, el CC2 desactiva la petición de bus cuyo objetivo era actualizar memoria. En la columna etiquetada como estado transitorio se indica que el estado del bloque es I (quinta fila). Seguidamente el CC2 obtiene el bus para leer el bloque que almacena la variable t. Al finalizar la transacción el estado del bloque es L.

En el tercer grupo de accesos concurrentes hay un fallo de lectura y un acierto de lectura. El CC2, debido a un conflicto en cache, debe expulsar un bloque que no ha sido actualizado. Como memoria está actualizada la expulsión no determina una transacción. La expulsión del bloque y la petición de bloque por parte del CC de P2 se muestran en dos filas consecutivas. En la transacción emitida por el CC2 responde memoria. Respecto al acierto de lectura no se indica ninguna información en la fila, ya que no se efectúa ninguna transacción.

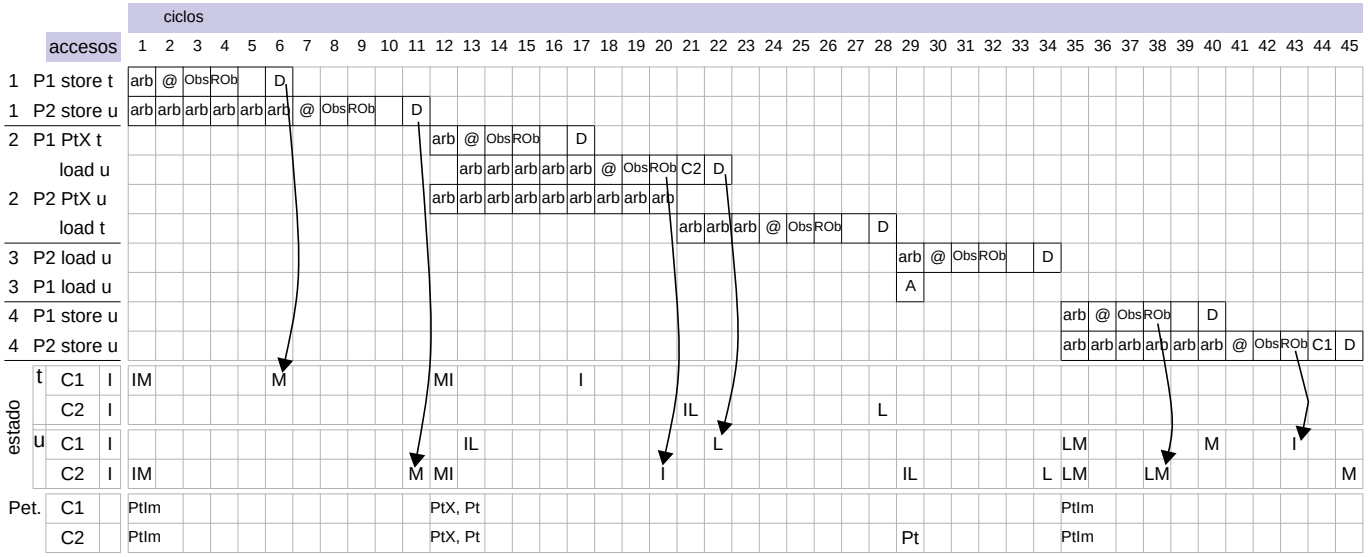
El cuarto grupo de accesos concurrentes son aciertos de escritura, siendo el estado del bloque L en las dos caches. La petición de los dos controladores de coherencia (CC1 y CC2) es lectura de bloque con intención de modificación y el estado transitorio es LM. La concesión de bus la obtiene en primer lugar el CC1. El estado estable en la cache C1 al finalizar la transacción es M y el estado transitorio en la cache C2 es LM. Notemos que la transacción y la petición pendiente referencian la misma variable. Por tanto, el CC2 debe tener en cuenta la ordenación determinada por el bus. Como la petición pendiente obtendrá como respuesta el bloque, no se modifica el estado transitorio. En la transacción iniciada por el CC2, la cache C1 suministra el bloque y activa la señal correspondiente para que la memoria se inhiba. La transacción del CC2 invalida la copia en la cache de C1. Al finalizar la transacción iniciada por el CC2, el estado estable del bloque en la cache C1 es I y en la cache C2 es M.

Pregunta 2: *Muestre en un diagrama temporal la ocupación del bus y los cambios de estado de los bloques en las caches.*

Respuesta: En las siguiente figura se muestra un diagrama temporal de la ocupación del bus y de los cambios de estado.

En el segundo grupo, la petición de P1, que determina la expulsión de un bloque, solicita el bus después de que haya obtenido el bus la petición de expulsión (PtX). La segunda expulsión, en el segundo grupo de accesos a memoria, no requiere una transacción de bus, ya que el bloque ha sido

suministrado en la transacción previa del mismo conjunto de accesos a memoria. Por tanto, el CC2 desactiva la petición de acceso al bus. En el siguiente ciclo el CC2 activa una solicitud de acceso a bus para obtener el bloque que contiene la variable t.



Buffer de expulsiones

Una secuencia de accesos a memoria referencia las variables u y t que están contenidas en bloques distintos. Estos bloques al almacenar en cache se ubican en el mismo contenedor. Los bloques no están inicialmente almacenadas en cache y los valores de las variables en memoria son u = 9 y t = 12.

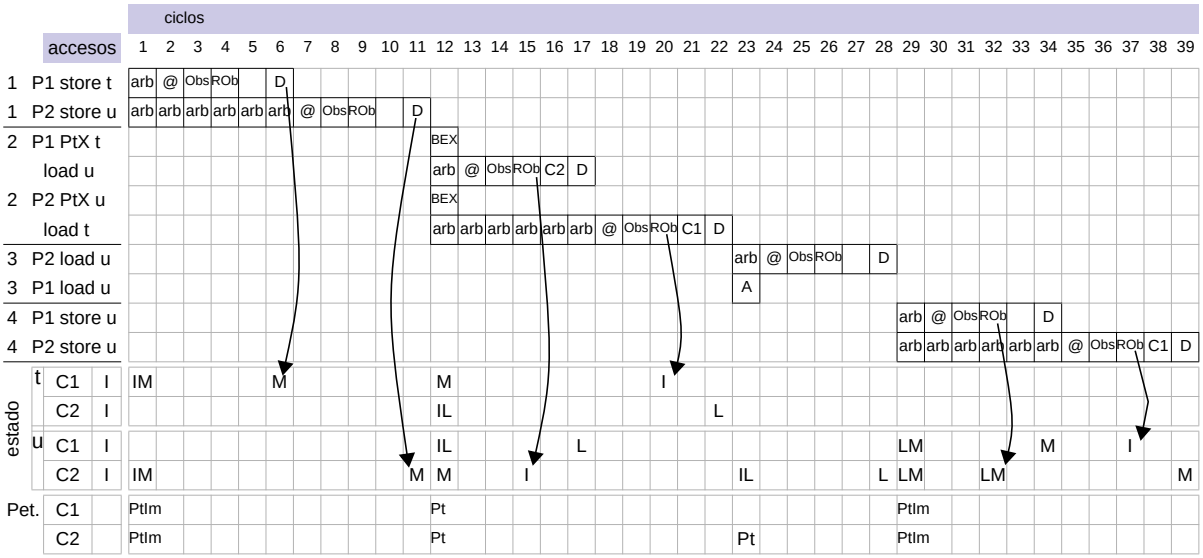
Utilice el protocolo de invalidación con escritura retardada (MLI) para mostrar las transacciones de bus y estados de los bloques en las caches. Cada cache dispone de un BEX para reducir la latencia de fallo cuando el contenedor asignado al bloque contiene un bloque en el estado M.

La secuencia de accesos a memoria se indica en el diagrama temporal mostrado en la respuesta. En esta secuencia se agrupan los accesos a memoria de dos en dos y la agrupación se muestra mediante líneas horizontales continuas.

El almacenamiento del bloque en el BEX se especifica mediante el acrónimo BEX. Cuando se almacena un bloque en el BEX no se solicita acceso al bus.

Pregunta 1: *Muestre en un diagrama temporal la ocupación del bus y los cambios de estado de los bloques en las caches.*

Respuesta: En las siguiente figura se muestra un diagrama temporal de la ocupación del bus y de los cambios de estado. Notemos que los bloques almacenados en el BEX se invalidan antes de efectuar una petición de expulsión. Al almacenar los bloques en el BEX no ha sido solicitado el bus. En el segundo grupo de accesos, el suministro de los bloques se efectúa desde los BEX.



EJERCICIOS

Descripción de un protocolo de observación VI denominado A

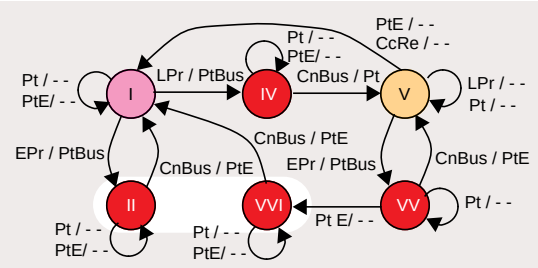
Un multiprocesador utiliza un bus como red de interconexión. Las caches privadas utilizan escritura inmediata, sin asignación de contenedor en fallo de escritura. El multiprocesador utiliza la técnica de invalidación para mantener la coherencia. En cada contenedor de cache se identifican dos posibles estados estables: inválido (I) y válido (V) y cuatro estado transitorios.

Las peticiones de procesador y las transacciones de bus son las siguientes.

Procesador	Controlador de cache	
Peticiones	**Transacciones**	**Acciones**
LPr: lectura del proc.	Pt: petición de bloque	CcRe: reemplazo de un bloque
EPr: escritura del proc.	PtE: petición de escritura de un dato	

En este multiprocesador no se puede considerar una acción atómica la detección de necesitar efectuar una transacción de bus y la realización de la misma. Un CC compite con otros CC para obtener el bus. Mientras un CC está pendiente de que se le conceda el bus, el CC debe poder efectuar las acciones de observación inducidas por la transacción en curso en el bus.

Por ello, hay que distinguir la acción de solicitar el bus (PtBus) y la acción de concesión del bus (CnBus). En el siguiente diagrama de estados se muestran todas las transiciones entre estados estables y estados transitorios.



Un procesador inicia los accesos a memoria en orden de programa y se bloquea en un acceso a memoria cuando el procesador tiene una petición pendiente.

Descripción de un protocolo de observación MLI denominado B

Un multiprocesador utiliza un bus como red de interconexión. Las caches privadas utilizan escritura retardada, con asignación de contenedor en fallo. El multiprocesador utiliza la técnica de invalidación para mantener la coherencia.

En cada contenedor de cache se identifican tres posibles estados estables: inválido (I), lectura (L) y modificado (M). Además, se utilizan cuatro estados transitorios.

Las peticiones de procesador y las transacciones de bus son las siguientes.

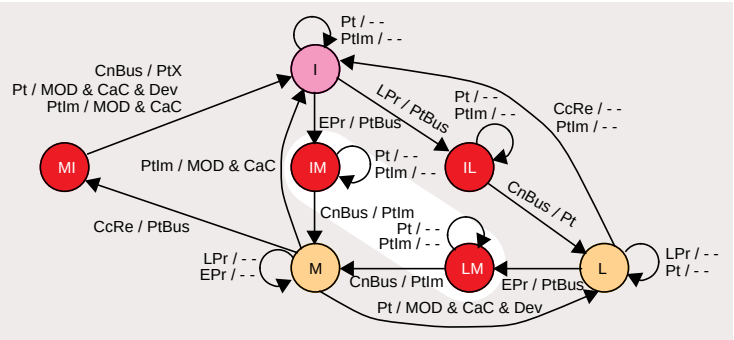
Procesador	Controlador de cache		Memoria
Peticiones	**Transacciones**	**Acciones**	**Acciones**
LPr : lectura del proc.	Pt : petición de bloque	CcRe: reemplazo de un bloque	Dev: almacenar en memoria
EPr : escritura del proc.	PtIm: petición de bloque con intención de modificarlo	MOD: señal que indica bloque en estado M en una cache	
	PtX: actualización de memoria	CaC: suministro del bloque	

Una cache puede suministrar directamente el dato dentro de una transacción de bus iniciada por otro procesador (CaC). Además, en este caso se actualiza memoria, si lo determina el protocolo. Cuando una cache tiene un bloque en estado M activa la señal denominada MOD. En el bus se dispone de un cable que es la función OR de las señales MOD de cada una de las caches. Esta señal inhibe el suministro de memoria.

En este multiprocesador no se puede considerar una acción atómica la detección de necesitar efectuar una transacción de bus y la realización de la misma. Un CC compite con otros CC para obtener el bus. Mientras un CC está pendiente de que se le conceda el bus, el CC debe poder efectuar las acciones de observación inducidas por la transacción en curso en el bus, y en su caso suministrar un bloque de cache.

Por ello, hay que distinguir la acción de solicitar el bus (PtBus) y la acción de concesión del bus (CnBus). El el estado MI, cuando se suministra el bloque, al observar una transacción, se desactiva la solicitud de arbitraje.

En el siguiente diagrama de estados se muestran todas las transiciones entre estados estables y estados transitorios.



Cuando es necesario efectuar una acción de reemplazo, primero se actualiza memoria, si es el caso. Posteriormente se efectuan las acciones relacionadas con el acceso que determina el reemplazo.

Un procesador inicia los accesos a memoria en orden de programa y se bloquea en un acceso a memoria cuando el procesador tiene una petición pendiente.

Descripción de un protocolo de observación MLI denominado C

Un multiprocesador utiliza un bus como red de interconexión y el protocolo de coherencia es del tipo invalidación, con escritura retardada e intervención directa con actualización de memoria. Esto es, cuando es necesario, un CC suministra el bloque (CaC) en los ciclos correspondientes de la transacción y en paralelo se actualiza la memoria, si es el caso.

Las cache privadas utilizan escritura retardada con asignación de contenedor en fallo. En cada contenedor de cache se dispone de dos bits para identificar los tres posibles estados: Inválido (I), Lectura (L) y Modificado (M).

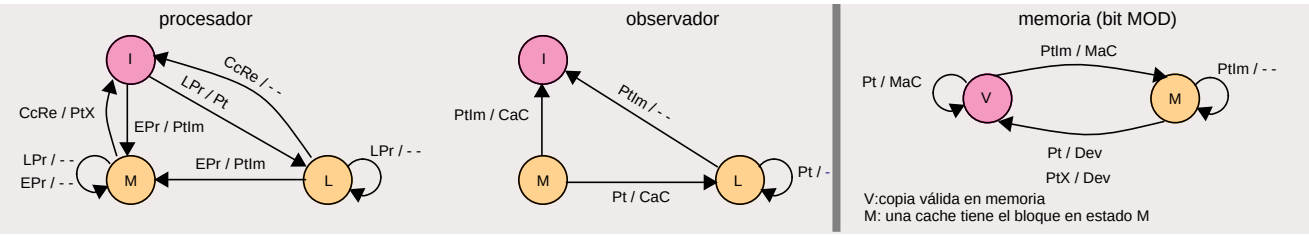
Cada bloque de memoria tiene asociado un bit que indica si una cache tiene el bloque en estado modificado (bit MOD). Este bit se utiliza para conocer si una cache suministrará el bloque. En este caso, memoria se inhibe del suministro.

Cuando un CC detecta que tiene una copia actualizada (estado M) suministra el bloque en los ciclos correspondientes de la transacción (CaC). En paralelo se actualiza la memoria (Dev), si es el caso. En los otros casos el bloque lo suministra la memoria (MaC).

Las peticiones del procesador, las transacciones de bus y las acciones son las siguientes.

Procesador	Controlador de coherencia (CC)		Memoria
Peticiones	**Transacciones**	**Acciones**	**Información / acciones**
LPr: lectura del proc.	Pt: petición de bloque	CcRe: reemplazo de un bloque	Dev: actualización de memoria
EPr: escritura del proc.	PtIm: petición de bloque con intención de modificación	CaC: suministro del bloque	bit MOD: bloque en estado M en una cache
	PtX: petición para actualizar memoria		MaC: suministro desde memoria

Los diagramas de transiciones entre estados en la cache y en memoria son los siguientes:



Cuando es necesario efectuar una acción de reemplazo, primero se actualiza memoria, si es el caso. Posteriormente se efectuan las acciones relacionadas con el acceso que determina el reemplazo. Un procesador inicia los accesos a memoria en orden de programa.

Ejercicio 5.1

El protocolo de coherencia que se utiliza es el denominado B en este capítulo.

Una secuencia de accesos a memoria referencia las variables u y t que están contenidas en bloques distintos. Estos bloques al almacenar en cache se ubican en el mismo contenedor. Las variables no están inicialmente almacenadas en cache y los valores en memoria son u = 9 y t = 12.

Suponga la siguiente secuencia de accesos. Las líneas más oscuras se utilizan para identificar agrupaciones de accesos concurrentes.

accesos	accesos	accesos	accesos
1. P1 store u	2. P1 load t	3. P1 store t	4. P2 store t
1. P2 store t	2. P2 store t	3. P2 load u	4. P1 store u

Pregunta 1: *Muestre en un diagrama temporal la ocupación del bus y los cambios de estado de los bloques en las caches al ejecutarse la anterior secuencia de accesos a memoria.*

Pregunta 2: *Indique el número total de reemplazos y el número de ellos que requieren actualizar memoria. De estos últimos, indique en cuántos el CC desactiva la petición de bus, ya que el CC ha suministrado el bloque para servir una petición en el bus.*

En una versión posterior de los procesadores, utilizados en el multiprocesador, se añade un buffer de expulsiones (BEX). El diagrama de estados de un bloque en el BEX es un subconjunto del diagrama de estados (estados M, I y MI), añadiendo una transición del estado M al estado I cuando se observa una petición Pt o PtIm.

El almacenamiento de un bloque en el BEX se especifica mediante el acrónimo BEX (su duración es una fase de una transacción de bus) y el CC no solicita acceso al bus al efectuarse el almacenamiento en el BEX.

Pregunta 3: *Utilice la secuencia de accesos previa. Muestre en un diagrama temporal la ocupación del buffer de expulsiones, del bus y los cambios de estado de los bloques en las caches. Además, suponga que el tamaño del BEX es ilimitado y por tanto no es necesario vaciarlo antes de que finalice la secuencia.*

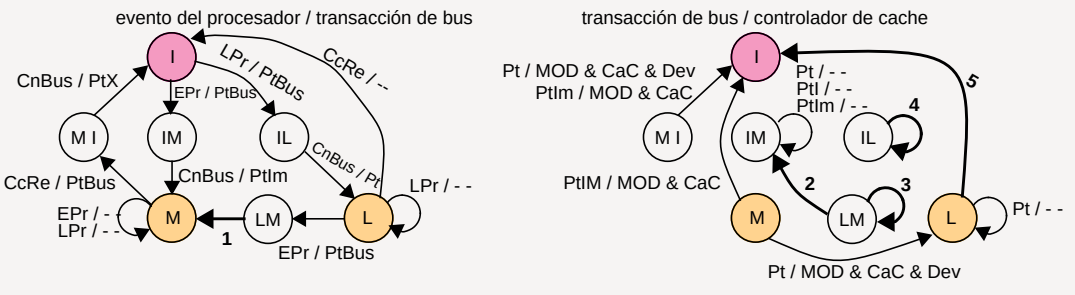
Pregunta 4: *Indique el número de bloques que han sido almacenados en el BEX y que permanecen en el BEX cuando finaliza la secuencia de accesos a memoria.*

Ejercicio 5.2

El protocolo de coherencia que se utiliza es el denominado B del Capítulo 4. Este protocolo se modifica con el mecanismo de transacción de exclusividad (PtI) descrito en el Capítulo 4.

En el protocolo descrito no se han especificado los estados transitorios que son necesarios para su implementación. Estos estados son necesarios debido a que no se puede considerar una acción atómica la detección de necesitar efectuar una transacción de bus y la realización de la misma. Un controlador de coherencia (CC) compite con otros CC para obtener el bus. Un CC, mientras está pendiente de que se le conceda el bus, debe poder efectuar las acciones de observación inducidas por la transacción en curso en el bus, y en su caso suministrar un bloque de cache.

En el diseño que analizamos se considera atómica una transacción de bus. Por ello hay que distinguir la acción de solicitar el bus (PtBus) y la acción de concesión del bus (CnBus). En el siguiente diagrama de estados se muestran todas las transiciones entre estados estables y estados transitorios (IL, IM, LM, MI) y se etiquetan la mayoría de ellas con los eventos y acciones.



Pregunta 1: *Indique en las transiciones entre estados que se han etiquetado con números los eventos y acciones.*

Pregunta 2: *Suponga que se decide no utilizar la petición PtI y en su lugar se utiliza la petición PtIm. Indique en las transiciones entre estados que se han etiquetado con números los eventos y acciones. En el caso de que no se pueda producir una transición indique "no se puede producir". Dada una petición en un CC, al tener en cuenta posibles observaciones, el número de cambios de estado debe ser el mínimo posible.*

Suponga que no se distingue entre tipos de petición al efectuar una transacción. Las fases de una transacción de bus son:

referencia	ciclos 1	2	3	4	5	6
Pt, PtIm, PtX	arb	@	Obs	ROb		D
PtI	arb	@	Obs	ROb		D

Una secuencia de accesos a memoria referencia las variables u y t que están contenidas en bloques distintos. Estos bloques al almacenar en cache se ubican en contenedores distintos. Las variables no están inicialmente almacenadas en cache y los valores en memoria son u = 9 y t = 12.

Suponga la siguiente secuencia de accesos. Las líneas más oscuras se utilizan para identificar agrupaciones de accesos concurrentes.

accesos
1. P1 load u
1. P2 load u
2. P1 store u
2. P2 store u

accesos
3. P1 load t
3. P2 store t
4. P1 load t
4. P2 load t

Pregunta 3: *Cuando se utiliza la petición PtI para obtener la exclusividad de acceso a un bloque en el estado L, muestre en un diagrama temporal la ocupación del bus y los cambios de estado de los bloques en las caches al ejecutarse la anterior secuencia de accesos a memoria. Así mismo, muestre las peticiones que un CC espera emitir y la petición que finalmente emite.*

Suponga que las fases de una transacción de bus son:

referencia	ciclos 1	2	3	4	5	6	7	8	9
Pt, PtIm, PtX	arb	@	Obs	ROb		D	D	D	D
PtI	arb	@	Obs	ROb					

Pregunta 4: *Utilice la duración de las transacciones en función de la petición que ocupa el bus. Para la secuencia de accesos mostrada en el enunciado, calcule la duración en ciclos en los siguientes dos casos. En una escritura del procesador, estando el bloque en el estado L, se utiliza: a) la petición PtI y b) la petición PtIm.*

Ejercicio 5.3

En algunas implementaciones de las operaciones adquirir y liberar, utilizadas para serializar el acceso a variables compartidas, se produce bastante tráfico cuando se libera la zona de exclusión y existen hilos que están esperando, utilizando espera activa, para acceder a las variables compartidas. Todos estos hilos están observando el valor de una única variable.

Supongamos un multiprocesador donde se utiliza una política de invalidación para mantener la coherencia de cache. En las implementaciones comentadas previamente, cuando un hilo libera la zona de exclusión se invalidarán todas las copias del bloque en las otras cache y posteriormente se producirá una ráfaga de peticiones del bloque invalidado. Sin embargo, sólo uno de los hilos obtendrá el acceso exclusivo y los otros hilos volverán a la fase de espera activa. Esta fase se denomina cambio de mano.

Una forma de reducir el tráfico en un cambio de mano es utilizar una variable y dos vectores cuyo número de entradas es igual al número de hilos. Los vectores se denominan índice y espera y la variable se denomina cola. El algoritmo que se describe seguidamente se denomina vector de llaves.

Cada hilo tiene asignada estáticamente una entrada en el vector índice a la que se accede utilizando un identificador (hid). Las entradas en el vector espera se asignan dinámicamente, a medida que los hilos pretenden acceder a la zona de exclusión. La entrada asignada en una acción adquirir es la indicada por la variable cola.

Seguidamente se muestra el código de las primitivas adquirir y liberar, donde se utiliza la operación atómica fetch&add en la primitiva adquirir.

fetch&add (dir, valor)	comentario
tmp = dir	dir es una variable global
dir = tmp + valor	valor: cantidad en la que se incrementa el valor almacenado en dir
return tmp	

En el diseño del código se ha efectuado la hipótesis de que posteriormente no se va a reutilizar la estructura de datos y que el vector espera tiene N+1 elementos. Los valores de la variable hid están dentro del rango [0 . . . N - 1], siendo N el número de hilos.

adquirir ( )	liberar ( )	Inicialización	hipótesis
indice [hid] = fetch&add (cola, 1)	espera [indice [hid]] = 0	espera [0 . . . N] = [1, 0, . . . 0, 0]	N es multiplo de 4
while (espera [indice [hid]] = 0) { }	espera [indice [hid] +1] = 1	indice [0 . . . N-1] = [0, 0, . . . 0, 0]	
		cola = 0	

Las entradas en el vector espera se asignan de forma contigua. Por tanto, todas las entradas previas a la indicada por la variable cola han sido asignadas a hilos que han solicitado el acceso exclusivo previamente.

Cuando un hilo pretende acceder a la zona de exclusión, el valor de la variable cola se almacena, en el vector índice, en la entrada asignada estáticamente al hilo.

Un hilo efectúa espera activa consultando el valor de la entrada del vector espera que le ha sido asignada dinámicamente.

Los códigos previos se ejecutan en un sistema multiprocesador que no garantiza consistencia secuencial. Para garantizarla utilizaremos la directiva LNbarrera en el código. Al insertar esta directiva entre dos sentencias se inhibe en el compilador la posibilidad de reordenar operaciones de acceso a memoria. El compilador no puede ubicar después de la primitiva LNbarrera operaciones de acceso a memoria que el programador ha especificado antes de la primitiva LNbarrera y viceversa. Desde el punto de vista hardware la directiva se traduce en una instrucción INbarrera, que garantiza que las operaciones de acceso a memoria, previas a la instrucción INbarrera, han sido consolidadas antes de iniciar la ejecución de las operaciones de acceso a memoria, especificadas después de la instrucción INbarrera.

Pregunta 1: *Indique las posiciones en las cuales hay que insertar la directiva LNbarrera para garantizar consistencia secuencial. Justifique la respuesta. El número de inserciones de la directiva LNbarrera debe ser el mínimo posible.*

El protocolo de coherencia que se utiliza es el denominado C en este capítulo.

El tamaño de un bloque es 16 bytes y un elemento de los vectores y la variable cola ocupan 8 bytes. Suponga que cuando se declara un vector o una variable el compilador ubica el vector o variable, en el espacio lógico, alineada a tamaño de bloque.

En las siguientes preguntas no se tienen en cuenta los bloques accedidos mientras se ocupa la zona de exclusión.

Suponga que el multiprocesador dispone de cuatro procesadores y hay cuatro hilos. El procesador P0 ocupa la zona de exclusión y el resto de procesadores (P1, P2, P3) están en espera activa. Los hilos han obtenido las entradas del vector espera en el mismo orden que tiene asignadas las entradas del vector índice. El número de entrada en el vector índice se corresponde con el ordinal que identifica al procesador.

Pregunta 2: *El primer bloque del vector espera se almacena en el conjunto cero de una cache. En las condiciones descritas previamente, indique los bloques del vector espera que están almacenados en la cache de cada procesador, los elementos del vector espera que contiene cada bloque, el conjunto de cache que ocupa y el estado del bloque en cada cache.*

Cuando se libera la zona de exclusión por el procesador P0 se observa la siguiente secuencia de accesos a memoria.

acceso	descripción
1. P0 store R0, 0(R1); espera [0]	Liberación y despertar. P0 libera el acceso exclusivo
2. P0 store R1, 8(R1); espera [1]	
3. P1 load R3, 0(R2); espera [1]	Comprobación, entrar y esperar. P1 obtiene el acceso exclusivo
4. P2 load R3, 0(R2); espera [2]	
5. P3 load R3, 0(R2); espera [3]	
6. P1 store R0, 0(R1); espera [1]	Liberación y despertar. P1 libera el acceso exclusivo
7. P1 store R1, 8(R1); espera [2]	
8. P2 load R3, 0(R2); espera [2]	Comprobación, entrar y esperar. P2 obtiene el acceso exclusivo
9. P3 load R3, 0(R2); espera [3]	
10. P2 store R0, 0(R1); espera [2]	Liberación y despertar. P2 libera el acceso exclusivo
11. P2 store R1, 8(R1); espera [3]	
12. P3 load R3, 0(R2); espera [3]	Comprobacióny entrar. P3 obtiene el acceso exclusivo
13. P3 store R0, 0(R1); espera [3]	Liberación. P3 libera el acceso exclusivo
14. P3 store R1, 8(R1); espera [4]	

Pregunta 3: *Muestre en una tabla el estado en cache de los bloques referenciados en la anterior secuencia de accesos a memoria. Así mismo, muestre el valor de las variables en cache y en memoria y las transacciones de bus en cada acceso de los procesadores, si es el caso. En la columna variable indique el índice del vector espera. Suponga que el estado de los bloques del vector espera que están almacenados en las caches, en las condiciones descritas antes de la pregunta anterior, es L.*

	bus	mem.				C 0				C 1				C 2				C 3			
acceso	trans.	bit	var.	val.	sum.	cjto	var.	val.	est.	cjto	var.	val.	est.	cjto	var.	val.	est.	cjto	var.	val.	est.
1. P0 store R0, 0(R1); espera [0]																					

Pregunta 4: *A partir de la secuencia de accesos mostrada previamente, calcule el número de bloques invalidados en las caches, el número de peticiones de exclusividad (el bloque está en estado L) y el número de fallos (excluyendo peticiones de exclusividad). Recuerde que todos los hilos menos uno están en espera activa. Además, muestre expresiones algebraicas cuando el número de procesadores e hilos es N en lugar de cuatro. Para ello identifique un patrón y efectúe el cálculo con este patrón. Suponga que N es par.*

En las siguiente preguntas y hasta que se indique lo contrario suponga que el multiprocesador mantiene consistencia secuencial. Después de que todos los hilos hayan accedido a la zona de exclusión los vectores y la variable deben quedar inicializados para un uso posterior. El número de elementos de los vectores es igual a N.

Pregunta 5: *Modifique el algoritmo descrito al principio del enunciado, código y tamaño de las estructuras de datos, para que los vectores y variables queden inicializados para un uso posterior. Nota: el desbordamiento del valor que almacena la variable cola no es un problema, ya que el número de procesador e hilos es menor que el máximo número natural que puede almacenar la variable cola.*

Pregunta 6: *Modifique el algoritmo descrito al principio del enunciado, código y tamaño de las estructuras de datos, para que no se produzca compartición falsa. Los valores de hid siguen siendo los mismos, pero no los valores para acceder a los elementos del vector indice. Calcule el número de bloques invalidados y el número de fallos (excluyendo peticiones de exclusividad) con esta modificación. Para el resto de hipótesis utilice las de la pregunta 4.*

El lenguaje máquina de los procesador dispone de las primitivas Load Linked (LL) y Store conditional (SC).

instrucción	descripción
LL Rd, dir	la palabra leída en la posición de memoria dir se almacena en el registro Rd y se activa el bit de enlace (bit=1).
SC Rfd, dir	si el bit de enlace está activo (bit=1), almacena el contenido del registro Rfd en la posición de memoria dir y devuelve un 1 en el registro Rfd. En caso contrario no se ejecuta el store y devuelve un 0 en el registro Rfd.

Pregunta 7: *Especifique el código de una implementación de la operación atómica fetch&add con el menor número de instrucciones. La dirección de la variable está almacenada en el registro R1 y valor está almacenado en el registro R2. La función devuelve el resultado en el registro R4. Proponga una implementación para el caso de que se cumpla consistencia secuencial. Posteriormente suponga que el multiprocesador no mantiene consistencia secuencial. Utilice la instrucción INbarrera para garantizar consistencia secuencial. El número de inserciones de la instrucciones INbarrera debe ser el mínimo posible.*

Ejercicio 5.4

Suponga un multiprocesador con P procesadores, donde la organización de las caches privadas es mapeo directo y utilizan escritura retardada. El tamaño de bloque es 32 bytes y el número de contenedores es 16. Los estados de un bloque pueden ser I (inválido) y V (válido). La red de interconexión utilizada en el multiprocesador es un bus y el protocolo de coherencia es del tipo invalidación.

Las caches privadas de los procesadores son bloqueantes. En un fallo de cache se suspende la interpretación de instrucciones y se reanuda al finalizar la transacción.

Una cache puede suministrar directamente el dato dentro de una transacción de bus iniciada por otro procesador. Además, en este caso se actualiza memoria. Cuando una cache tiene un bloque en estado V activa la señal denominada MOD. En la red de interconexión se dispone de un cable que es la función OR de las señales MOD de cada una de las caches. Las peticiones de procesador, las transacciones de bus y el diagrama de transiciones entre estados son las siguientes.

Procesador	Controlador de cache (CC)		Memoria	Diagrama de transiciones entre estados utilizado en los CC
Peticiones	**Transacciones**	**Acciones**	**Acciones**	
LPr : lectura	Pt : petición de bloque	CcRe: reemplazo de un bloque	Dev: almacenar en memoria	Agente procesador LPr / Pt EPr / PtIm I V LPr / - - EPr / - - CcRe / PtX & Dev Agente observador I Pt / CaC & Dev PtIm / CaC & Dev V
EPr : escritura	PtIm: petición de bloque con intención de modificarlo	MOD: señal que indica bloque en estado V en una cache		
	PtX: petición de expulsión	CaC: suministro del bloque		

Pregunta 1: *Indique cuántas copias de un bloque puede haber en las cache del multiprocesador. Justifique la respuesta.*

Dos diseñadores discuten sobre el soporte de la arquitectura para implementar operaciones de acceso exclusivo. Para ello estudian la implementación de la instrucción test&set y del par de instrucciones load linked y store conditional.

Test&set	load linked	store conditional		
TS R_d, X(R_f)	LL R_d, X(R_f)	El que ejecuta: SC R_{fd}, X(R_f)	Otros: dirección de la transacción (dir)	Comentarios
Rd = M [X + R_f^v]	R_d^v = M [X + Rf^v]	if (RLD^v = X + Rf^v and RLR^v = 1) then	if (RLD^v = $dir_{transacción}$ and RLR^v = 1) then	El superíndice v indica valor
M [X + R_f^v] = 1	RLD^v = X + Rf^v	M [X + R_f^v] = R_{fd}^v	RLR^v = 0	RLD: registro que almacena la dirección a la que accede un load linked
	RLR^v = 1	R_{fd}^v = 1		RLR: bit de enlace o reserva
		else R_{fd}^v = 0; la instrucción se convierte en una NOP		Los registros RLD y RLR los gestiona el CC
		RLR^v = 0		

El bit de enlace (RLR) se desactiva si el bloque al que hace referencia el contenido del RLD se invalida por alguna circunstancia. El CC garantiza que cualquiera de estas instrucciones se ejecuta de forma atómica. Si se produce un fallo de cache en una instrucción TS la transacción es del tipo PtIm.

Para comprobar la utilidad de las instrucciones se analiza la siguiente secuencia de accesos a memoria, observada al competir varios procesadores para acceder de forma exclusiva a la variable A.

secuencia de accesos (A)		secuencia de accesos (B)
1. P1 TS llave (1)	El valor inicial de las variables llave y A en memoria es 0 y 24 respectivamente. Inicialmente los bloque que contienen las variables no están almacenados en cache. Los bloques al mapearse en cache ocupan contenedores distintos	1. P1 LL llave
2. P1 store A (3)		2. P1 SC llave(1)
3. P3 TS llave (1)		3. P1 store A (3)
4. P2 TS llave (1)		4. P1 store llave (0)
5. P3 TS llave (1)		5. P3 LL llave
6. P1 store llave (0)		6. P2 LL llave
7. P2 TS llave (1)		7. P3 SC llave (1)
		8. P3 LL llave
		9. P2 SC llave (1)

Pregunta 2: *Para la secuencia A), muestre, mediante una tabla, el estado de los bloques que contienen las variables llave y A en las caches de cada procesador. Así mismo, muestre las transacciones de bus y quién suministra el bloque.*

Pregunta 3: *Para la secuencia B), muestre, mediante una tabla, el estado de los bloques que contienen las variables llave y A en las caches de cada procesador y el valor del bit de enlace o reserva (RLR). Así mismo, muestre las transacciones de bus y quién suministra el bloque. Si una instrucción no se ejecuta especifique el acrónimo NOP en el campo estado.*

Pregunta 4: *Justifique si la implementación de las instrucciones descritas previamente son de utilidad para obtener acceso exclusivo en cualquier circunstancia. Para ello analice las secuencias previas de acceso a memoria.*

En una implementación donde varios CC pueden competir para obtener el bus, no se puede considerar una acción atómica la detección de necesitar efectuar una transacción de bus y la realización de la misma. Por tanto, mientras un CC está pendiente de que se le conceda el bus, debe poder efectuar las acciones de observación inducidas por la transacción en curso en el bus, y en su caso suministrar un bloque de cache.

En el diseño que analizamos se considera atómica una transacción de bus. En consecuencia hay que distinguir la acción de solicitar el bus (PtBus) y la acción de concesión del bus (CnBus). Para ello se utilizan estados transitorios.

Pregunta 5: *Proponga el diagrama de transiciones entre estados cuando se considera que el bus es atómico. Utilice dos diagramas. En uno de ellos muestre las transiciones del agente procesador y en el otro las transiciones del agente observador. Un CC desactiva la solicitud de arbitraje cuando efectúa el suministro y ha soliciado el bus para expulsar el bloque.*

Pregunta 6: *Dada la siguiente secuencia de accesos indique el número de contenedor, si es acierto o fallo, si se produce una expulsión que requiere actualizar memoria y la transacciones de bus. El valor numérico es la dirección accedida.*

accesos	accesos
1. P1 load 2048	5. P2 load 3113
2. P2 store 1061	6. P1 store 3113
3. P3 load 1061	7. P1 load 2056
4. P3 store 1061	Las caches están vacías

Ejercicio 5.5

El protocolo de coherencia que se utiliza es el denominado B del Capítulo 4.

En este multiprocesador se ejecutan los siguientes dos trozos de código que implementan una sincronización mediante evento.

P0	P1	valores iniciales
A = 2. 46	while (flag =0) { };	A = flag = 0
flag = 1	y = A	
. . .	. . .	

Una simplificación del código para efectuar un análisis de consistencia secuencial es la siguiente.

P0	P1	valores iniciales	comentarios
1. store # 2.46, A	3. load R8, flag	A = flag = 0	la sentencia 4 en P1 no se ejecuta hasta que R8 almacena el valor 1
2. store R4, flag	4. load R6, A		

Las variables A y flag están ubicadas en bloques distintos de memoria y su valor inicial es cero. Al almacenarse en cache los bloques ocupan contenedores distintos.

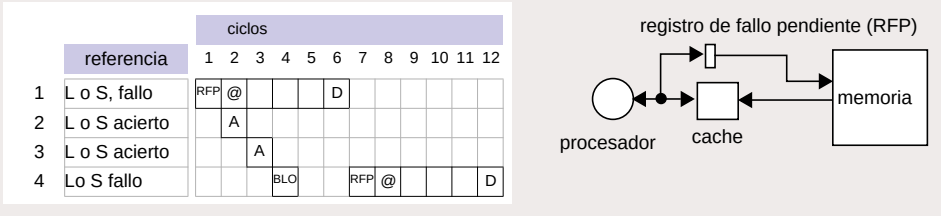
Suponga el siguiente entrelazado de acceso a memoria.

	instrucción	estado inicial
1.	P0 store A	A: estado L en C0 y C1
2.	P0 store flag	flag: estado M en C0 e I en C1
3.	P1 load flag	
4.	P1 load A	

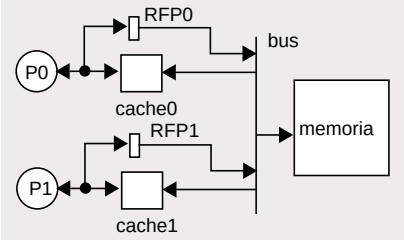
Pregunta 1: *Muestre, mediante un diagrama temporal, las transacciones de bus y los estados de los bloques en las caches.*

En una variante del procesador se permiten aciertos en cache después de un fallo de cache. Esto es, mientras que la cache está esperando el servicio de un fallo, accesos posteriores (más jóvenes) que aciertan en cache se ejecutan. Para ello el fallo se almacena en un registro denominado registro de fallo pendiente (RFP). La cache no soporta un fallo después de un fallo previo. Esto es, el procesador se bloquea cuando se detecta el segundo fallo.

En la parte izquierda de la siguiente figura se muestra un ejemplo donde se observa como dos instrucciones de acceso a memoria, que aciertan al acceder a cache (2, y 3), adelantan a una instrucción de acceso a memoria más vieja (1) que ha fallado al acceder a cache.



El acrónimo A indica acierto en cache. También se muestra una instrucción de acceso a memoria más joven (4) que falla en cache antes de que finalice el servicio del fallo previo. El procesador se bloquea hasta que finaliza el servicio previo (BLO).



Este procesador se utiliza en el diseño de un multiprocesador cuya red de interconexión es un bus y las caches son privadas. En la figura adjunta se muestra un esquema de un multiprocesador con dos procesadores.

En las siguientes preguntan debe considerarse que las caches son no bloqueantes en el primer fallo.

Cuando se especifique una secuencia de accesos a memoria, la ejecución de una instrucción de acceso a memoria, que requiere acceder al bus y no lo obtiene, se especificará mediante dos filas. En la primera fila se especifica la instrucción de acceso a memoria, la cual se almacena en el RFP. Posteriormente, después de otros accesos a memoria se especificará un acceso a memoria desde el RFP para indicar la petición de bloque.

	ciclos											
	1	2	3	4	5	6	7	8	9	10	11	12
L o S	RFP											
. . .												
RFP L o S							arb	@	Obs	ROb		D

En estas condiciones suponga el siguiente entrelazado de accesos a memoria y peticiones desde el RFP.

	instrucción	comentario	estado inicial
1.	P0 store A	inserción en RFP	A: estado L en C0 y C1
2.	P0 store flag		flag: estado M en C0 e I en C1
3.	P1 load flag		
4.	P1 load A		
5.	P0 RFP store A	petición desde RFP	

Pregunta 2: *Muestre, mediante un diagrama temporal, que el multiprocesador no cumple las condiciones de consistencia secuencial, al ejecutarse el entrelazado previo de accesos a memoria y peticiones desde el RFP. Esto es, hay que mostrar que P1 observa los accesos de P0 en un orden distinto al orden de programa.*

El bus dispone de una señal que permite que los procesadores intervengan en una transacción activando esta señal. Por intervención entendemos que se anula la transacción en curso. Esto es, los CC actúan como si no se hubiera efectuado la transacción. El CC al cual se le ha anulado una transacción debe, posteriormente, volver a solicitar el bus para que se repita la transacción.

Pregunta 3: *Indique como se puede aprovechar la señal de intervención descrita para mantener consistencia secuencial en el código anterior (en general no es factible). En una cache que está observando una transacción, especifique exclusivamente las acciones que se efectúa para activar la señal de intervención.*

Suponga que cuando una transacción es anulada pasa a la cola de peticiones pendientes del árbitro. En un diagrama temporal indique el periodo en el que actuaría la acción de anulación. Para ello, utilice filas adicionales después de las filas utilizadas para indicar el estado de los bloques.

En estas condiciones suponga el siguiente entrelazado de accesos a memoria y peticiones desde el RFP.

	instrucción	comentario	estado inicial
1.	P0 store A	inserción en RFP	A: estado L en C0 y C1
2.	P0 store flag		flag: estado M en C0 e I en C1
3.	P1 load flag		
4.	P0 RFP store A	petición desde RFP	
5.	P1 load flag		
6.	P1 load A		

Pregunta 4: *Muestre, mediante un diagrama temporal, que utilizando el mecanismo de intervención se cumplen las condiciones de consistencia secuencial, al ejecutarse el entrelazado previo de accesos a memoria.*

El mecanismo descrito requiere, para que funciones correctamente, que únicamente exista una petición pendiente en el multiprocesador. Esto es, si ya existe una petición pendiente, un fallo en otro procesador requiere que el procesador se bloquee.

Para responder a la siguiente pregunta suponga las siguientes secuencias de instrucciones correspondientes a dos hilos.

	hilo 1	
1	store A	fallo
2	store flag1	acierto en estado M
3	load flag2	fallo
4	load B	

	hilo 2	
5	store B	fallo
6	store flag2	acierto en estado M
7	load flag 1	fallo
8	load A	

Pregunta 5: *Utilice las secuencias de código previo para razonar que se produce abrazo mortal si se permiten en el multiprocesador varios fallos pendientes.*

Otra posibilidad para garantizar consistencia secuencial es mediante la inserción de la instrucción INbarrera, la cual al interpretarse bloquea al procesador si hay fallos pendientes que no han obtenido el bus. Esto es, no se ejecuta ninguna instrucción posterior hasta que la petición en RFP obtiene el bus. En estas condiciones el sistema no utiliza el mecanismo de anulación de transacciones.

Pregunta 6: *Indique en el código simplificado las posiciones en las cuales debe incluirse la instrucción INbarrera. Justifique la respuesta.*

Ejercicio 5.6

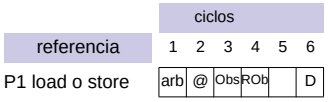
El protocolo de coherencia que se utiliza es el denominado C en este capítulo. Un ingeniero decide añadir al protocolo la funcionalidad denominada transacción de exclusividad (petición PtI), descrita en el Capítulo 4 junto con la descripción suscinta de los protocolos.

Pregunta 1: *Muestre las transiciones entre estados en el agente procesador, el agente observador y en el autómata de memoria cuando se utiliza en el protocolo la petición PtI.*

Como programa de prueba utilizaremos el que se muestra en la parte izquierda de la siguiente tabla. Las variables a y b se ubican en bloques distintos y al almacenarse en cache utilizan contenedores distintos. En la parte derecha de la misma tabla se muestra el entrelazado de accesos a memoria que utilizaremos, los valores de las variables en memoria y en cache, si es el caso, y el estado de los bloques en las caches.

hilo 1	hilo 2	accesos a memoria	valores en memoria	estado en las caches	
a = 3.2	while (b = 0) { }	1. P1 store a (3.2)	a = 8	C1	C2
b = 1	T = a	2. P1 store b (1)	b = 10	a: L	a: L
		3. P2 load b		b: M	b: I
		4. P2 load a		b = 0	

En los diagramas temporales que se soliciten una transacción de bus se representa de la siguiente forma.



En los diagramas temporales añada filas adicionales al final para representar el valor de las variables en las caches.

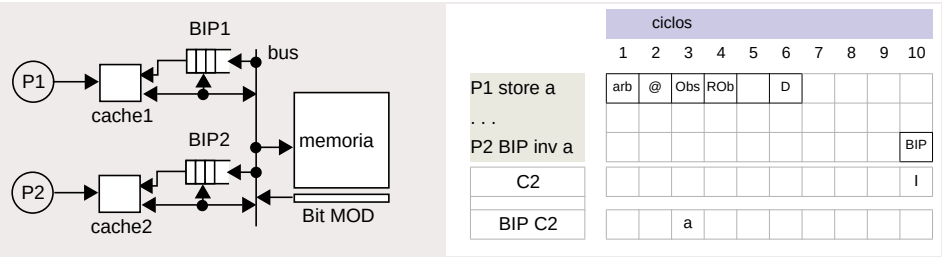
Pregunta 2: *Muestre en un diagrama temporal la secuencia de accesos especificada previamente. Un acceso a memoria que no genera transacción se efectúa en tiempo cero y la fila correspondiente en el diagrama temporal se deja vacía (si se modifica el valor se indica en la fase arb de la siguiente transacción). Indique el número de accesos a memoria que no generan transacción e identifíquelos con el número de orden.*

Una petición PtI requiere que, durante la transacción de bus, las caches que tienen copia del bloque deban invalidarlo. En ocasiones, la actividad en algunos procesadores puede requerir extender la duración de la transacción, para que se realice explícitamente la invalidación del bloque. Esta extensión de la duración de una transacción retrasa la concesión del bus a otros procesadores que quieran utilizarlo y en consecuencia se reduce el rendimiento.

Para no tener que extender la duración de una transacción, una posible solución es añadir en los CC un buffer que almacene invalidaciones pendientes (BIP) en la cache asociada. En estas condiciones, la respuesta a la petición de invalidación, implícita en la transacción de bus, es la inclusión de la petición de invalidación en el BIP (respuesta temprana). Además, para respetar las dependencias de datos que determina el orden de bus, esta petición será procesada por el CC antes de que éste emita cualquier petición relacionada con el bloque al que hace referencia.

Los accesos a memoria, que no referencian bloques con peticiones de invalidación almacenadas en el BIP, pueden generar una transacción antes de procesar las peticiones de invalidación almacenadas en el BIP y recibir la respuesta.

En la parte izquierda de la siguiente figura se muestra un esquema de un multiprocesador con dos procesadores que utilizan un BIP.



Cuando se especifique una secuencia de accesos a memoria, el almacenamiento en BIP se especificará en una fila utilizada para este propósito, ubicada después de las filas dedicadas a la representación del estado de las cache (parte derecha de la figura previa). El instante de inserción en el diagrama temporal se corresponde con la columna que determina la fase Obs de la

transacción. Posteriormente, usualmente después de otros accesos a memoria, se especificará el procesado de la invalidación almacenada en el BIP mediante “Px BIP inv variable”. El procesado de una entrada en el BIP es un ciclo.

En la secuencia anterior de accesos a memoria, las peticiones de invalidación se almacenan en el BIP de los procesadores. Añada al final de la secuencia la siguiente transacción pendiente de finalizar en P2, la cual se realiza desde el BIP, “P2 BIP inv a”, que indica una invalidación pendiente que referencia el bloque que almacena la variable a. Recordemos que cuando una invalidación se almacena en el BIP no se invalida el bloque en la cache. La invalidación se produce cuando se especifica “BIP inv”.

En una transacción Pt o PtIm todos los CC comprueban si tienen el bloque en el estado M. El procesado de esta transacción adelanta al procesado de cualquier entrada en el BIP, que son peticiones PtI.

Pregunta 3: *Al añadir un BIP a cada procesador muestre, mediante un diagrama temporal, que la secuencia de accesos produce un resultado incorrecto. Indique el número de accesos a memoria que no generan transacción e identifíquelos con el número de orden. Indique el valor que lee P2 al acceder a la variable a.*

Pregunta 4: *En una frase, exponga una razón por la cual el resultado es incorrecto.*

Un mecanismo para que el resultado sea correcto es utilizar una instrucción de lenguaje máquina denominada INbarrera. Cuando esta instrucción se ejecuta, todas las entradas ocupadas en el BIP son procesadas antes de que el procesador ejecute una instrucción load más joven. En las condiciones que se acaban de describir, se inserta una instrucción INbarrera entre las dos instrucciones del hilo 2 (mostrado en la parte izquierda de la siguiente tabla). En la parte derecha de la misma tabla se muestra el entrelazado de accesos a memoria que utilizaremos, los valores en memoria de las variables, en cache, si es el caso y el estado de los bloques en las caches.

hilo 1	hilo 2	accesos a memoria	valores en memoria	estado en las caches	
a = 3.2	while (b = 0) { }	1. P1 store a (3.2)	a = 8	C1	C2
b = 1	INbarrera	2. P1 store b (1)	b = 10	a: L	a: L
	T = a	3. P2 load b		b: M	b: I
		4. P2 INbarrera		b= 0	
		1.a. P2 BIP inv a			
		5. P2 load a			

Suponga que el tiempo de ejecución de la instrucción INbarrera es cero. El procesado de las entradas del BIP, que determina la instrucción INbarrera, se especifica en la siguiente fila de la secuencia que se ha mostrado ("1.a P2 BIP inv a").

Pregunta 5: *Muestre, mediante un diagrama temporal, que la secuencia de accesos mostrada previamente produce resultados correctos. Indique el número de accesos a memoria que no generan transacción e identifíquelos con el número de orden. Indique el valor que lee P2 al acceder a la variable a.*

Ejercicio 5.7

Un CC dispone de circuitería en el BEX para suministrar valores al procesador y actualizar un bloque.

Pregunta 1: *Diseñe el diagrama de transiciones entre estados de un bloque en el BEX.*

Ejercicio 5.8

En un multiprocesador las caches privadas son de mapeo directo, utilizan escritura retardada y son bloqueantes. En un fallo de cache o en una solicitud de exclusividad se suspende la interpretación de instrucciones y se reanuda al finalizar la transacción. El multiprocesador dispone de 8 módulos de memoria. En estos módulos de memoria, los bloques están entrelazados utilizando los bits menos significativos de la dirección. La red de interconexión utilizada en el multiprocesador es un bus segmentado y se utiliza el protocolo de invalidación MLI, denominado B.

El arbitraje del bus es cada 3 ciclos y en el siguiente diagrama se muestran las distintas fases de una transacción de bus y el solapamiento de las fases. El bus lógico son dos buses físicos. El bus por el que se transmiten peticiones y el bus por el que se transmiten respuestas de observación y el bloque.

El número de transacciones concurrentes es 3 y cada una de ellas es a un módulo de memoria distinto. El árbitro no concede el bus a una petición que referencia a un módulo de memoria que está siendo accedido por una transacción en curso. La respuesta de observación (ej. señal MOD) se efectúa a una latencia fija desde el inicio de una transacción. El orden de las peticiones, que observan los CC, es el que determina el arbitraje.

		ciclos																								
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
peticiones	Módulo M1	Ar	d0																							
	Módulo M2 ≠ M1				Ar	d1																				
	Módulo M3 ≠ M2 ≠ M1							Ar	d2																	
	Cualquier módulo										Ar	d3														
														Ar	d4											
respuestas						d0		Ro			D	D														
									d1		Ro			D	D											
												d2		Ro			D	D								
															d3		Ro			D	D					
																		d4		Ro			D	D		

Ar: arbitraje
d0 en peticiones: emisión de la dirección
d0 en respuestas: indicación de transferencia (etiqueta de la petición)

Ro: respuesta de observación
D: transmisión de datos

Tiempo de ciclo: 13.3 ns

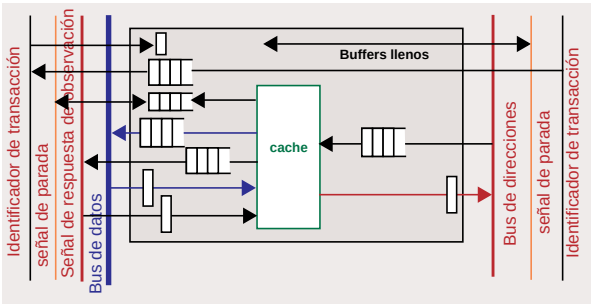
La fase Obs no se muestra en la figura. Se inicia en el ciclo siguiente al de la transmisión de la dirección

El acceso a los campos de la cache, por parte del agente observador, se efectúa en serie. El acceso al campo etiquetas requiere de 3 ciclos y el acceso al campo de datos requiere de 4 ciclos. La latencia de acceso a memoria son 7 ciclos. El agente observador utiliza un duplicado de etiquetas para realizar la acción de observación.

Pregunta 1: *Razone la causa de que el acceso al campo etiquetas no tenga una duración mayor de 3 ciclos. Por ejemplo, 4 ciclos.*

En este diseño el agente procesador tiene prioridad respecto del agente observador para acceder a la cache. En estas condiciones, se puede disponer de una señal de parada para extender la duración de la transacción. Ahora bien, como el bus es síncrono se extiende la duración de todas las transacciones en curso.

Pregunta 2: *¿Puede ser necesario la señal de parada por alguna otra circunstancia?. Analice una ráfaga de transacciones.*

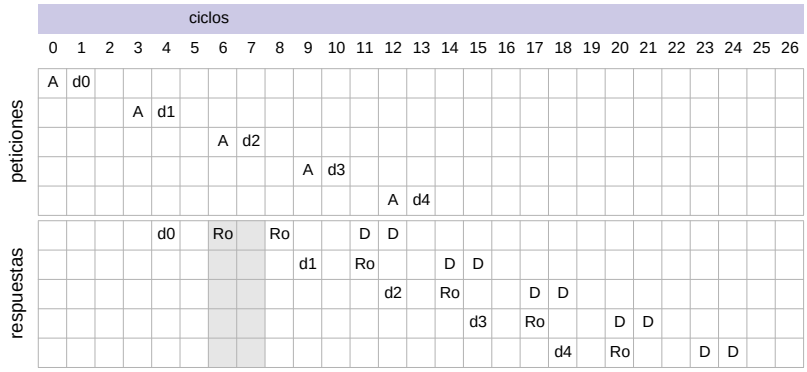


Parar todas las transacciones al unísono representa una perdida de rendimiento. Por ello, se utilizan dos buses. Uno de ellos se utiliza para transmitir la petición y el otro para transmitir la respuesta, la cual incluye la fase de respuesta de observación.

En estas condiciones, las peticiones tienen una etiqueta que se asignar al conceder el bus para emitir la petición. Las respuestas utilizan esta etiqueta para identificarse.

Las respuestas utilizan el bus en el mismo orden que las peticiones y no hay arbitraje. El módulo de memoria accedido indica el inicio de la respuesta 3 ciclos después de haberse emitido la petición, si el bus de repuestas no está ocupado. El envio de datos es 5 ciclos después de esta indicación. La respuesta de observación (e.j. señal MOD) se transmite 2 ciclos después de la indicación de inicio de la respuesta. En este ciclo se puede indicar que se pare el bus de respuestas (parada). Todas las respuestas en curso se paran al unísono.

Cada vez que se activa la señal de parada, el bus de respuestas se para durante 2 ciclos (actual y siguiente). La señal de parada se puede activar las veces que sea necesaria, cada 2 ciclos. En la figura se muestra la activación, una vez, de la señal de parada del bus de respuestas (ciclo 6), debido a que un CC no puede suministrar el dato y activa la señal de parada en el ciclo de respuesta de observación.



Las peticiones, emitidas por el bus de direcciones, se almacenan en cada CC en una cola FIFO junto con el identificador de la transacción. El CC analiza las peticiones en orden. Por tanto, el bus determina el orden en el cual los CC observan las peticiones. Si la cola de algún CC está llena se para el bus de peticiones. Mediante esta cola cada CC conoce la respuesta que se está transmitiendo por el bus.

La parada de cada uno de los buses (direcciones, respuestas) es independiente.

Dada una ráfaga de peticiones, se puede incrementar el rango de ciclos de que dispone un agente observador para acceder al campo de datos de la cache. Para ello, se reduce la latencia de inicio en el bus de direcciones. Esto

es, en lugar de emitir una petición cada 3 ciclos se puede emitir una petición cada 2 ciclos. La latencia de inicio en el bus de respuestas sigue siendo 3 ciclos.

Pregunta 3: *¿Cuál debe ser la latencia de acceso al duplicado de etiquetas para que un CC pueda suministrar el bloque en dos transacciones consecutivas de una ráfaga?.*

Pregunta 4: *Dada una ráfaga de 4 peticiones, muestre en un diagrama temporal, para cada transacción en curso, el rango de ciclos que tiene un agente observador para acceder al campo de datos de la cache. Suponga que el acceso al campo etiquetas es 1 ciclo, después de que se transmita la dirección por el bus y que el acceso al campo de datos requiere 4 ciclos. Suponga que, en peticiones consecutivas de la ráfaga, el agente observador que debe suministrar el bloque es distinto. Indique el número de ciclos de holgura que tienen la 2ª, 3ª y 4ª petición.*

Pregunta 5: *Dada una ráfaga de 4 peticiones, suponga que el primer acceso para el bus 2 veces. Muestre en un diagrama temporal, para cada transacción en curso, el rango de ciclos que tiene un agente observador para acceder al campo de datos de la cache. Suponga que el acceso al campo etiquetas es 1 ciclo, después de que se transmita la dirección por el bus y que el acceso al campo de datos requiere 4 ciclos. Suponga que, en peticiones consecutivas de la ráfaga, el agente observador que debe suministrar el bloque es distinto. Indique el número de ciclos de holgura que tienen la 2ª, 3ª y 4ª petición.*

Un procesador inicia las instrucciones en orden de programa. Si en un mismo ciclo hay más de una petición de acceso, obtiene acceso al bus, si es factible, la petición que ha sido especificada previamente en la secuencia de acceso.

En el multiprocesador se ejecutan los siguientes dos hilos. En la parte derecha se muestra un entrelazado de las peticiones que deben considerarse. El orden de la secuencia de accesos no indica el orden de acceso al bus. El acceso “P2 load aviso” representa el bucle. Entonces, en la fila correspondiente indique todos los accesos que son acierto (A), fallo (F) y la transición de bus, si es el caso.

H1	H2	Comentarios	Accesos
A = 34.23	while (aviso <> 1) { }	Las variables A y aviso están ubicadas en los módulos de memoria M1 y M2 respectivamente.	P1 store A P1 store aviso P2 load aviso P2 load A
aviso = 1	T = A	El valor de las variables A y aviso inicialmente es 1.34 y 0 respectivamente. Las variables A y aviso están almacenada en la cache C2 en el estado L.	

Pregunta 6: *Muestre un diagram temporal de la secuencia de accesos previa.*

En el multiprocesador se ejecutan los siguientes dos hilos. En la parte derecha se muestra un entrelazado de las peticiones que deben considerarse.

H1	H2	Comentarios	Accesos
aviso1 = 1	aviso2 = 1	Las variables aviso1 y aviso2 están ubicadas en los módulos de memoria M1 y M2 respectivamente.	P1 store aviso1 P2 store aviso2 P1 load aviso2 P2 load aviso1
H = aviso2	T = aviso1	El valor de las variables aviso1 y aviso2 inicialmente es 0. Las variables aviso1 y aviso2 están almacenada en la cache C2 en el estado L.	

Pregunta 7: *Muestre un diagram temporal de la secuencia de accesos previa.*