Animaciones interactivas para la enseñanza y aprendizaje de los protocolos de coherencia de cachés*

Alberto Alcón Laguéns, Sergio Barrachina Mir, Enrique S. Quintana Ortí
Dpto. de Ingeniería y Ciencias de los Computadores
Universidad Jaume I
albertoalcon@gmail.com, {barrachi,quintana}@icc.uji.es

Resumen

Entre los objetivos formativos de los cursos avanzados de arquitectura de computadores suele estar el de que los estudiantes sean capaces de describir y analizar el funcionamiento de los protocolos de coherencia de cachés. Aunque dichos protocolos son relativamente sencillos, es necesario analizar muchas situaciones diferentes para entender cómo abordan todos los detalles del problema que quieren resolver. Lo que hace que sean complejos de explicar y de comprender. Una herramienta que ilustrara gráficamente el funcionamiento de dichos protocolos facilitaría enormemente su enseñanza/aprendizaje.

Con objeto de mejorar la docencia de dicha materia, hemos desarrollado tres animaciones interactivas que muestran cómo funcionan tres de los protocolos de coherencia de caché más frecuentemente utilizados. Para cada protocolo, una serie de operaciones de lectura/escritura ilustran todas las posibles situaciones que pueden darse. Las animaciones permiten avanzar y retroceder para poder entender/estudiar mejor las acciones que tienen lugar en cada paso.

Summary

Among the educational objectives in advanced courses of computers architecture there is usually one that states that students should be able to describe and analyze how the cache coherence protocols work. Although these protocols are relatively simple, it is necessary to analyze many different situations to understand how they address all the details of the problem they solve. This makes them complex to be explained and to be understood. A tool that illustrates graphically the operation of these protocols should greatly facilitate the teaching/learning of these protocols.

With the aim of improving the teaching on this subject, we have developed three interactive animations that show how some of the most frequently used cache coherence protocols work. For each protocol, a sequence of read and write operations illustrates all possible situations that can take place in each protocol. The tool is interactive in that the student can go forward and backward to understand/study the different actions that occur at each step.

Palabras clave

Arquitectura de computadores, multiprocesadores, coherencia de cachés, protocolos, animación interactiva

1. Introducción

Todos los multiprocesadores actuales contienen uno o más niveles de memoria caché para mitigar la diferencia entre las velocidades del procesador y de la memoria principal. En estos sistemas, los datos pueden almacenarse/replicarse en múltiples sitios. Esto introduce un problema serio ya que, si no se tiene cuidado, diferentes procesadores podrían acabar almacenando valores distintos para los mismos datos. Para evitar que esto ocurra, se necesita un esquema (protocolo o algoritmo) que garantice la coherencia del sistema de memoria.

En los cursos avanzados de arquitectura de computadores se enseña cómo se puede abordar el problema de la coherencia de cachés utilizando protocolos basados en espionaje y en directorio. Aunque los algoritmos son relativamente sencillos, hay que analizar muchas situaciones diferentes para entender cómo lidian estos protocolos con todos los detalles del problema. Creemos que una herramienta que ilustre gráficamente el modo en el que los distintos protocolos tratan las diferentes situaciones puede

*Este trabajo ha sido parcialmente financiado por el proyecto «Activitats formatives per a assignatures de la matèria Arquitectura de Computadores» de la Unitat de Suport Educatiu de la Universitat Jaume I (10G136-16).

facilitar el aprendizaje de dichos protocolos.

Para explicar los protocolos de coherencia de cachés en nuestras clases de arquitectura avanzada de computadores, buscamos herramientas de simulación o animaciones que pudieran ilustrar como funcionan dichos protocolos. Queríamos que nuestros estudiantes pudieran experimentar cómo cambian los estados en cada bloque de caché cuando los diferentes procesadores acceden a diferentes posiciones de memoria, cuáles son las transiciones entre estados, y cuáles son los mensajes que se generan en cada caso. Aunque dimos con algunos simuladores sofisticados, éstos eran demasiado complejos, principalmente porque no estaban diseñados para servir como herramientas de ayuda a la docencia. También encontramos varias animaciones sencillas que capturaban algunas situaciones de coherencia de cachés pero que, desafortunadamente, eran demasiado simplistas y, por otro lado, para nada exhaustivas en cuanto a las posibles situaciones que cubrían.

El simulador desarrollado por Jeremy Jones para el protocolo MECI de coherencia de cachés [4] ha resultado ser la herramienta que más se parecía a lo que estábamos buscando. Dicho simulador representa de forma gráfica un multiprocesador con un banco de memoria y tres procesadores con sus respectivas cachés. El usuario puede indicar de forma interactiva las operaciones de lectura y escritura que debe realizar cada procesador. Cuando se realiza una operación, se muestra de forma animada tanto el trasiego de los datos como la variación de los estados de las líneas de caché implicadas.

No obstante, dicho simulador presenta desde nuestro punto de vista, una serie de inconvenientes para el aprendizaje de los protocolos de coherencia. El principal inconveniente para utilizarlo como herramienta de apoyo a la docencia es que es necesario conocer de antemano cómo funciona el protocolo MECI de coherencia de cachés. Esto es así ya que para poder observar cómo funciona el protocolo en cada uno de los casos posibles, es necesario generar una secuencia de lecturas/escrituras que provoque el caso que se quiere observar. Otro de los inconvenientes que presenta actualmente es que no es multiplataforma. El simulador está desarrollado sobre Vivio [3], que por el momento solo puede ejecutarse en Windows (aunque está prevista una versión de Vivio basada en Qt, que presumiblemente será más portable). Por último, dicho simulador tan solo cubre uno de los tres protocolos que queremos mostrar.

Debido a todo lo anterior, decidimos desarrollar un conjunto de animaciones interactivas que pudieran utilizarse en clase para mostrar a los estudiantes el funcionamiento de los protocolos de coherencia de cachés, y que les pudieran servir para ganar un mayor grado de comprensión de los mismos. También decidimos desarrollar las animaciones en Flash para que éstas tuvieran la mayor portabilidad posible.

Las animaciones interactivas que hemos desarrollado cubren los protocolos MCI y MECI basados en espionaje, así como el protocolo MECI basado en directorio. Los casos propuestos en estas animaciones son exhaustivos, cubriendo todos las posibles situaciones que se pueden dar en cada protocolo. Con cada ejemplo se muestra una descripción completa de qué está ocurriendo, y el estudiante puede avanzar y retroceder para examinar detenidamente cómo se resuelve cada una de los casos. Además, hemos intentado que las animaciones fueran intuitivas y para que no requirieran de preparación previa para comenzar a utilizarlas.

El resto del artículo está organizado como sigue. En el Apartado 2, ofrecemos un breve visón general del problema de la coherencia de cachés. El Apartado 3 describe las animaciones desarrolladas. Finalmente, en el Apartado 4, se resumen las conclusiones y el trabajo futuro que esperamos llevar a cabo.

2. El problema de la coherencia de cachés

Los multiprocesadores, especialmente los construidos a partir de microprocesadores de bajo costo, ofrecen una solución efectiva en coste a la siempre creciente demanda de mayor potencia de computación [1]. Sin embargo, diseñar y programar correcta y eficientemente multiprocesadores, plantea una serie de problemas complejos, siendo uno de ellos el de mantener la coherencia del sistema de memoria.

El problema de la coherencia de cachés surge cuando diferentes procesadores guardan en sus respectivas cachés el valor de la misma posición de memoria y posteriormente lo modifican [2]. Una solución inteligente, basada en el uso de un bus como red de interconexión, aborda dicho problema en multiprocesadores a pequeña escala con memoria compartida. La idea básica es la de garantizar que, antes de que se escriba en una dirección de memoria, todas

las otras copias de dicha dirección, que podrían estar en otras cachés, sean invalidadas. De esta forma, el sistema permite que existan múltiples copias de lectura de una posición de memoria, pero solo permite una copia de escritura.

El componente clave para los protocolos habituales de coherencia basados en espionaje es el bus que interconecta los procesadores y los módulos de memoria distribuida. Cuando un procesador quiere escribir en un bloque de caché, que podría estar compartido, un protocolo basado en espionaje coloca la petición en el bus. De esta forma, todas las cachés pueden ver la petición y si tienen una copia de dicho bloque de caché, simplemente la invalidan.

En resumen, las operaciones de espionaje se implementan colocando la petición en el bus y haciendo que todas las cachés lean la dirección implicada; si dicha dirección coincide con una de las guardadas en la caché del procesador, o bien realizan una invalidación, o bien proporcionan los datos de su caché. Puesto que todas las peticiones tienen que colocarse en el bus, que solo puede servir una petición a la vez, el bus serializa las escrituras posiblemente concurrentes de dos o más procesadores. Esto impone un orden en todas las escrituras (incluyendo aquellas que van a la misma dirección), lo que es crítico para mantener la coherencia.

La reducción en la complejidad de la programación que se consigue al mantener la coherencia de cachés, junto con su relativamente barata implementación, llevó a su adopción en todos los multiprocesadores de pequeña escala basados en bus. En los últimos años, se han popularizado los multiprocesadores diseñados con un pequeño número de núcleos (dos a ocho) en el mismo chip que implementan protocolos de coherencia de cachés, reduciendo aún más el coste de los multiprocesadores a pequeña escala e incrementando su popularidad.

Desafortunadamente, los esquemas de espionaje utilizados en los multiprocesadores simétricos no escalan bien, puesto que el bus se convierte rápidamente en un cuello de botella cuando crece el número de procesadores conectados a él.

La solución a este problema consistió en desarrollar un mecanismo de coherencia que pudiera extenderse eficientemente a otros diseños arquitecturales o de interconexión. La respuesta fueron los esquemas basados en directorio. Estos esquemas confían en una estructura adicional, llamada directorio, que realiza un seguimiento de qué procesadores han guardado en caché cualquier bloque de memoria principal. Puesto que el directorio mantiene información actualizada de qué cachés tienen copia de un bloque de memoria dado, un protocolo de coherencia puede utilizar dicha información para conseguir una visión consistente del sistema de memoria. Para garantizar la coherencia, el estado de cada bloque de caché es almacenado en la caché y se guarda información adicional para cada bloque en el directorio.

Los protocolos basados en directorio no fueron adoptados inicialmente debido a que, en un pequeño multiprocesador, un bus cubre ampliamente las necesidades de comunicación y, además, los esquemas basados en espionaje son fáciles y baratos de implementar [2]. También hay que tener en cuenta que, aunque la aproximación basada en directorios evita la utilización de difusiones para interrogar a todas las cachés, un directorio único y centralizado tampoco es la solución, ya que el cuello de botella se desplazaría del bus al directorio. Estos protocolos han empezado a utilizarse ampliamente con la llegada de multiprocesadores con un gran número de procesadores que utilizan una red de interconexión distinta a un bus y en los que el directorio se ha distribuido junto con los bancos de memoria.

Los protocolos de coherencia de cachés basados en espionaje y en directorio pueden utilizar tres o cuatro estados diferentes para marcar el estado de cada bloque de caché [5]. En el protocolo MCI de coherencia de caché, cada bloque de caché puede estar en uno de los siguientes tres estados: Modificado, Compartido o Inválido. Un bloque de caché se marca como modificado cuando su contenido ha sido modificado. Cuando un bloque de caché está en dicho estado, esa caché es la única que tiene una copia válida de dicho bloque. Un bloque de caché se marca como compartido cuando la información que contiene es fruto de una operación de lectura. Cuando un bloque está en dicho estado, hay otras copias válidas de los datos que contiene (como mínimo, en memoria principal). Finalmente, un bloque de caché se marca como inválido cuando otro procesador ha realizado una petición de escritura sobre el bloque de memoria al que hace referencia dicho bloque de caché.

Por otro lado, el protocolo MECI de coherencia de cachés incorpora un nuevo estado: Exclusivo. Este estado se utiliza para indicar que la caché tiene

una copia válida y no modificada de los datos y que no hay otras cachés con copia de dichos datos. Gracias a este estado, cuando un procesador realiza una operación de escritura sobre un bloque de caché que está únicamente en su caché (en estado exclusivo), no necesita poner en el bus una petición de escritura. Esto aligera la utilización del bus, especialmente cuando se ejecutan programas donde hay pocos datos que se compartan realmente entre los distintos procesos.

Las animaciones que presentamos en el siguiente apartado tienen por objeto facilitar la comprensión de cómo funcionan los protocolos MCI y MECI de coherencia de cachés basados en espionaje y el protocolo MECI basado en directorio.

3. Animaciones interactivas de protocolos de coherencia de cachés

Para desarrollar las animaciones hemos utilizado Adobe Flash. Esta herramienta proporciona dos formas de desarrollar aplicaciones. La primera de ellas consiste en definir una línea del tiempo y dibujar las imágenes que forman la animación. Afortunadamente, no es necesario dibujar manualmente todas las imágenes, ya que Adobe Flash es capaz de crear automáticamente las imágenes requeridas entre una imagen inicial y otra final. En particular, Adobe Flash hace esto siguiendo la pista a los objetos que cambian su tamaño o posición entre estas dos imágenes y generando las imágenes intermedias. Utilizando este método se pueden obtener fácilmente animaciones sencillas.

La otra forma de definir el comportamiento de una animación en Adobe Flash es utilizando ActionScript, un lenguaje de programación orientado a objetos, que se puede emplear para acceder a los métodos y propiedades de los objetos dibujados en el área de trabajo. Utilizando ActionScript se pueden conseguir efectos más complejos y, lo que es más importante, se pueden crear animaciones con las que el usuario pueda interactuar.

Para desarrollar las animaciones hemos utilizado ambos métodos: la definición de una línea de tiempo y la programación con ActionScript. El primero de ellos principalmente para definir todos los casos de estudio y el segundo para permitir al usuario la navegación por toda la animación.

Las animaciones están disponibles en tres ficheros Flash, uno para cada uno de los siguientes protocolos de coherencia de cachés: MCI basado en espionaje, MECI basado en espionaje y MECI basado en directorio (este último utiliza un directorio implementado como un vector de bits completo). Estas animaciones pueden descargarse desde: «`http://lorca.act.uji.es/projects/ccp/`».

En la Figura 1 puede verse la interfaz gráfica de la animación correspondiente al segundo de los protocolos: el protocolo MECI basado en espionaje. La interfaz gráfica está dividida en las siguientes partes:

1. El título de la animación.
2. Una descripción detallada de qué ocurre en el paso actual.
3. Botones «anterior», «reproducir» y «siguiente». El usuario puede utilizarlos para ir al paso anterior, para reproducir (de nuevo) el paso actual y para ir al siguiente paso, respectivamente.
4. La representación del multiprocesador utilizado como ejemplo (esta misma representación se utiliza para el protocolo MCI basado en espionaje, mientras que para el protocolo basado en directorio se utiliza otra bastante diferente).
5. Botones para cada paso. El usuario puede utilizarlos para ir directamente a un paso determinado. Esta parte también muestra cuál es el paso actual y qué pasos se han completado ya.

El multiprocesador mostrado en las animaciones de los protocolos basados en espionaje consta de tres procesadores (P1, P2 y P3) y de dos bancos de memoria (MEM1 y MEM2), todos ellos interconectados mediante un bus. Cada procesador tiene los siguientes elementos (de arriba a abajo y de izquierda a derecha):

- Un área de texto que representa el núcleo del microprocesador y que se utiliza para indicar la petición de lectura o escritura que se está realizando.
- Una tabla que representa el contenido de la caché del procesador. La caché consta de dos bloques de 4 palabras cada uno.
- Una columna que muestra el estado actual de cada bloque de caché: modificado (M), exclusivo (E), compartido (C) o inválido (I).

Cada memoria consta de los siguientes elementos:

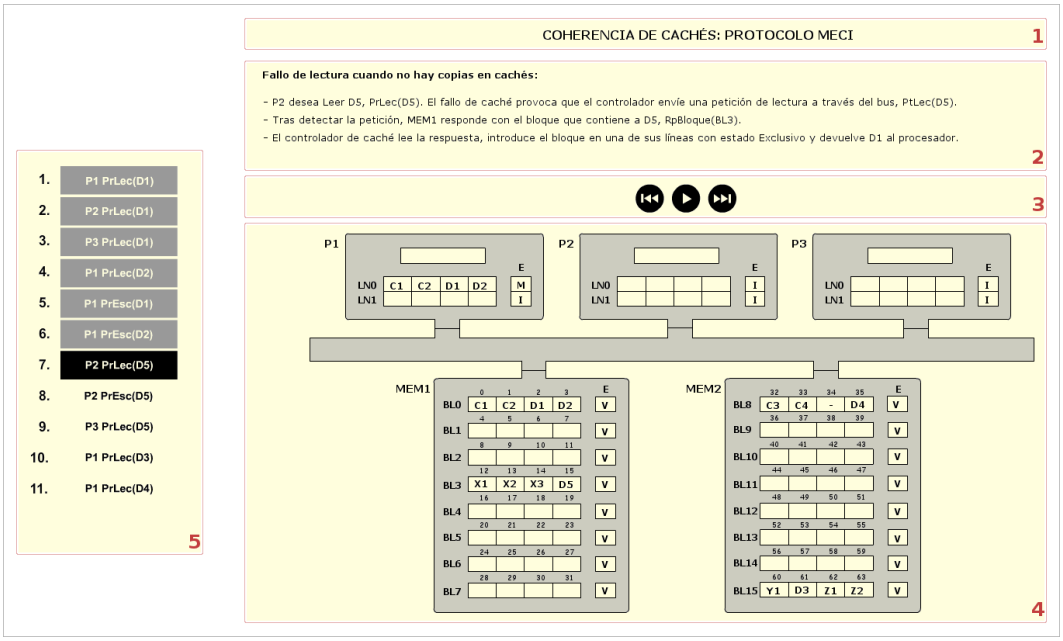

Figura 1: Interfaz gráfica de la animación correspondiente al protocolo de coherencia MECI basado en espionaje.

- Una tabla que representa el contenido de la memoria. La memoria contiene 32 palabras organizadas en 8 bloques de 4 palabras cada uno.
- Una columna que muestra el estado actual de cada bloque de memoria: válido (V) o inválido (I).

La animación del protocolo MECI basado en directorio utiliza un multiprocesador diferente (ver Figura 2). Esta arquitectura consta de cuatro nodos interconectados mediante una red. Cada nodo tiene un procesador, una caché y una memoria similares a los ya descritos. La principal diferencia entre este multiprocesador y los de las otras dos animaciones es que en éste, para cada bloque de memoria, hay un vector de bits donde se almacena la información del directorio.

El funcionamiento general de las animaciones se resume a continuación.

1. Al comenzar cada paso se muestra el estado del multiprocesador en dicho paso y una descripción detallada de qué es lo que va ocurrir. El usuario debe pulsar el botón «reproducir» para que comience la animación de dicho paso. Esto se ha hecho así para permitir al estudiante leer qué es lo que va a ocurrir antes de que efectivamente ocurra.
2. En cuanto se pulsa el botón «reproducir», se muestra de forma animada cómo uno de los procesadores realiza una petición de lectura o de escritura y cómo esta petición provoca una consulta en su caché.
3. A continuación se puede observar cómo el controlador de caché responde al procesador con el dato solicitado. O alternativamente, cómo el controlador de caché coloca en el bus la petición requerida para conseguir dicho dato o para invalidar las otras copias de dicho dato.
4. En el caso de que se transmita la petición por el bus, se muestra cómo los controladores de caché afectados por dicha petición realizan las acciones indicadas por el protocolo de coherencia para dicha situación.
5. Finalmente, y si es el caso, se representa de forma animada cómo los datos viajan por el bus desde donde estén actualizados en ese momento hasta la caché, y de la caché al procesador que inició la operación.

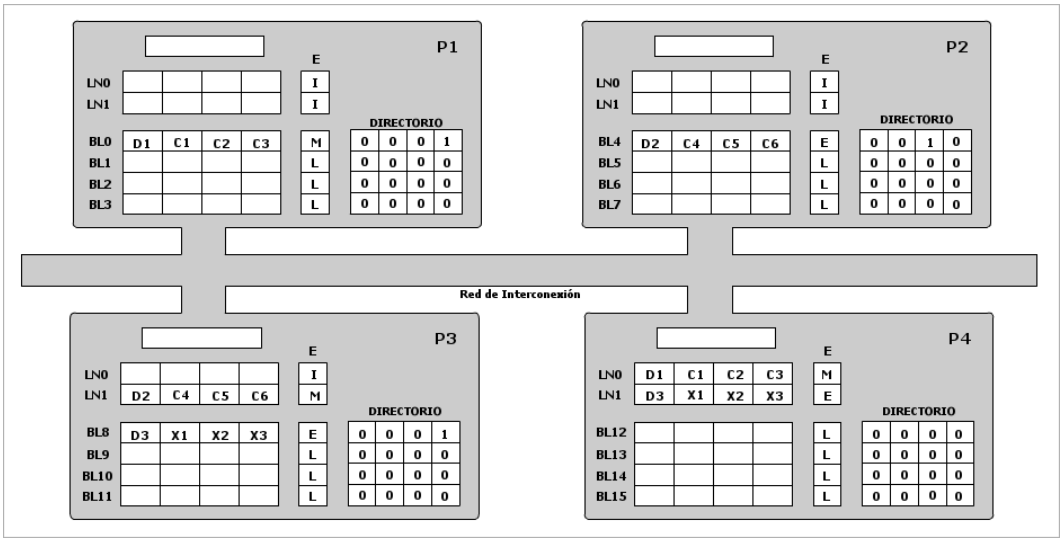

Figura 2: Multiprocesador utilizado en la animación del protocolo de coherencia MECI basado en directorio.

Añadir que mientras transcurre la animación, se resalta también de forma gráfica si los controladores de caché o los bancos de memoria actualizan el estado de alguno de sus bloques. De igual forma, en la animación del protocolo MECI basado en directorio, también se resalta gráficamente, cuando se da el caso, la modificación del directorio asociado a cada bloque de memoria.

El profesor puede explicar en clase en funcionamiento de los distintos protocolos utilizando estas animaciones como material de apoyo, en lugar de utilizar transparencias. (Las animaciones se han simplificado al máximo para que se visualicen correctamente cuando sean proyectadas en clase.) Otra posible forma de utilizarlas es que el profesor de un tiempo a los estudiantes para que estudien individualmente o en grupo el funcionamiento de los distintos protocolos. Para posteriormente hacer que sean los propios estudiantes los encargados de explicar en clase lo que esperan que ocurra en cada paso de la animación. Una vez que han descrito qué esperan que ocurra, se pone en marcha la animación para comprobar si estaban en lo cierto. En el supuesto de que los estudiantes hubieran fallado en su descripción, y puesto que la animación permite repetir cada paso, se puede incidir en el comportamiento que no había sido previsto correctamente. Ésta es la forma en la que estamos utilizando actualmente las animaciones en el contexto del aula. Nuestra experiencia ha sido tremendamente positiva ya que ha conseguido involucrar a los estudiantes en el aprendizaje de los protocolos, a la vez que ha estimulado su participación en clase.

También se pueden proporcionar estas animaciones a los estudiantes. De esta forma, pueden utilizarlas por su cuenta para profundizar en su conocimiento sobre el comportamiento de los protocolos de coherencia de cachés. Utilizar estas animaciones de forma autónoma, preferiblemente después de haberlas visto en clase, creemos que les ha ayudado a alcanzar el objetivo formativo de ser capaces de describir y analizar el funcionamiento de dichos protocolos.

Lamentablemente, y puesto que solo tenemos un grupo de estudiantes y que los protocolos de coherencia representan una parte de la materia, no hemos sabido cómo medir numéricamente el impacto que la utilización de estas animaciones han tenido en el aprendizaje de nuestros estudiantes.

Por último, consideramos que las animaciones presentadas permiten adquirir una comprensión básica del funcionamiento de los protocolos de coherencia de cachés. Que es el objetivo en nuestro curso de arquitectura de computadores. No obstante, en el

caso de que los estudiantes debieran adquirir un conocimiento más profundo de dichos protocolos, sería conveniente complementar la utilización de estas animaciones con prácticas con un simulador como el de Jeremy Jones [4].

4. Conclusiones y trabajo futuro

Hemos desarrollado tres animaciones en Flash que abarcan todas las posibles situaciones que pueden darse en la práctica en los tres protocolos considerados de coherencia de cachés. Hemos comprobado en nuestra docencia que estas animaciones pueden ayudar a entender cómo funcionan estos protocolos de coherencia de cachés y que el profesor puede utilizarlas eficazmente en clase. Es más, consideramos que estas animaciones también pueden ser utilizadas por los estudiantes para analizar por su cuenta cómo funcionan los protocolos de coherencia de cachés.

Desarrollar las animaciones ha sido mucho más complejo de lo que esperábamos, especialmente cuando hemos necesitado introducir cambios sobrevenidos en sus líneas del tiempo. La experiencia nos ha enseñado que es sumamente conveniente planificar cuidadosamente todos los detalles antes de comenzar la codificación propiamente dicha de la animación. En particular, los cambios de última hora fueron muy costosos de implementar.

Finalmente, tenemos previsto seguir utilizando estas animaciones en nuestro curso avanzado de arquitectura de computadores y obtener realimentación de los estudiantes sobre qué mejoras piensan que podríamos realizar para conseguir una comprensión mejor de cómo funcionan los diferentes protocolos de coherencia de cachés.

Referencias

[1] Culler, David E., Pal Singh, Jaswinder, Gupta, Anoop, *Parallel computer architecture: a hardware/software approach*, Morgan Kaufmann, 1998.

[2] Hennessy, John L., Patterson, David A., *Computer Architecture: A Quantitative Approach*, Morgan Kaufmann, 2006.

[3] Jones, Jeremy, *Vivio 5.1 - A Tool For Creating Interactive Reversible E-Learning Animations for the WWW*, `http://www.scss.tcd.ie/~jones/vivio/home.htm`

[4] Jones, Jeremy, *Vivio MESI cache coherency protocol animation*, `http://www.scss.tcd.ie/~jones/vivio/caches/MESI.htm`

[5] Ortega, Julio, Anguita, Mancia, Prieto, Alberto, *Arquitectura de Computadores*, Editorial Thomson, 2005.