Universidad de Costa Rica | Facultad de Ingeniería | Escuela de Ingeniería Eléctrica

Anteproyecto VII

Roberto Sánchez Cárdenas — B77059
Gabriel Jiménez Amador — B73895

San José, 31 de agosto de 2021
IE0424 — Laboratorio de Circuitos Digitales

Índice

1. Introducción

Con el fin de continuar profundizando el conocimiento respecto a las memorias cache, se propone la implementación de una cache asociativa con el uso de políticas de reemplazo.

2. Objetivos

- Analizar interfaces de memoria de un CPU e implementar un cache asociativo.
- Recolectar y analizar datos de rendimiento de un cache

3. Anteproyecto

3.1. ¿Qué es la asociatividad de un cache? ¿En qué afecta?

La asociatividad es una capacidad que se le integra a las memorias cache con el fin de reducir la hiperpaginación. A grande rasgos, la asociatividad lo que hace es dividir la memoria cache en varias memorias de menor tamaño y de igual tamaño, a cada una se le conoce como way. Por ejemplo, un caché de 128 lineas, se puede dividir en 4 caches de 32 líneas. [1]

Cuando se lee un dato de memoria y se trae a la cache, se puede escribir en cualquiera de las ways sin afectar el funcionamiento. Como las memorias se hacen más pequeñas, la probabilidad de mappear un dado es 4 veces más alta. [1]

3.2. Mencione y explique 3 políticas de reemplazo diferentes. ¿Cuál es la importancia de las políticas de reemplazo?

Se sabe que mantener memoria en caché incrementa el desempeño al realizar transacciones de lectura y escritura[1], sin embargo, una caché es más costosa de producir en sistemas reales, que una memoria RAM. Es por esto que sus tamaños son considerablemente menores a los de una memoria principal. La caché busca mantener los bloques de memoria más recientes o más usados para poder accesarlos más rápido o con un costo computacional menor. Sin embargo, como tienen un tamaño reducido, cuando esta se llena hay que elegir cómo realizar el reemplazo de bloques.

Las políticas de reemplazo en una caché buscan encontrar un óptimo reemplazo de bloques en una caché, de forma que se mantienen los bloques de memoria estadísticamente más accesados y se reemplazan los menos accesados [3]. Se enumeran tres políticas de reemplazo, cada uno con algoritmos de reemplazo diferentes cuando ocurre un *hit* o *miss*, a continuación:

- **LRU (Least Recently Used)**: Esta política es simple y comúnmente utilizada. Funciona similar a un FIFO, es decir el primer bloque que se escribe a la caché será el primero en reemplazarse si no se utiliza y será considerado el LRU. Pero, si se referencia al bloque antes de ser reemplazado, se debe refrescar y poner en la posición MRU (Most Recently Used) al inicio de la cola [3]. En resumen, en caso de un:
  - **Hit**:

1 Como se concluyó en el Laboratorio 6.

    - El bloque se coloca en la posición MRU.
  - **Miss**:
    - Se reemplaza el bloque LRU.
    - Pone el bloque entrante en MRU.

- **NRU (Not Recently Used)**: NRU usa un solo bit por bloque de caché llamado nru-bit para determinar el bloque a reemplazar. Con solo un bit de información, NRU permite dos intervalos de predicción de re-referencia: re-referencia casi inmediata y referencia distante. Un valor de nru-bit de 0 implica que un bloque fue recientemente utilizado y se predice que el bloque será re-referenciado en un futuro próximo. Un valor de nru-bit de 1 implica que el bloque fue no se ha utilizado recientemente y se predice que el bloque será re-referenciado en el futuro lejano [3]. En resumen, en caso de un:
  - **Hit**:
    - El bloque se coloca en la caché con un nru-bit de 0.
  - **Miss**:
    - Se busca el primer bloque con 1 en la caché.
    - Si se encuentra un 1, se reemplaza ese bloque y se ingresa el nuevo bloque con un valor de 1.
    - Si no se encontró ningún bloque con 1, se cambian todos los bloques a 1 y se repite el proceso de Miss.

- **SRRIP (Static Re-Reference Interval Prediction)**: Esta política expande la idea de NRU, pero en lugar de 1 bit de re-referencia se disponen de M bits, para poder tener 2^M posibles valores RPPV (Re-reference Prediction Values). Así como NRU, un RRPV de cero implica que se predice que un bloque de caché será re-referenciado en el futuro cercano inmediato mientras que un RRPV de saturación (es decir, $2^M - 1$) implica que se predice que se volverá a hacer referencia a un bloque de caché en el futuro distante [3]. En resumen, para un SSRIP de $M = 2$ bits, en caso de un:
  - **Hit**:
    - Pone el RRPV del bloque en 0.
  - **Miss**:
    - Se busca el primer bloque con 3 en la caché.
    - Si se encuentra un 3, se reemplaza ese bloque y se ingresa el nuevo bloque con un valor de 2.

- Si no se encontró ningún bloque con 1, se incrementa en uno todos los RRPVs de los bloques y se repite el proceso de Miss.

3.3. Documente, cuál es el tamaño del byte offset, el tamaño del índice del bloque y el tamaño del tag para los siguientes tamaños de cache, tamaños de bloque y asociatividad

Tamaño bloque	Tamaño cache	Asociatividad	Bytes x bloque	Offset	Set	Tag
2 palabras	1kB	2-way	8	3	6	23
2 palabras	2kB	2-way	8	3	7	22
4 palabras	2kB	2-way	16	4	6	22
2 palabras	4kB	2-way	8	3	8	21
4 palabras	4kB	2-way	16	4	7	21
2 palabras	1kB	4-way	8	3	5	24
2 palabras	2kB	4-way	8	3	6	23
4 palabras	2kB	4-way	16	4	5	23
2 palabras	4kB	4-way	8	3	7	22
4 palabras	4kB	4-way	16	4	6	22

Tabla 1: Tamaños de las direcciones en memoria caché

3.4. Investigue acerca de cómo funciona la generación de números aleatorios usando registros de desplazamiento (LSFR o Linear feedback shift register).

El LSFR son un método por el cual se pueden obtener números pseudo aleatorios. La implementación de estos se puede realizar por medio de hardware o software. Para el caso del hardware, se tiene un cirucito digital con 2n-1 estados, donde n es el número de registros que posee el módulo. [2]

En cada flanco de reloj, el contenido de los registros se desplaza a la derecha una posición, en este momento se tiene retroalimenzatción del registro más a la derecha por medio de una XNOR o una XOR. Un caso ilegal es tener todo 1s con una compuerta XNOR o solo 0 con una XOR, pues no sería posible salir de este caso. [2]

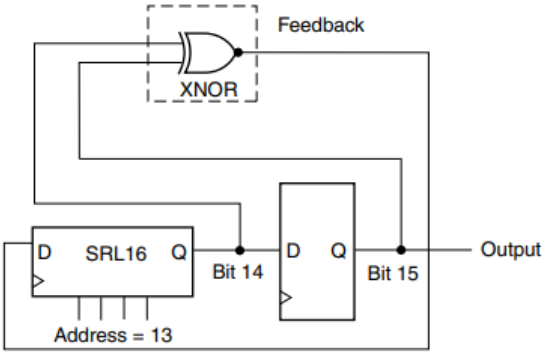

Figura 1: Diseño de circuito LFSR [2]

4. Diseño

4.1. Cache asociativo con 2 ways y política de reemplazo aleatoria

Para la implementación de una cache asociativa en dos direcciones (2 ways) con política de remplazo aleatoria y escritura writeback, se toma como base la memoria cache diseñada e implementada en el laboratorio 6 del curso de Laboratorio de Circuitos Digitales, sin embargo se deben realizar modificaciones pues la asociatividad requiere tomar la memoria cache principal y hacer memorias más pequeñas, de modo que cada memoria de menor tamaño se accesa como una vía.

Además se implementa un política de reemplazo aleatoria, que indica que se reemplazan los datos de una vía aleatoria en lugar de como se realiza con una sola vía, que siempre se reemplaza si no encuentra si el dato no se encontró en única vía. Los números aleatorios de generan por medio de un módulo LSFR.

En alto nivel, el diseño de la máquina de estados que controla el funcionamiento de la cache viene dado por el diagrama de la Figura 2. Este diagrama ilustra la cache diseñada en el Laboratorio 6.

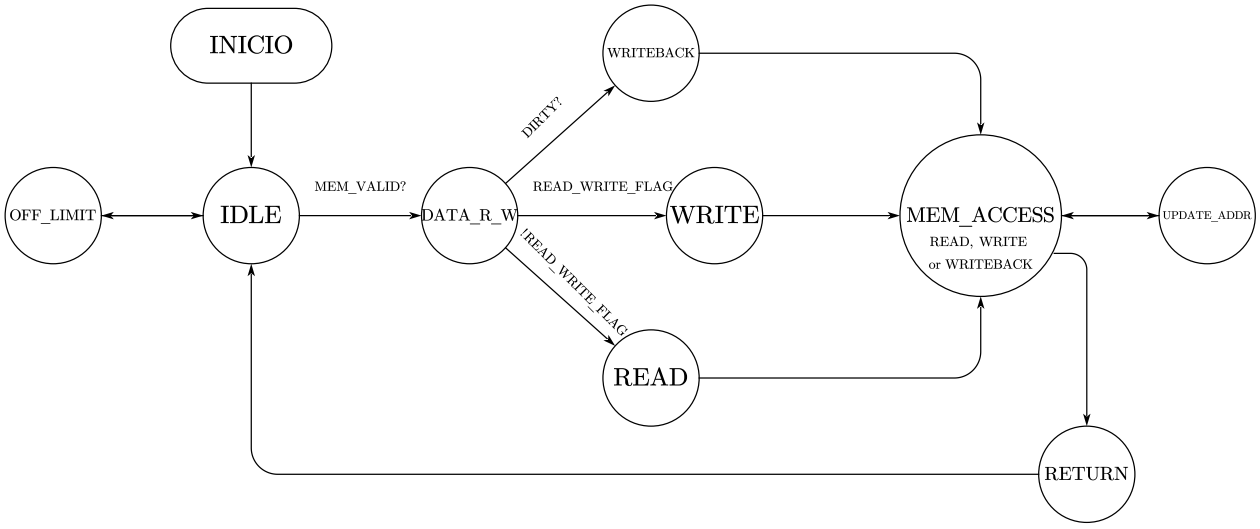

Figura 2: Máquina de estados de cache mapeo directo

Para implementar las vías de la asociatividad se aprovecha que el lenguaje de descripción de hardware permite crear arrays multidimensionales, de modo que se genera una matriz de 3 dimensiones, donde se tienen dimensiones para elegir el tamaño de la caché, el tamaño de las líneas y por último el número de vías. Para una caché de 1 KB y bloques de 2 palabras quedaría (según la Tabla 1):

- 8 bytes por bloque
- $\frac{1 \text{ KB}}{8 \text{ bytes/bloque} \cdot 2 \text{ vías}}$ =64 bloques/vía
- Offset de 3 bits
- Set de 6 bits
- Tag de 23 bits

Para tener un cache asociativa se toma la cache de mapeo directo y se le realizan algunos cambios en la escritura y lectura de datos, de modo que se implementa el reemplazo de datos en vías aleatorias a la hora de realizar *hit/miss*. Esto se visualiza en el diagrama de flujo de la Figura 3.

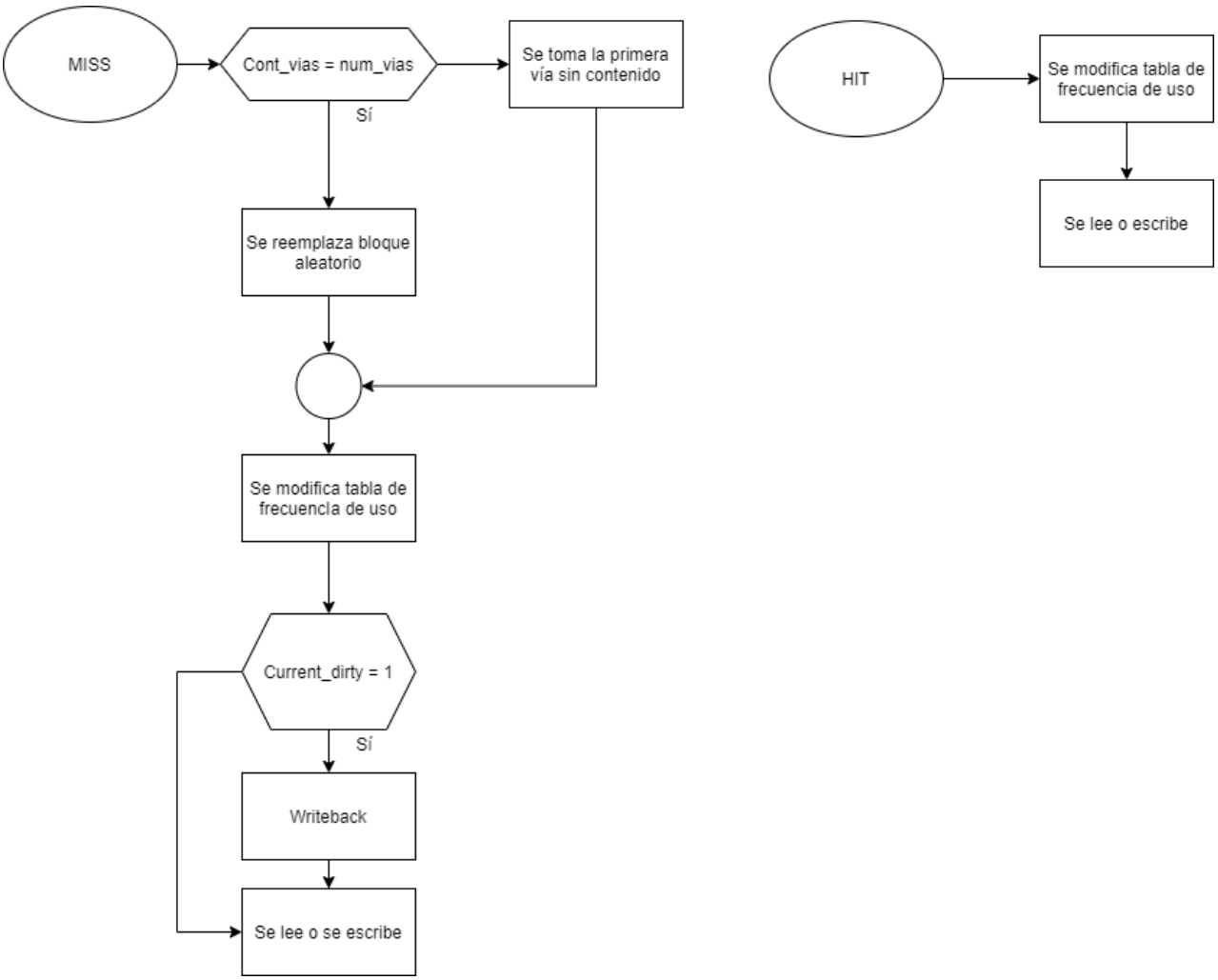

Figura 3: Cambios en lectura y escritura dependiendo de si es hit o miss para el caché asociativo con 2 ways y política de reemplazo aleatoria.

4.2. Modificación de los tamaños del cache

La cache diseñada es parametrizable, por lo que debe funcionar con otros tamaños.

4.3. Cache asociativo con 4 ways y política de reemplazo LRU

Siguiendo el diseño del caché realizado en el Laboratorio 6 (cuyo diagrama de estados se encuentra en la Figura 2) se debe dividir el caché en espacios en aún más reducidos para hacer espacio para las 4 vías. El caché entonces efectivamente se le agrega un grado de profundidad de 4 vías como se ve ejemplificado en la visualización del caché de la Figura 5. En verilog el caché quedaría como una matriz o una memoria de 3 dimensiones, para una caché de 1 KB y bloques de 2 palabras quedaría (según la Tabla 1):

- 8 bytes por bloque
- $\frac{1 \text{ KB}}{8 \text{ bytes/bloque} \cdot 4 \text{ vías}}$ =32 bloques/vía
- Offset de 3 bits
- Set de 5 bits
- Tag de 24 bits

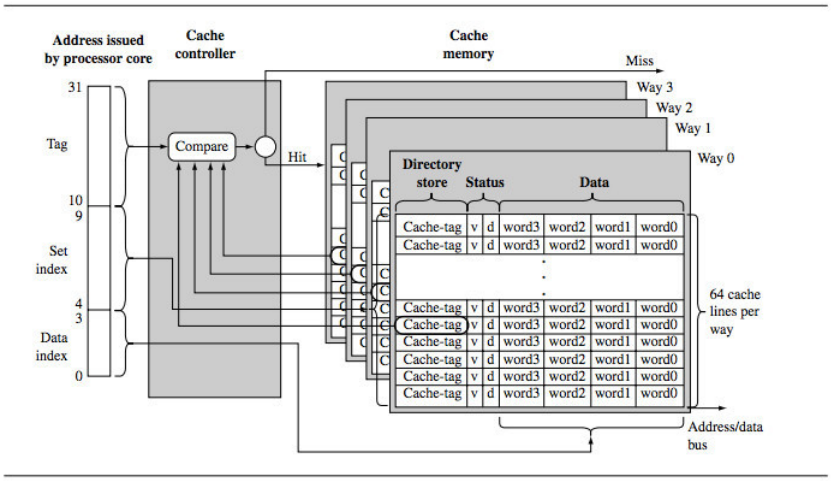

Figura 4: Visualización del caché con 4 vías. Tomado de https://www.researchgate.net/figure/4-KB-4-way-set-associative-cache-with-256-cache-lines_fig3_318860805

La política de reemplazo LRU es aplicado a las vías, es decir si las 4 vías están llenas y se da un *miss* se debe reemplazar algún bloque de alguna vía: Vía 0, Vía 1, Vía 2 o Vía 3. Para esto se debe mantener un indicador de bloque MRU (Most Recently Used) y otro LRU (Least Recently Used).

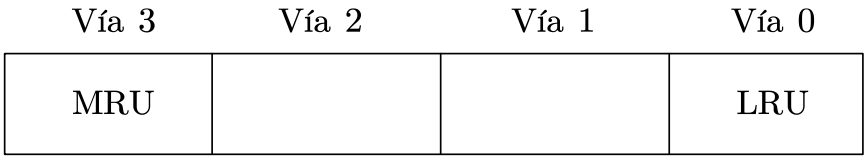

Figura 5: Ejemplo de indicadores de vías MRU y LRU. El bloque en la vía 3 es el LRU y será reemplazado para el próximo *miss* del caché.

La funcionalidad de *hit/miss* se debe modificar para acomodar la lógica LRU, para esto se diseñó el diagrama de flujo de la Figura 6. Se diseñaron los indicadores MRU y LRU como los extremos de 4 posibles valores en cada vía donde 0 representa el LRU y 3 el MRU. La tabla de frecuencia de uso pretende mantener estos valores para cada bloque en cada vía a lo largo del caché. De esta forma solo se debe modificar un indicador cada vez que haya un hit o un miss en lugar de estar cambiando de posiciones bloques en el caché.

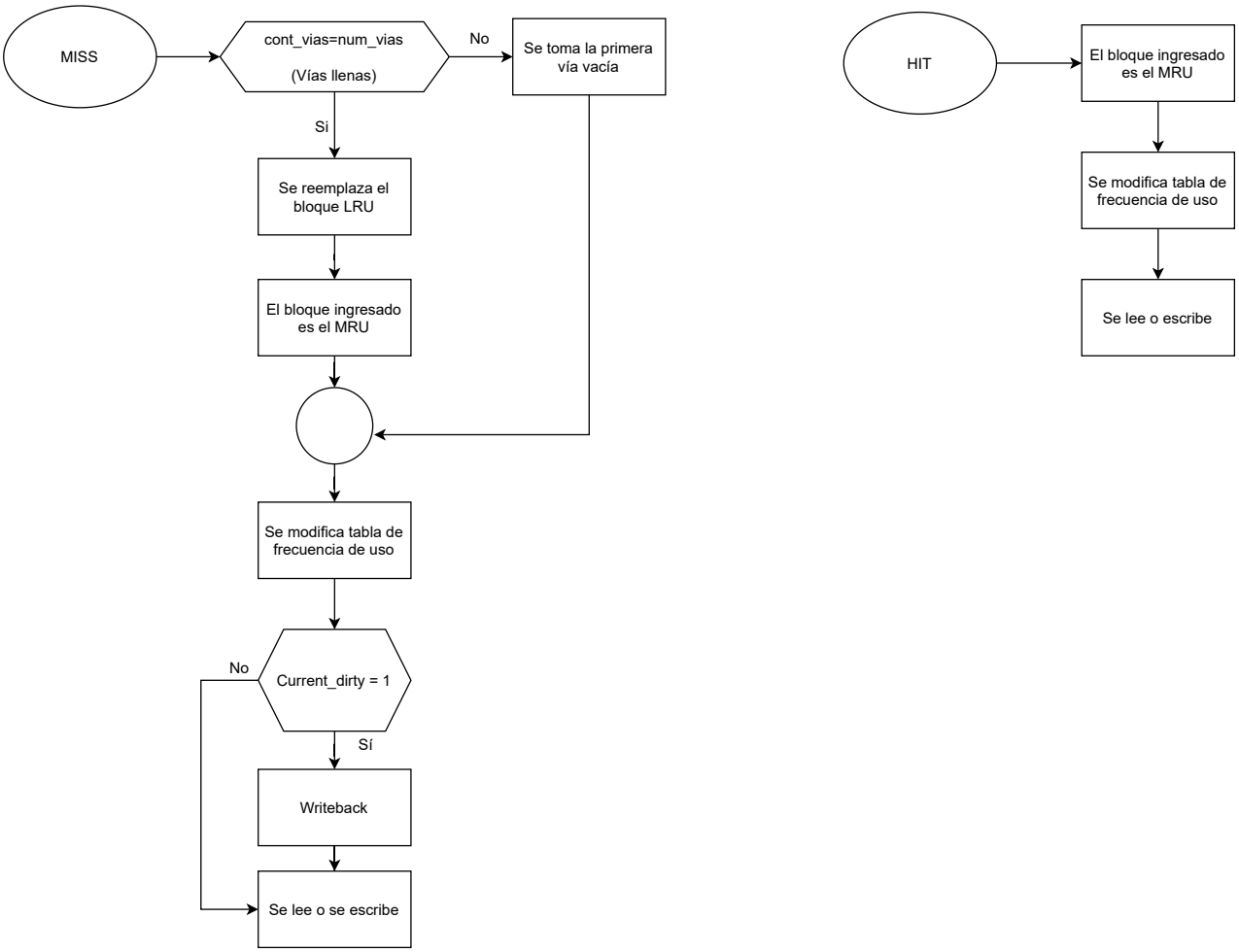

Figura 6: Cambios en lectura y escritura dependiendo de si es hit o miss para el caché asociativo con 4 ways y política de reemplazo LRU.

4.4. Modificación de los tamaños del cache

La caché diseñada es parametrizable, por lo que debe funcionar con otros tamaños.

Referencias

[1] Chris Wright Andrew Sloss Dominic Symes. *ARM System Developer's Guide: Designing and Optimizing System Software*. 1.ª ed. The Morgan Kaufmann Series in Computer Architecture and Design. Morgan Kaufmann, 2004. ISBN: 1558608745,9781558608740,9780080490496.

[2] Maria George y Peter Alfke. *Linear Feedback Shift Registers in Virtex Devices*. 2017.

[3] Aamer Jaleel y col. "High performance cache replacement using re-reference interval prediction (RRIP)". En: *ACM SIGARCH Computer Architecture News* 38.3 (2010), págs. 60-71. DOI: 10.1145/1816038.1815971.