Diseño de una Cache de Primer Nivel con Tecnología DWM

Hugo Tárrega[1], Alejandro Valero[2], Vicente Lorente[1], Salvador Petit[1] y Julio Sahuquillo[1]

Resumen— **La cache de primer nivel es especialmente crítica para el rendimiento del procesador. Los procesadores actuales implementan esta cache con un tamaño relativamente pequeño debido a las restricciones de área existentes en el pipeline del procesador.**

Estas caches se implementan con tecnología SRAM, la cual, comparada con otras tecnologías, presenta una baja densidad de integración. Se han propuesto caches de nivel inferior con tecnologías magnéticas más densas como es el caso de DWM. Estas tecnologías presentan inconvenientes como la latencia introducida por el desplazamiento de la información necesario para leer/escribir. Este incremento de latencia es especialmente crítico para las caches de primer nivel.

En este trabajo se propone el diseño de una cache DWM para la cache de datos de primer nivel. El diseño introduce dos innovaciones principales (múltiples puertos y organización de conjuntos entrelazada). Además, se propone una nueva organización interna que elimina por completo la latencia por desplazamiento en un 73 % de los aciertos. La organización propuesta mejora en un 15 % el rendimiento global del procesador con respecto a una organización convencional.

Palabras clave— **Caches L1, tecnología DWM, racetrack.**

I. Introducción

En los procesadores comerciales actuales, la jerarquía de cache es un subsistema clave para el rendimiento, ya que reduce la diferencia de velocidad entre los núcleos del procesador y la memoria principal. Normalmente, la jerarquía de cache se organiza en al menos tres niveles. Las caches de primer nivel (L1) deben ser lo suficientemente pequeñas como para ser implementadas en el *pipeline* del procesador y cumplir con sus restricciones temporales. Para implementar una cache L1 se utilizada tradicionalmente la tecnología de memoria estática de acceso aleatorio (SRAM). Esta tecnología limita la capacidad de almacenamiento de la L1 a unas decenas de KB para cumplir las restricciones temporales. En cambio, la cache L2 es mucho mayor, aunque lo suficientemente pequeña como para permitir la ejecución fuera de orden y ocultar su latencia.

Debido al empuje constante tanto de la tecnología como de las aplicaciones, la jerarquía de cache evoluciona continuamente, tanto en el número de sus estructuras como en el diseño y la implementación de las mismas. Por ejemplo, Intel ha aumentado el tamaño de la cache de datos L1 desde los 32 KB incluidos en la microarquitectura Skylake hasta los 48 KB en Sunny Cove, mientras que la capacidad L2 de los procesadores Intel Xeon ha pasado de 256 KB en Broadwell a 1 MB actualmente. En cambio, la capacidad máxima de L3 por núcleo se ha mantenido constante o incluso se ha reducido (es decir, de 2,5 MB a 1,375 MB) cuando se introdujo la microarquitectura Skylake.

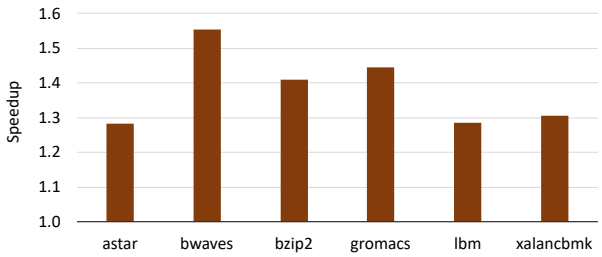


Fig. 1: *Speedup* de una única cache de datos L1 de 160 KB con respecto a una jerarquía convencional con una cache de datos L1 de 32 KB y una cache L2 de 128 KB.

Siguiendo esta tendencia se podría esperar que las caches L1 más grandes proporcionaran mejoras significativas en el rendimiento. Esta afirmación se puede verificar comparando el rendimiento ofrecido por una jerarquía de cache convencional consistente en una cache de datos L1 de 32 KB y una cache L2 de 128 KB frente al rendimiento ofrecido por una única cache de datos L1 con el mismo tiempo de acceso que la L1 convencional pero con capacidad de almacenamiento igual a la suma de ambas estructuras de cache (es decir, 160 KB)[1]. La Figura 1 muestra el aumento de velocidad conseguido con esta última configuración para seis aplicaciones con una alta demanda de memoria. Como puede observarse, los resultados muestran que las mejoras de rendimiento son cercanas o incluso superiores al 30 % respecto a la jerarquía convencional. Desafortunadamente, esta organización de cache no puede implementarse con la tecnología SRAM actual manteniendo las restricciones de área y tiempo de L1. Este trabajo persigue el diseño de L1 mediante tecnologías alternativas y compatibles con CMOS, las cuales podrían utilizarse para evitar estas limitaciones.

La tecnología de memoria RAM dinámica embebida (eDRAM) [17], utilizada en las cache de último nivel (LLC) de algunos procesadores comerciales [3], [14], presenta una densidad superior a SRAM, del orden de 8× [22]. Esta característica ha llevado a varios enfoques académicos a proponer celdas basadas en eDRAM para caches L1 [15], [29]. Sin embargo, las operaciones de lectura en eDRAM son destructivas y se requieren operaciones de refresco, lo cual complica la lógica de la cache y compromete la disponibilidad de los datos.

[1]Dpto. de Informática de Sistemas y Computadores, Universitat Politècnica de València, contacto: hutarsan@inf.upv.es

[2]Dpto. de Informática e Ingeniería de Sistemas, Universidad de Zaragoza.

[1]Se refiere al lector a la sección IV para más detalles acerca del entorno experimental.

STT-RAM [35] es una tecnología de memoria magnética y no volátil (NVM), más densa que SRAM y que también se ha explotado para implementar LLCs [33], [10], [23], [32], [2], [37]. Sin embargo, las operaciones de escritura en las celdas STT-RAM consumen mucha energía y son mucho más lentas que en SRAM. Para superar estos problemas, las propuestas centradas en L1 sacrifican la densidad con diseños híbridos SRAM/STT-RAM [9], [36] o relajan el tiempo de retención [25], [34], [12], [13] que, como en eDRAM, conlleva operaciones de refresco.

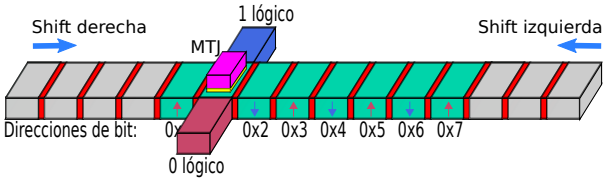
Las memorias *racetrack*, como *Domain Wall Memory* (DWM) [20], son memorias no volátiles que presentan tanto una densidad extrema como una gran eficiencia energética. Estas memorias están formadas por finas cintas de nanohilos magnéticas, denominadas *racetracks*, que contienen valores lógicos, llamados dominios. Para poder acceder a ellos, los dominios se desplazan y se alinean bajo un puerto de acceso pasando una corriente a lo largo de la cinta. Este enfoque, comúnmente conocido como DWM multi-bit, ha demostrado tener una densidad de almacenamiento 12× y 28× más alta que STT-RAM y SRAM, respectivamente [4].

Cuando el puerto de acceso está cerca del dominio de destino, el desplazamiento de la cinta para alinear el puerto y acceder al dominio lleva unos pocos ciclos de procesador. Sin embargo, el tiempo máximo de acceso está definido por la longitud de la cinta. Por ejemplo, suponiendo que el desplazamiento de un dominio conlleva un ciclo, el tiempo de acceso a una cinta de 64 dominios puede ser de 63 ciclos en el peor de los casos, siendo superior a las latencias de las LLCs actuales (e.g., la latencia de L3 en Skylake es de 40 ciclos). Para hacer frente a esta problemática, numerosas investigaciones se han centrado en la propuesta de organizaciones de la LLC y en políticas de gestión de los puertos de acceso para minimizar el impacto de las operaciones de desplazamiento [30], [26], [5], [11], [16], [6].

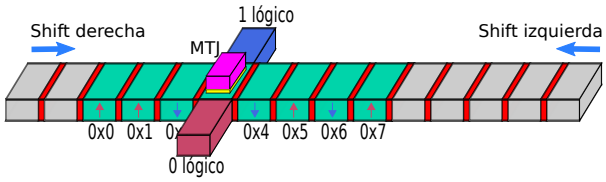
A diferencia de estos trabajos, en el presente trabajo se diseña una organización de cache basada en DWM con tiempos de acceso de unos pocos ciclos de reloj, lo cual permite utilizar esta tecnología en las caches L1 del pipeline del procesador. Además, gracias a la alta densidad de DWM, este trabajo propone combinar tanto la cache de datos L1 como la cache L2 en un único nivel de cache, mejorando el rendimiento de diseños convencionales de jerarquía de cache basados en SRAM y permitiendo un ahorro adicional de área.

II. Antecedentes

Esta sección examina los fundamentos físicos de la tecnología DWM y cómo se organizan los dominios en una cinta. La sección también se refiere cómo se realizan las operaciones lectura y escritura en DWM y a diferentes políticas de gestión de los puertos de acceso del estado-del-arte.



(a) Estado inicial



(b) Estado tras un desplazamiento de 2 dominios a derechas

Fig. 2: Celda DWM multi-bit.

A. Fundamentos Físicos y Organización

La Figura 2 muestra una implementación simplificada de una celda DWM de varios bits. Esta celda consiste en una cinta magnética y una unión de túnel magnético (puerto MTJ). La cinta está compuesta por un hilo ferromagnético que contiene múltiples dominios válidos (resaltados en color verde) y dominios auxiliares (en gris) para evitar pérdidas de datos en los desplazamientos o *shifts*. A su vez, los dominios están separados por pequeños delimitadores de dominio (en rojo). De este modo, cada dominio se magnetiza por separado para definir un bit de información. La dirección en la que se magnetiza cada dominio representa los valores lógicos '0' o '1'. El puerto MTJ está compuesto por tres capas, donde la capa inferior libre (en verde y formando parte de la cinta) corresponde al dominio accedido. La capa superior (en rosa) también es una capa ferromagnética, denominada capa fija, magnetizada en una dirección fija que actúa como referencia cuando se accede a un dominio. La capa fija y la capa libre están separadas por una barrera de MgO ultrafina (en amarillo) no conductora.

Este diseño permite realizar dos operaciones principales: lectura y escritura. La lectura del valor lógico almacenado en un dominio se realiza midiendo la magnetorresistencia del túnel del MTJ, que cambia según la dirección de magnetización del dominio. Esto permite distinguir entre un '0' y un '1'.

Una forma eficaz y rápida de realizar las operaciones de escritura es mediante el desplazamiento de dominios fijos [7]. Estos dominios fijos (en borgoña y azul) atrapan al MTJ y se establecen con valores '0' y '1' fijos. De esta manera, los dominios fijos se utilizan para escribir el valor lógico correspondiente en el dominio libre sin intervención del MTJ. De esta manera, aunque se añaden dos dominios extra por cada cabezal, se consiguen escrituras rápidas y con bajo consumo.

Para realizar una operación de lectura o escritura se debe acceder al dominio de destino alineándolo bajo el MTJ. Para ello, pueden ser necesarias múltiples operaciones de desplazamiento (a la izquierda o a la derecha). En este sentido, se aplican tantos pulsos de corriente a la cinta como dominios haya que desplazar.

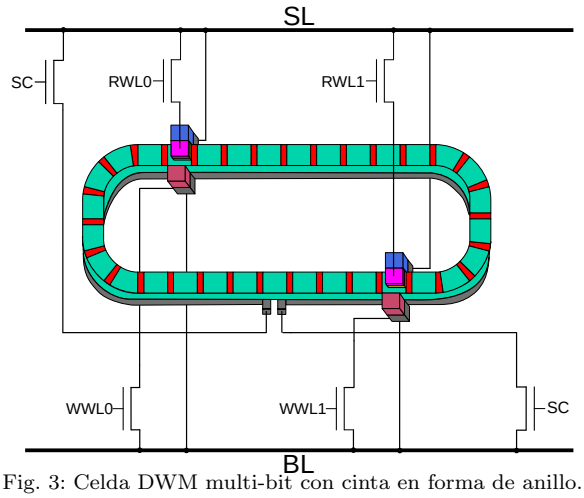


Fig. 3: Celda DWM multi-bit con cinta en forma de anillo.

Cada pulso suele durar un ciclo de reloj [26]. La Figura 2a muestra un posible estado inicial en el que el MTJ está alineado con el dominio en la dirección 0x3. Supóngase que la siguiente dirección solicitada es 0x1. En ese caso se requieren dos desplazamientos (es decir, dos ciclos de reloj) para posicionar el dominio de destino bajo el MTJ. La Figura 2b muestra el estado final de la cinta después de completar la operación de desplazamiento.

Una operación de desplazamiento puede implicar un desbordamiento de datos en los extremos del circuito, provocando la pérdida de los contenidos almacenados en los dominios afectados. Para superar este problema, una solución habitual es implementar cintas cuya longitud, medida en dominios, es mayor que el número de bits efectivos que se pueden almacenar. De este modo, para almacenar N bits, se requiere una cinta con $2N-1$ dominios. En la Figura 2, la cinta necesita 15 dominios para contener 8 bits. Recuérdese que los dominios que no se utilizan para almacenar información están resaltados en gris.

Una alternativa para evitar los dominios inútiles es considerar una cinta en forma de anillo [31]. La Figura 3 muestra un diagrama con una celda DWM de 32 bits en forma de anillo. La figura también ilustra cómo dos puertos MTJ proporcionan el acceso a diferentes dominios en una misma cinta.

Para realizar una lectura, cada MTJ se conecta a su correspondiente transistor de paso accionado por la línea de palabra (RWL_i), formando un puerto de lectura. Por el contrario, para realizar la escritura de un 1 en el dominio bajo el MTJ, SL se carga a 1, mientras que BL se conecta a tierra. De esta manera, la corriente circula desde SL hacia BL si la puerta del transistor WWL_i se conecta a 1, consiguiendo un desplazamiento de la polarización magnética presente en el dominio azul hacia el dominio debajo del MTJ en un sólo ciclo. Nótese que la escritura de un 0 ocurre manera similar, pero en sentido contrario, circulando la corriente desde BL cargada a 1 hacia SL cargada a 0 e involucrando al dominio borgoña.

Además de los puertos de acceso, la celda está conectada a dos transistores accionados por la señal SC. Dependiendo de los valores lógicos contrarios en las líneas SL y BL, se realiza un desplazamiento de la cinta en sentido horario o anti-horario.

B. *Políticas de Gestión de Puertos*

El acceso a un bit en una cinta requiere de una operación de desplazamiento previa para alinear el bit solicitado bajo un puerto MTJ. Dado que las operaciones de desplazamiento implican una latencia de acceso variable en un DWM, es imperativo minimizar el número de ciclos de desplazamiento tanto como sea posible.

Venkatesan *et al.* proponen diferentes políticas de gestión de puertos para una celda DWM multi-bit atendiendo a dos criterios diferentes: la selección y la actualización de puertos [30]. La selección de puertos establece cuál es el puerto adecuado para acceder a los datos solicitados. Existen dos enfoques, denominados estático y dinámico. El enfoque estático determina de antemano qué dominios específicos se pueden acceder a través de un puerto dado, mientras que el enfoque dinámico calcula el puerto más cercano en tiempo de ejecución.

La actualización del puerto define la posición de la cinta con respecto a los puertos tras la finalización de un acceso. Existen dos políticas, denominadas *eager* y *lazy*. La política *eager* restaura la posición de la cinta a su posición por defecto después de completar un acceso, mientras que la política *lazy* mantiene la última posición de la cinta hasta el siguiente acceso.

Este trabajo asume una selección dinámica de puertos para minimizar el número de ciclos requeridos en una operación de desplazamiento y una actualización lazy de la cinta para explotar la localidad espacial presente en las caches de datos L1. La sobrecoste hardware necesario para implementar esta política de gestión de puertos es insignificante. Se remite al lector a [30] para más detalles.

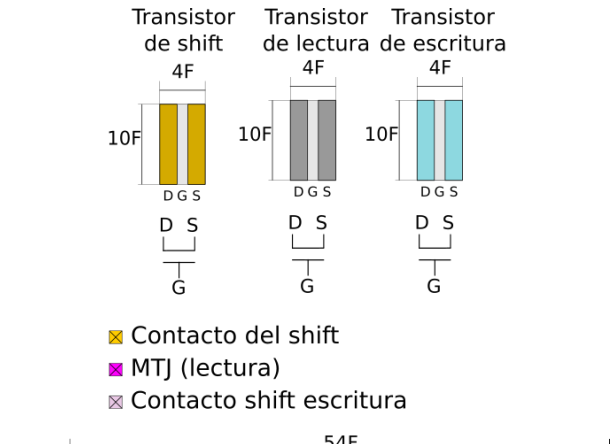
III. Propuesta de Cache L1 Basada en DWM

Esta sección presenta la propuesta de cache de datos L1 basada en tecnología DWM. Para ello, se presenta un diseño de cache ascendente. En primer lugar, se diseña una celda DWM multi-bit específica para L1. A continuación, la celda propuesta se utiliza para implementar un módulo DWM compuesto por líneas de cache y conjuntos. Finalmente, el módulo propuesto se utiliza para implementar el vector de datos de una memoria cache.

Nótese que este trabajo asume que el vector de etiquetas se implementa con tecnología SRAM.

A. *Celda DWM Multi-bit*

La latencia de acceso en una celda DWM viene determinada por la corriente que pueden proporcionar los transistores de acceso. Un tamaño de transistor de 10F establece un compromiso adecuado tanto para el rendimiento como para el área [4], [19]. Por otro lado, la latencia de desplazamiento está influida tanto por el número de dominios (longitud de la cinta magnética) como por el tamaño de los transistores utilizados para esta operación. Las cintas más largas requieren más



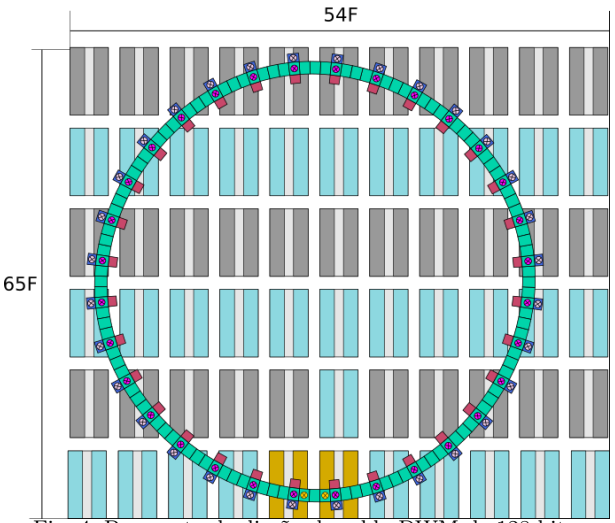


Fig. 4: Propuesta de diseño de celda DWM de 128 bits.

corriente para desplazar todos los dominios y, por tanto, transistores de desplazamiento más grandes. Para un tamaño de transistor de desplazamiento dado, la cinta debe ser lo más corta posible para que el desplazamiento sea rápido [31]. Este trabajo limita la longitud de la cinta en forma de anillo a 128 dominios para establecer transistores de desplazamiento con un tamaño de 10F y una latencia de desplazamiento dentro de un ciclo de reloj por dominio.

Para optimizar la disposición de los elementos de la celda, tanto los transistores de desplazamiento como los de acceso se sitúan bajo la cinta. El número de transistores de desplazamiento se limita a dos asegurar un desplazamiento en ambos sentidos. El número de transistores de acceso está determinado por el número de dominios entre dos puertos consecutivos. Este trabajo limita dicho número de dominios a cuatro. De este modo, se necesitan 32 puertos para la cinta de 128 dominios, lo que se traduce en 64 transistores de acceso (controlados por RWL_i y WWL_i). Nótese también que, suponiendo una gestión de puertos dinámica y *lazy*, la distancia de cuatro dominios entre puertos sucesivos define una latencia de desplazamiento máxima de dos ciclos para acceder al bit más lejano de un puerto.

La Figura 4 muestra el diseño de una celda DWM consistente en 11×6 transistores. Los transistores de desplazamiento están situados en la última fila de arriba a abajo. De acuerdo con los tamaños de los

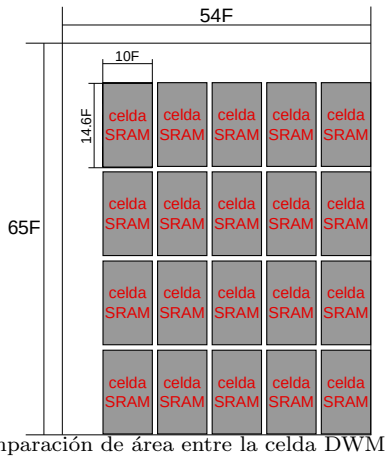


Fig. 5: Comparación de área entre la celda DWM propuesta de 128 bits y celdas SRAM de 1 bit.

transistores definidos y asumiendo un espacio de 1F entre transistores adyacentes, el área de la celda es $54F \times 65F = 3,510F^2$.

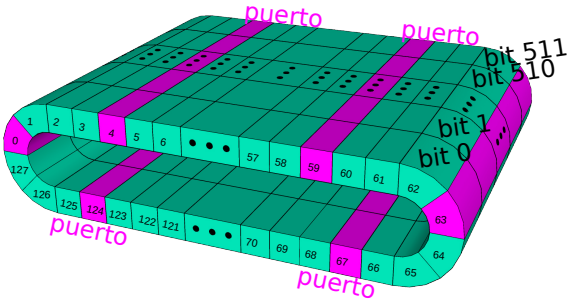
La Figura 5 muestra a simple vista el número de celdas SRAM que puede contener el área de la celda DWM propuesta. Teniendo en cuenta que una celda SRAM 6T típica ocupa una área de $10F \times 14{,}6F = 146F^2$ [27], caben hasta 20 celdas SRAM en dicha área. De este modo, la densidad de la celda propuesta es 6,4x respecto a SRAM (i.e., 128/20).

Al igual que en [31], este trabajo asume una longitud de 40 nm para cada dominio y una anchura entre dominios de 5 nm, por lo que cada unidad de almacenamiento tiene una longitud de 45 nm. Considerando un nodo tecnológico de 40 nm, la altura y anchura de la celda propuesta permitirían hasta 48 y 57 dominios, respectivamente, lo cual se ajustaría a una longitud ideal de cinta en forma rectangular de 210 dominios. Sin embargo, la longitud de la cinta se limita a 128 dominios no sólo para establecer la penalización por desplazamiento de un dominio dentro de un ciclo, sino también para simplificar el direccionamiento siendo 128 una potencia de 2.

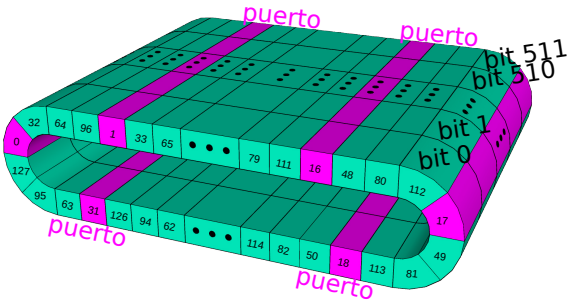
B. Módulo DWM

La celda DWM propuesta puede utilizarse para implementar un módulo de cache. Suponiendo un tamaño de línea de cache de 64 bytes, un módulo consta de 512 celdas DWM. Los dominios en la posición j-ésima de cada celda implementan la línea j-ésima del módulo. Esta línea se refiere a una línea de un conjunto con identificador j. De este modo, un módulo implementa 128 conjuntos, cada uno de los cuales consta de una única línea de 64 bytes. Todos los bits que implementan una línea comparten la señal de desplazamiento (SC) y las líneas de palabra (RWL_i y WWL_i), lo que permite desplazar todas las celdas al mismo tiempo y acceder en paralelo a una línea completa.

La Figura 6a muestra la distribución de las 512 celdas para implementar un módulo. El número dentro de un dominio se refiere al identificador de conjunto. Con fines ilustrativos, sólo se destacan algunos conjuntos. En una organización secuencial de conjuntos,



(a) Organización secuencial de conjuntos



(b) Organización entrelazada de conjuntos a nivel de puerto.

Fig. 6: Diseño de un módulo DWM. El módulo se organiza en 128 conjuntos que contienen una sola línea de 512 bits cada uno de ellos.

estos se distribuyen consecutivamente a lo largo del anillo. De esta manera, con un paso de cuatro conjuntos, los conjuntos 0, 4, 8, 12, 16, y así sucesivamente, se alinean bajo un puerto, constituyendo un estado inicial.

Una organización secuencial de conjuntos no explota la localidad espacial que muestran las caches de datos L1. Por este motivo, este trabajo propone una organización de conjuntos alternativa que ubica los conjuntos de forma intercalada en los puertos y correspondiente a la función *id set* % *num puertos*. De este modo, los conjuntos consecutivos se alinean con los puertos. La Figura 6b muestra una organización en la que los conjuntos desde 0 hasta 31 están alineados con los 32 puertos. De esta manera, el acceso a conjuntos vecinos no implica ninguna operación de desplazamiento. Obsérvese que, en la estrategia propuesta de puertos intercalados, el acceso a cualquier conjunto del rango 32-63 implicaría sólo una penalización de desplazamiento de un ciclo. Tras este desplazamiento, todos los conjuntos dentro de dicho rango permanecerán alineados bajo un puerto.

La Figura 7 muestra la eficacia de la propuesta basada en el entrelazado de conjuntos a nivel de puerto. La latencia por desplazamiento se desglosa en número de ciclos para las ambas estrategias secuencial (Sec) y de entrelazada (Ent). Una latencia de cero ciclos significa que no se requiere ninguna operación de desplazamiento, ya que el bloque solicitado está alineado bajo un puerto. Los resultados se muestran para una cache de datos L1 de 160 KB basada en DWM.

Al explotar la localidad espacial de los datos, la organización entrelazada mejora en gran medida la latencia de los desplazamientos con respecto a la organización secuencial en todas las aplicaciones. El porcentaje de latencia de cero ciclos aumenta particularmente en aplicaciones con una alta localidad espacial como es el caso de *GemsFDTD*, *gamess*, o *gromacs*, donde dicho porcentaje supera un 80 %. En general, la organización de conjuntos propuesta evita que se produzca una operación de desplazamiento en un 76 % de los accesos a la cache. Este porcentaje se reduce al 43 % en la versión secuencial. Para el resto de este artículo se asume la organización entrelazada de conjuntos dentro de un módulo DWM.

C. Organización del Vector de Datos

Un módulo compuesto por 128 conjuntos y una sola línea de 64 bytes por conjunto implementa un array de datos de 8 KB con mapeo directo. La alta densidad de la tecnología DWM permite implementar una capacidad de memoria cache mucho mayor con respecto a una cache convencional SRAM para una área determinada. La ampliación de la capacidad de la cache puede hacerse aumentando el tamaño de línea, el número de conjuntos o el número de vías. Esta sección explora la posibilidad de aumentar tanto el número de conjuntos como las vías bajo la restricción de área definida por el diseño de la cache convencional.

Para aumentar el número de conjuntos más allá de 128, este trabajo propone utilizar múltiples módulos para implementar una cache. Cada módulo contiene un subconjunto de 128 conjuntos adyacentes de la cache. Esta distribución de conjuntos en módulos beneficia la localidad espacial, ya que las operaciones de desplazamiento sólo afectan al módulo de destino donde reside el conjunto accedido. Esto evita que los módulos restantes realicen una operación de desplazamiento, manteniendo el mismo estado de acuerdo con su último acceso. El número total de conjuntos de cache se limita a un múltiplo de 128 para explotar todos los dominios disponibles de un módulo.

Al contrario que las caches de último nivel, las caches L1 asociativas de conjuntos suelen acceder en paralelo a todas las etiquetas y vías del conjunto para reducir el tiempo de acceso. Teniendo en cuenta este comportamiento, las diferentes vías de la cache DWM propuesta se separan físicamente en diferentes bancos implementados con diferentes módulos. Esta decisión de diseño proporciona un tiempo de acceso mínimo porque una misma operación de desplazamiento se lleva a cabo en todas las vías del conjunto de destino al mismo tiempo que la comparación de etiquetas.

De acuerdo con la mayor densidad de almacenamiento que ofrece la tecnología DWM, este trabajo explora combinar la cache de datos L1 y la cache L2, implementándolas en un único nivel de cache L1. De esta manera, la cache L2 se elimina por completo de la jerarquía de memoria propuesta. Bajo esta consideración, se exploran dos posibles estrategias de diseño denominadas iso-área e iso-capacidad.

El enfoque iso-área trata de implementar una cache DWM que ocupe la misma área que la cache de datos L1 original más el área de la cache L2. En este sentido, teniendo en cuenta que la densidad de la

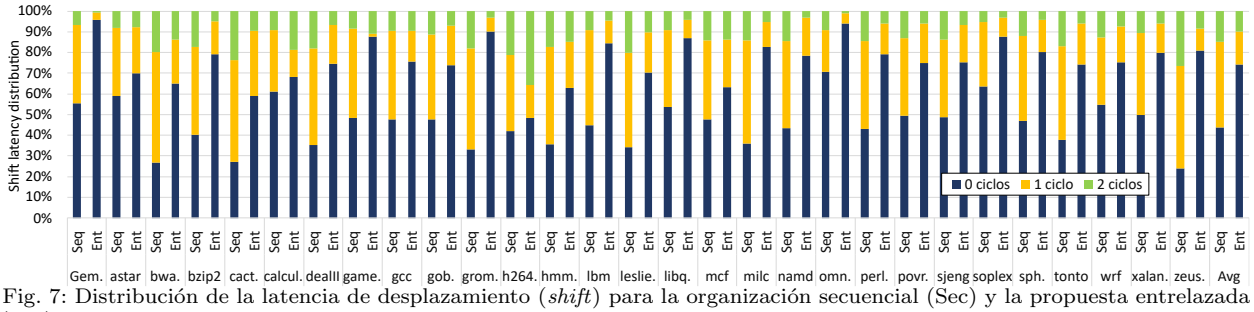


Fig. 7: Distribución de la latencia de desplazamiento (*shift*) para la organización secuencial (Sec) y la propuesta entrelazada (Ent) en una cache L1 de 160 KB basada en DWM.

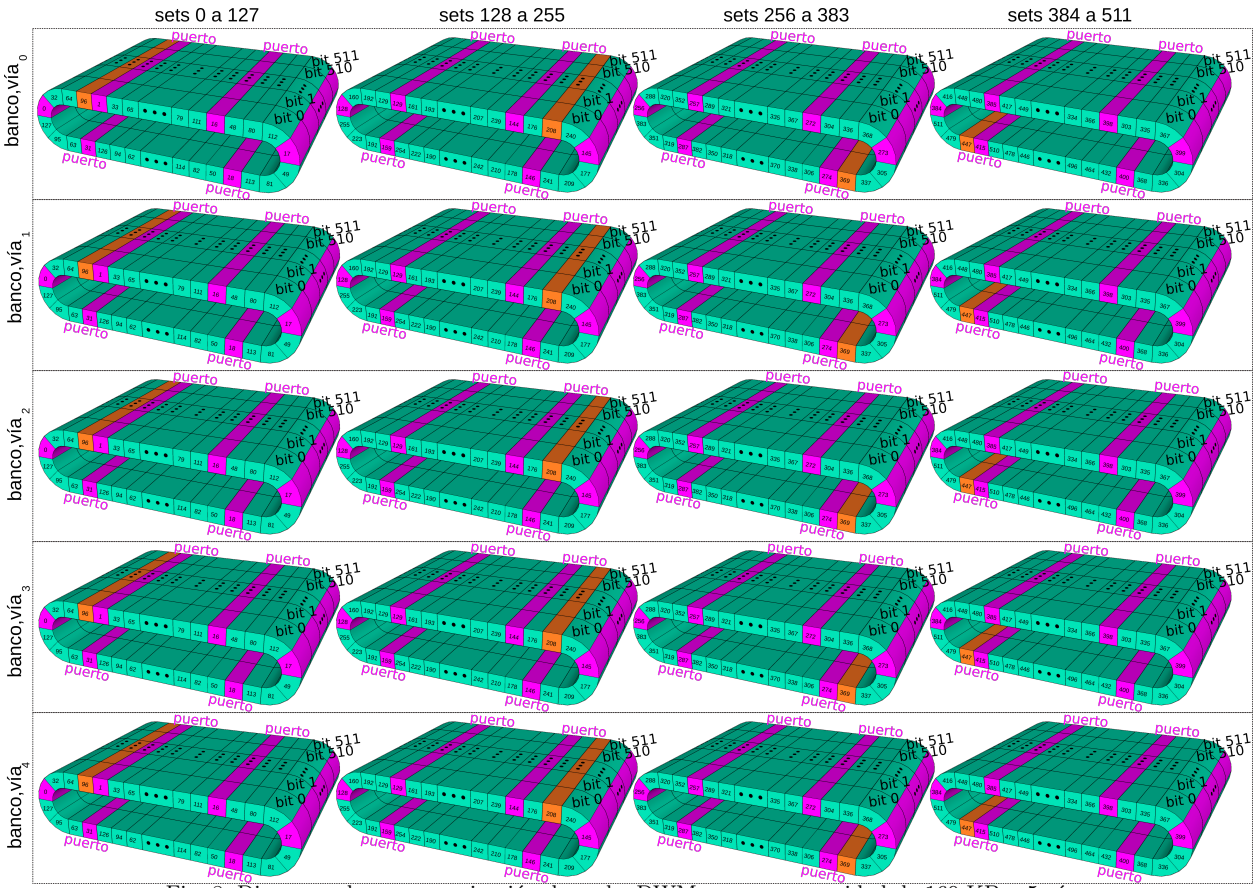


Fig. 8: Diagrama de una organización de cache DWM con una capacidad de 160 KB y 5 vías.

celda DWM propuesta es $6,4\times$ con respecto a SRAM (véase Sección III-A), se podría implementar una memoria cache de aproximadamente 1 MB en lugar de una cache de datos L1 de 32 KB y una cache L2 de 128 KB. Sin embargo, el principal reto de este enfoque es mantener una velocidad de acceso rápida. Esto se debe principalmente a un mayor sobrecoste temporal de los circuitos periféricos en una área mucho mayor en comparación con una cache L1 convencional. Por este motivo, este artículo deja el enfoque iso-área para trabajo futuro.

El enfoque iso-capacidad implementa una cache de datos L1 con la misma capacidad que una cache de datos L1 convencional más una cache L2. Por ejemplo, una jerarquía de memoria de dos niveles formada por una cache de datos L1 de 32 KB y una cache L2 de 128 KB se implementaría como una única cache de datos L1 de 160 KB. Este enfoque elimina potencialmente el área requerida por la L2 e incluso reduce el área original ocupada por la cache L1, dependiendo del tamaño de la L2. A modo de ejemplo, una cache DWM de 160 KB reduce el área en comparación con el área original de la cache de datos L1 original de 32 KB. En comparación con una jerarquía de memoria inclusiva, la idea clave detrás del enfoque de iso-capacidad es mejorar el rendimiento del sistema aumentando la capacidad de almacenamiento efectiva y manteniendo el tiempo medio de acceso por debajo del de una cache L2 a pesar de la necesidad de operaciones de desplazamiento.

La Figura 8 muestra cómo se organizan los conjuntos (*sets*) y las vías en módulos para una configuración de cache de 160 KB y 5 vías. Nótese que cada columna de módulos implementa un número de conjuntos consecutivos, empezando de izquierda a derecha por el rango 0-127. Por otro lado, cada fila implementa un banco/vía diferente. De esta forma, se puede realizar un acceso a todas las vías de un conjunto en paralelo en los módulos de destino. Por ejemplo, un acceso a todas las vías del conjunto resaltado en

Tabla I: Parámetros de la máquina.

Núcleo del procesador	
Política de issue	Fuera de orden
Predictor de saltos	Hybrid gshare/bimodal
Ancho de fetch, issue y commit	4 instructiones/ciclo
Tamaño ROB (entradas)	244
# ALUs enteros/reales	4/4
Jerarquía de memoria convencional on chip	
Todas las caches	64 B/línea, LRU
Cache inst. L1 SRAM privada	32 KB, 4 vías, 2 ciclos
Cache datos L1 SRAM privada	32 KB, 4 vías, 2 ciclos
Cache L2 SRAM privada	128 KB, 8 vías, 5 ciclos
Cache L3 SRAM compartida	8 MB, 16 vías, 20 ciclos
Jerarquía de memoria on chip propuesta	
Todas las caches	64 B/línea, LRU
Cache inst. L1 SRAM privada	32 KB, 4 vías, 2 ciclos
Cache datos L1 DWM privada	160 KB, 5 vías, 2 ciclos + desplaz.
Cache L2 SRAM compartida	8 MB, 16 vías, 20 ciclos
Memoria principal DRAM	120 ciclos

naranja en primera columna de izquierda a derecha se realiza simultáneamente en todos los módulos de la columna mencionada, lo cual requiere una operación de desplazamiento de un dominio en todos ellos. Por el contrario, un acceso al conjunto naranja ubicado en la siguiente columna implica una operación de desplazamiento de dos dominios en todos sus módulos.

IV. Resultados Experimentales

El diseño de cache DWM propuesto se ha modelado y evaluado con el simulador ciclo a ciclo Multi2Sim [28]. Los resultados de la simulación incluyen estadísticas individuales de múltiples componentes del microprocesador, incluyendo la jerarquía de memoria, así como el rendimiento del sistema. Todos los tiempos de acceso se han obtenido con los simuladores CACTI [27] y DESTINY [18].

Los resultados experimentales se obtuvieron para todo el conjunto de aplicaciones de SPEC CPU 2006 con los datos de entrada de referencia [1]. Las estadísticas se recogieron simulando 500 millones de instrucciones después de omitir los 800 millones de instrucciones iniciales y calentar las caches para 200 millones de instrucciones. La cache de datos L1 DWM propuesta se evalúa en un procesador de cuatro núcleos y un único hilo en cada núcleo. La tabla I resume los principales parámetros arquitectónicos de un núcleo del microprocesador y la jerarquía de memoria utilizada.

Para proporcionar una mayor información acerca del rendimiento, esta sección evalúa en primer lugar la distribución de la latencia en la jerarquía de cache y los fallos por kilo-instrucción (MPKI). Por último, se analiza el impacto en el rendimiento del sistema.

A. Distribución de Latencia

La Figura 9 representa la distribución de la latencia en la jerarquía de cache desde el 45 % hasta el 100 %. Los resultados se muestran para el subsistema de memoria convencional y el propuesto basado en DWM. Recordemos que el esquema convencional cuenta con una latencia constante en L1 de 2 ciclos, mientras que la cache L1 basada en DWM tiene una latencia variable de 2 a 4 ciclos y prescinde de la cache L2 de 5 ciclos.

Como era de esperar, algunas aplicaciones (10 de 29) presentan una distribución de latencia L1 superior al 98 % en la cache L1 convencional. Sin embargo, algunos otros benchmarks, con una localidad temporal relativamente baja, como *bwaves*, *gcc* y *mcf*, presentan un menor número de aciertos en la cache L1. Por otro lado, al combinar las caches L1 y L2 en un único nivel de cache, el enfoque DWM sustituye la penalización de la latencia L2 de 5 ciclos por una latencia de 4 ciclos como máximo, en función de la operación de desplazamiento requerida, aliviando la presión en la jerarquía de cache al actuar como un filtro respecto al nivel de cache inferior.

En algunas aplicaciones (10 de 29), como *gromacs*, *lbm* y *mcf*, la propuesta basada en DWM reduce el porcentaje de aciertos en la cache L3 (en realidad se trata de una cache L2) con respecto al esquema convencional. En otras palabras, DWM obtiene un mayor porcentaje de aciertos en comparación con el porcentaje de aciertos combinado de las caches L1 y L2 del diseño convencional. Esto se debe a dos razones principales. En primer lugar, un bloque puede estar presente tanto en la cache L1 como en L2 en el diseño convencional, lo cual reduce la capacidad efectiva de almacenamiento con respecto al diseño propuesto. En segundo lugar, DWM tiene un mayor número de conjuntos de cache (es decir, 512 conjuntos) con respecto a la combinación de las caches L1 y L2 convencionales (i.e., $128 + 256 = 384$ conjuntos). Esto significa que DWM reduce los fallos de capacidad respecto al esquema convencional. Por otro lado, en algunos otros benchmarks, como *bwaves* y *gamess*, DWM muestra un mayor número de aciertos en L3. Esto se debe principalmente a que la organización de cache propuesta tiene menos vías con respecto a las caches L1 y L2 convencionales, lo que implica un mayor número de fallos de conflicto.

En general, el 73 % de los aciertos no requieren ninguna operación de desplazamiento para acceder a los datos solicitados en la cache DWM.

B. Fallos por Kilo-Instrucción

El estudio anterior no muestra explícitamente si el enfoque propuesto reduce el ratio de fallos en la cache de datos L1 con respecto a los ratios combinados de fallos en la cache de datos L1 y la cache L2 del diseño convencional. Para ello, esta sección evalúa los fallos por kilo-instrucción (MPKI).

La Figura 10 ilustra los resultados de MPKI. El diseño convencional diferencia entre MPKI de la cache de datos L1 y de la cache L2. DWM mitiga en gran medida el número de fallos en aquellas aplicaciones con valores de MPKI medios y altos. Esto se debe principalmente a que la capacidad de almacenamiento efectiva de la propuesta de cache DWM es mayor en comparación con el diseño convencional con bloques duplicados en la caches L1 y L2. Este efecto se aprecia notablemente en aplicaciones con requerimientos de memoria como *astar* y *bwaves*.

Otras aplicaciones con requerimientos de memoria

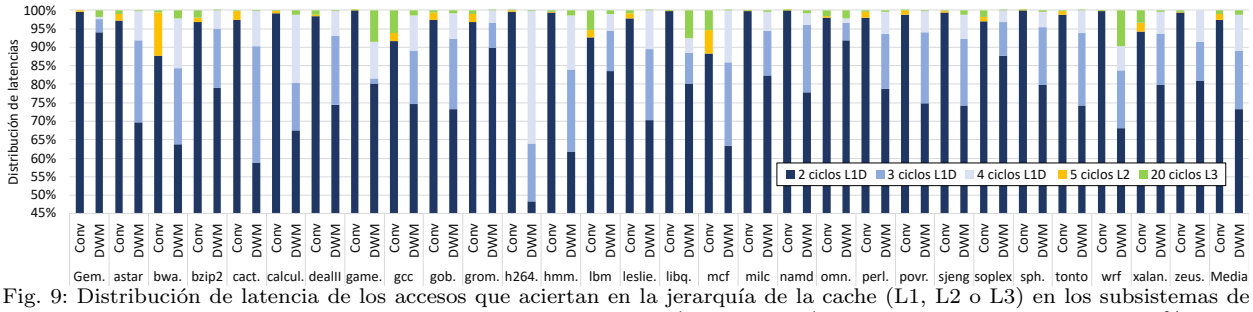


Fig. 9: Distribución de latencia de los accesos que aciertan en la jerarquía de la cache (L1, L2 o L3) en los subsistemas de memoria convencional y propuesto. Los accesos a memoria principal (no mostrados) representan de media menos del 1 %; sólo 3 de los 29 benchmarks superan el 4 %.

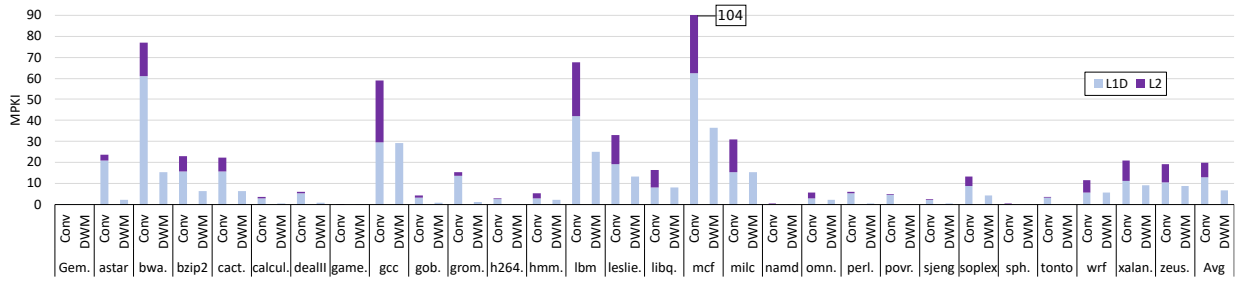


Fig. 10: Fallos por kilo-instrucción (MPKI) en los enfoques convencional y propuesto. El esquema convencional distingue entre MPKI de la cache de datos L1 y de la cache L2.

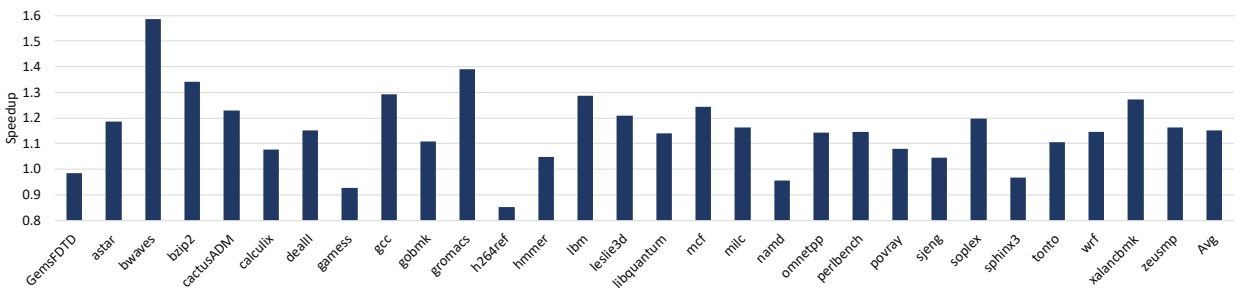


Fig. 11: *Speedup* del diseño de cache DWM frente al diseño convencional.

mucho menores, como *gobmk* y *perlbench*, muestran un MPKI bastante bajo, pero incluso en estos benchmarks la cache DWM reduce el número de fallos. Por último, el conjunto de datos de unos pocos benchmarks como *GemsFDTD*, *gamess* y *namd* es lo suficientemente pequeño como para estar contenido en su mayor parte en la cache L1, mostrando por tanto valores de MPKI insignificantes en ambos subsistemas de memoria.

En general, la cache DWM reduce el MPKI medio en un 60 % con respecto al diseño convencional.

C. Rendimiento del Sistema

Esta sección analiza el rendimiento del sistema, tomando para ello la métrica de Instrucciones por Ciclo (IPC). La Figura 11 muestra el *speedup* de la propuesta de cache DWM frente al diseño convencional. En aquellas aplicaciones con un MPKI insignificante, como es el caso de *GemsFDTD*, *gamess* y *namd*, el *speedup* es nulo o incluso ligeramente por debajo de la unidad. En el caso de *h264ref*, se produce un *slowdown* mayor que un 10 %. Esto se debe a que es la única aplicación con una latencia de 2 ciclos en L1 por debajo de un 50 % en la distribución de latencia (véase la Figura 9). Sin embargo, para el resto de aplicaciones, la mejora en el rendimiento resulta considerable, llegando hasta mejoras de un 40 % para *gromacs* o un 60 % para *bwaves*. En promedio, el rendimiento mejora en un 15 % respecto al diseño convencional.

V. Trabajos Relacionados

Las propuestas de cache anteriores basadas en DWM se centran principalmente en el último nivel de cache, concretamente en la cache L2.

La arquitectura TapeCache implementa la cache L1 con celdas DWM de un bit, es decir, con un puerto MTJ por dominio en la cinta, mientras que la cache L2 combina las celdas anteriores con celdas DWM multi-bit. [30]. La cache L2 de n vías se organiza en una única vía rápida implementada con celdas DWM de un único bit, mientras que el resto de $n-1$ vías se implementan con celdas multi-bit y se denominan como densas. La vía rápida almacena los bloques más recietemente utilizados e identificados por el algoritmo de reemplazo LRU, mientras que las vías densas almacenan el resto de bloques. Para ello, se requieren operaciones de intercambio entre la vía rápida y las densas.

Sun *et al.* proponen una organización de cache

DWM multi-bit donde los conjuntos se implementan mediante cintas consecutivas por encima del área de los transistores de acceso [24]. La longitud de las cintas determina el número de vías por conjunto, las cuales se almacenan de manera consecutiva en una cinta, mientras que el número de cintas por conjunto queda determinado por el tamaño de bloque, siguiendo una organización entrelada de bits. Las instrucciones y datos se asignan estáticamente a conjuntos específicos, y se acceden mediante puertos de sólo lectura y de lectura/escritura, respectivamente. Además, el número de vías activas de un conjunto de datos se ajusta dinámicamente de acuerdo con las demandas de la aplicación en tiempo de ejecución, reduciendo el número de operaciones de desplazamiento.

De manera similar a [24], el diseño referido como HDART emplea cintas sucesivas por encima del área definida por los transitores de acceso [26]. Una cinta i-ésima es lo suficientemente larga como para almacenar los bits i-ésimos de un subconjunto de vías consecutivas de un número dado de conjuntos sucesivos. De acuerdo con una distribución diagonal de los puertos de acceso, un subconjunto de las mismas vías no consecutivas de múltiples conjuntos quedan alineadas con los puertos y se acceden simultáneamente. Además, los autores proponen una gestión de los datos mediante la cual los bloques con un acceso intensivo se mantinen cerca de los puertos mediante operaciones de intercambio.

La arquitectura de cache SKM propone una organización entrelazada a nivel de bits donde subconjuntos de cintas conteniendo un bloque entero se organizan en bancos [5]. La cinta i-ésima de un banco almacena los bits i-ésimos de conjuntos consecutivos. De esta manera, todas las vías de un conjunto dado se pueden acceder en paralelo. Este trabajo tiene en cuenta las variaciones en el proceso de fabricación, lo cual conlleva bancos con tiempos de acceso diferentes. Los autores proponen una política de migración de datos que permite mantener los bloques con un uso intensivo en bancos rápidos.

Otros trabajos proponen políticas de gestión de los puertos de acceso y se basan principalemnte en la TapeCache [6], [8], [21].

VI. Conclusiones

La alta densidad de la tecnología DWM comparada con la tecnología SRAM permite diseñar caches del procesador con una mayor capacidad. Algunos trabajos previos han demostrado que es posible diseñar caches minimizando sus inconvenientes (e.g., alta latencia de desplazamiento de la cinta), lo cual que permite mejorar significativamente el rendimiento del procesador. Estos diseños se han centrado principalmente en las caches de bajo nivel (L2 o L3) donde la latencia no es crítica y se puede solapar con la ejecución fuera de orden.

Por el contrario, este trabajo ha presentado el diseño de una cache L1 basada en DWM, cuya latencia es crítica para el rendimiento. En consecuencia, el diseño minimiza el desplazamiento de la cinta. Para ello, se han presentado dos innovaciones principales: i) implementación de varios puertos por cinta limitando la latencia máxima de desplazamiento a dos ciclos, ii) implementación de un entralazado de conjuntos a nivel de puertos para explotar la localidad espacial.

Los resultados experimentales han mostrado que, comparada contra una jerarquía de cache (L1+L2) convencional con la misma capacidad (32KB+128KB), la propuesta mejora el rendimiento notablemente alcanzando un 15 % en promedio, y llegando en algunas aplicaciones hasta un 60 % de mejora. Estas mejoras se deben a que la organización propuesta consigue reducir el MPKI frente al diseño convencional, además de eliminar la necesidad del desplazamiento en hasta un 73 % de los aciertos y por diseño, nunca superando dos ciclos.

Agradecimientos

Este trabajo ha sido financiado por el Ministerio de Ciencia, Innovación y Universidades y el FEDER europeo bajo los proyectos RTI2018-098156-B-C51 y PID2019-105660RB-C21, el Gobierno de Aragón (grupo T58_20R) y FEDER 2014-2020 “Construyendo Europa desde Aragón”.

Referencias

[1] *Standard Performance Evaluation Corporation, available online at http://www.spec.org/cpu2006/.*

[2] J. Ahn, S. Yoo, and K. Choi. DASCA: Dead Write Prediction Assisted STT-RAM Cache Architecture. In *Proceedings of the IEEE 20th International Symposium on High Performance Computer Architecture*, pages 25–36, 2014.

[3] J. Barth, D. Plass, E. Nelson, C. Hwang, G. Fredeman, M. A. Sperling, A. Mathews, T. Kirihata, W. R. Reohr, K. Nair, and N. Caon. A 45 nm SOI Embedded DRAM Macro for the POWER™ Processor 32 MByte On-Chip L3 Cache. *IEEE Journal of Solid-State Circuits*, 46(1):64–75, 2011.

[4] C. Zhang, G. Sun, W. Zhang, F. Mi, H. Li, and W. Zhao. Quantitative Modeling of Racetrack Memory, a Tradeoff among Area, Performance, and Power. In *Proceedings of the 20th Asia and South Pacific Design Automation Conference*, pages 100–105, 2015.

[5] F. Chen, Z. Li, W. Kang, W. Zhao, H. Li, and Y. Chen. Process Variation Aware Data Management for Magnetic Skyrmions Racetrack Memory. In *Proceedings of the 23rd Asia and South Pacific Design Automation Conference*, pages 221–226, 2018.

[6] A. Colaso, P. Prieto, P. Abad, J. A. Gregorio, and V. Puente. Architecting Racetrack Memory Preshift through Pattern-Based Prediction Mechanisms. In *Proceedings of the IEEE International Parallel and Distributed Processing Symposium*, pages 273–282, 2019.

[7] S. Fukami, T. Suzuki, K. Nagahara, N. Ohshima, Y. Ozaki, S. Saito, R. Nebashi, N. Sakimura, H. Honjo, K. Mori, C. Igarashi, S. Miura, N. Ishiwata, and T. Sugibayashi. Low-Current Perpendicular Domain Wall Motion Cell for Scalable High-Speed MRAM. In *Proceedings of the IEEE Symposium on VLSI Technology*, pages 230–231, 2009.

[8] H. Xu, Y. Li, R. Melhem, and A. K. Jones. Multilane Racetrack Caches: Improving Efficiency Through Compression and Independent Shifting. In *Proceedings of the 20th Asia and South Pacific Design Automation Conference*, pages 417–422, 2015.

[9] M. Imani, S. Patil, and T. Rosing. Low Power Data-Aware STT-RAM Based Hybrid Cache Architecture. In *Proceedings of the 17th International Symposium on Quality Electronic Design*, pages 88–94, 2016.

[10] A. Jog, A. K. Mishra, C. Xu, Y. Xie, V. Narayanan, R. Iyer, and C. R. Das. Cache Revive: Architecting Volatile STT-RAM Caches for Enhanced Performance in CMPs. In *Proceedings of the Design Automation Conference*, pages 243–252, 2012.

[11] A. A. Khan, F. Hameed, R. Bläsing, S. S. P. Parkin, and J. Castrillon. ShiftsReduce: Minimizing Shifts in Racetrack Memory 4.0. *ACM Transactions on Architecture and Code Optimization*, 16(4), 2019.

[12] K. Kuan and T. Adegbija. MirrorCache: An Energy-Efficient Relaxed Retention L1 STTRAM Cache. In *Proceedings of the Great Lakes Symposium on VLSI*, pages 299–302, 2019.

[13] K. Kuan and T. Adegbija. Energy-Efficient Runtime Adaptable L1 STT-RAM Cache Design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(6):1328–1339, 2020.

[14] N. Kurd, M. Chowdhury, E. Burton, T. P. Thomas, C. Mozak, B. Boswell, P. Mosalikanti, M. Neidengard, A. Deval, A. Khanna, N. Chowdhury, R. Rajwar, T. M. Wilson, and R. Kumar. Haswell: A Family of IA 22 nm Processors. *IEEE Journal of Solid-State Circuits*, 50(1):49–58, 2015.

[15] X. Liang, R. Canal, G. Wei, and D. Brooks. Process Variation Tolerant 3T1D-Based Cache Architectures. In *Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 15–26, 2007.

[16] H. Mao, C. Zhang, G. Sun, and J. Shu. Exploring Data Placement in Racetrack Memory Based Scratchpad Memory. In *Proceedings of the IEEE Non-Volatile Memory System and Applications Symposium*, pages 1–5, 2015.

[17] R. E. Matick and S. E. Schuster. Logic-based eDRAM: Origins and rationale for use. *IBM Journal of Research and Development*, 49(1):145–165, 2005.

[18] S. Mittal, R. Wang, and J. Vetter. DESTINY: A Comprehensive Tool with 3D and Multi-Level Cell Memory Modeling Capability. *Journal of Low Power Electronics and Applications*, 7(3):1–24, 2017.

[19] S. Motaman, A. S. Iyengar, and S. Ghosh. Domain Wall Memory-Layout, Circuit and Synergistic Systems. *IEEE Transactions on Nanotechnology*, 14(2):282–291, 2015.

[20] S. S. Parkin, M. Hayashi, and L. Thomas. Magnetic Domain-Wall Racetrack Memory. *Science*, 320(5873):190–194, 2008.

[21] A. Ranjan, S. G. Ramasubramanian, R. Venkatesan, V. Pai, K. Roy, and A. Raghunathan. DyReCTape: A Dynamically Reconfigurable Cache using Domain Wall Memory Tapes. In *Proceedings of the Design, Automation, and Test in Europe Conference Exhibition*, pages 181–186, 2015.

[22] B. Sinharoy, J. A. Van Norstrand, R. J. Eickemeyer, H. Q. Le, J. Leenstra, D. Q. Nguyen, B. Konigsburg, K. Ward, M. D. Brown, J. E. Moreira, D. Levitan, S. Tung, D. Hrusecky, J. W. Bishop, M. Gschwind, M. Boersma, M. Kroener, M. Kaltenbach, T. Karkhanis, and K. M. Fernsler. IBM POWER8 Processor Core Microarchitecture. *IBM Journal of Research and Development*, 59(1):2:1–2:21, 2015.

[23] C. W. Smullen, V. Mohan, A. Nigam, S. Gurumurthi, and M. R. Stan. Relaxing Non-Volatility for Fast and Energy-Efficient STT-RAM Caches. In *Proceedings of the IEEE 17th International Symposium on High Performance Computer Architecture*, pages 50–61, 2011.

[24] Z. Sun, X. Bi, A. K. Jones, and H. Li. Design Exploration of Racetrack Lower-Level Caches. In *Proceedings of the IEEE/ACM International Symposium on Low Power Electronics and Design*, pages 263–266, 2014.

[25] Z. Sun, X. Bi, H. Li, W. Wong, Z. Ong, X. Zhu, and W. Wu. Multi Retention Level STT-RAM Cache Designs with a Dynamic Refresh Scheme. In *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 329–338, 2011.

[26] Z. Sun, X. Bi, W. Wu, S. Yoo, and H. Li. Array Organization and Data Management Exploration in Racetrack Memory. *IEEE Transactions on Computers*, 65(4):1041–1054, 2016.

[27] S. Thoziyoor, N. Muralimanohar, J. H. Ahn, and N. P. Jouppi. CACTI 5.1. *HP Development Company, Palo Alto, CA, USA. Technical Report HPL-2008-20*, 2008.

[28] R. Ubal, J. Sahuquillo, S. Petit, and P. López. Multi2Sim: A Simulation Framework for CPU-GPU Computing. In *Proceedings of the 19th International Symposium on Computer Architecture and High Performance Computing*, pages 62–68, 2007.

[29] A. Valero, S. Petit, J. Sahuquillo, P. López, and J. Duato. Design, Performance, and Energy Consumption of eDRAM/SRAM Macrocells for L1 Data Caches. *IEEE Transactions on Computers*, 61(9):1231–1242, 2012.

[30] R. Venkatesan, V. J. Kozhikkottu, M. Sharad, C. Augustine, A. Raychowdhury, K. Roy, and A. Raghunathan. Cache Design with Domain Wall Memory. *IEEE Transactions on Computers*, 65(4):1010–1024, 2016.

[31] G. Wang, Y. Zhang, B. Zhang, B. Wu, J. Nan, X. Zhang, Z. Zhang, J. Klein, D. Ravelosona, Z. Wang, Y. Zhang, and W. Zhao. Ultra-Dense Ring-Shaped Racetrack Memory Cache Design. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 66(1):215–225, 2019.

[32] Z. Wang, D. A. Jiménez, C. Xu, G. Sun, and Y. Xie. Adaptive Placement and Migration Policy for an STT-RAM-Based Hybrid Cache. In *Proceedings of the IEEE 20th International Symposium on High Performance Computer Architecture*, pages 13–24, 2014.

[33] W. Xu, H. Sun, X. Wang, Y. Chen, and T. Zhang. Design of Last-Level On-Chip Cache Using Spin-Torque Transfer RAM (STT RAM). *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 19(3):483–493, 2011.

[34] J. Yao, J. Ma, T. Chen, and T. Hu. An Energy-Efficient Scheme for STT-RAM L1 Cache. In *Proceedings of the IEEE 10th International Conference on High Performance Computing and Communications and the IEEE International Conference on Embedded and Ubiquitous Computing*, pages 1345–1350, 2013.

[35] M. Zabihi, Z. I. Chowdhury, Z. Zhao, U. R. Karpuzcu, J. Wang, and S. S. Sapatnekar. In-Memory Processing on the Spintronic CRAM: From Hardware Design to Application Mapping. *IEEE Transactions on Computers*, 68(8):1159–1173, 2019.

[36] J. Zhang, M. Jung, and M. Kandemir. FUSE: Fusing STT-MRAM into GPUs to Alleviate Off-Chip Memory Access Overheads. In *Proceedings of the IEEE International Symposium on High Performance Computer Architecture*, pages 426–439, 2019.

[37] P. Zhou, B. Zhao, J. Yang, and Y. Zhang. Energy Reduction for STT-RAM Using Early Write Termination. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design - Digest of Technical Papers*, pages 264–268, 2009.