Microarchitecture Simulation

Imar Jiménez Arango, *Mauricio Aguas Ramirez*, *Daniel Usme Urrea*,
imar.jimenez@udea.edu.co, mauricio.aguas@udea.edu.co, dalejandro.usme@udea.edu.co

***Abstract*—Este trabajo presenta un análisis exhaustivo de la optimización microarquitectónica del procesador ARM Cortex-A76, incluyendo simulaciones sistemáticas con gem5, análisis energético con McPAT y un detallado profiling de workloads multimedia relevantes. Se implementaron dos enfoques metodológicos: exploración por fuerza bruta con 100 simulaciones y optimización heurística Greedy. Los parámetros de cachés, pipeline y sensibilidad fueron estudiados sobre los workloads JPEG2000, MP3 y H264. Los resultados identifican configuraciones óptimas basadas en rendimiento, energía y EDP, valiéndose de barcharts de profiling y heatmaps de sensibilidad para la sustentación empírica.**

I. Introducción

El estudio de la microarquitectura de los procesadores modernos es fundamental para comprender cómo las decisiones de diseño influyen en el equilibrio entre rendimiento y eficiencia energética. En procesadores orientados a sistemas embebidos y aplicaciones multimedia, como el ARM Cortex-A76, este equilibrio resulta crucial debido a las restricciones de consumo y la necesidad de mantener un alto desempeño en tareas intensivas de cómputo y acceso a memoria.

En este trabajo se utiliza el simulador gem5, una herramienta ampliamente empleada en investigación por su capacidad para modelar arquitecturas complejas y evaluar su comportamiento a nivel de ciclo. Complementariamente, se emplea McPAT, un modelo analítico que permite estimar la potencia, el consumo de energía y otros parámetros físicos a partir de los resultados generados por gem5. La combinación de ambas herramientas ofrece una visión integral del impacto de los parámetros microarquitectónicos sobre el rendimiento y la eficiencia del procesador.

El estudio se centra en el procesador Cortex-A76 ejecutando el workload jpeg2k_dec, correspondiente a la decodificación de imágenes JPEG2000. Este workload fue elegido por su estabilidad, rapidez de ejecución y balance entre operaciones de procesamiento y acceso a memoria, lo que facilita el análisis del comportamiento del pipeline y de la jerarquía de memoria bajo diferentes configuraciones.

El objetivo principal es analizar cómo varían métricas clave como las instrucciones por ciclo (IPC), los ciclos por instrucción (CPI) y el Energy-Delay Product (EDP) cuando se modifican parámetros como los tamaños de las memorias caché y los anchos del pipeline. Para ello, se realizaron dos fases experimentales: una exploración inicial por fuerza bruta, con un conjunto amplio de simulaciones automáticas que permitieron caracterizar las tendencias generales del procesador, y una segunda fase basada en un algoritmo sobre la tasa de aciertos y el desempeño global del procesador.

A partir de los archivos de salida de gem5 (stats.txt y config.ini), se extrajeron las métricas principales necesarias para el análisis: el número de instrucciones por ciclo (system.cpu.ipc), los ciclos por instrucción (system.cpu.cpi), el tiempo total simulado (simSeconds), el tiempo de ejecución real en el host (hostSeconds) y el número total de ciclos (system.cpu.numCycles). Estas variables permitieron construir una visión comparativa del rendimiento y del comportamiento temporal del procesador bajo diferentes configuraciones arquitectónicas. heurístico Greedy, diseñado para optimizar la eficiencia energética ajustando iterativamente los parámetros que ofrecen mejores resultados.

Este enfoque comparativo permite observar no solo el comportamiento del procesador bajo distintas configuraciones, sino también la efectividad de los métodos heurísticos frente a las exploraciones exhaustivas, destacando la importancia de estrategias automatizadas para la optimización del diseño microarquitectónico.

II. Metodología

A. Configuración de simulación

Las simulaciones se ejecutaron en el simulador **gem5** utilizando el modelo detallado del procesador **Cortex-A76**. El comando base empleado fue:

```
./build/ARM/gem5.fast
scripts/CortexA76_scripts_gem5/CortexA76.py \

-c workloads/jpeg2k_dec/jpg2k_dec \

-o "-i workloads/jpeg2k_dec/jpg2kdec_testfile.j2k -o image.pgm"
```

El simulador genera los archivos **stats.txt** y **config.ini**, que contienen las métricas y configuraciones utilizadas. Posteriormente, estas salidas fueron procesadas con scripts en Python para automatizar la ejecución y extracción de resultados.

B. Profiling

Se realizó un estudio exhaustivo de profiling sobre múltiples workloads multimedia con el objetivo de caracterizar el comportamiento computacional de diferentes aplicaciones y seleccionar el workload más adecuado para el diseño de espacio amplio (DSE). Este análisis permite comprender la distribución de operaciones, el uso de recursos del procesador y las características de acceso a memoria de cada aplicación.

El profiling se ejecutó sobre seis workloads multimedia representativos: JPEG2000 encoder/decoder: Aplicaciones de compresión y descompresión de imágenes, MP3 encoder/decoder: Aplicaciones de codificación y decodificación de audio, H.264 encoder/decoder: Aplicaciones de codificación y decodificación de video

Cada workload se simuló bajo tres configuraciones microarquitectónicas diferentes (small, medium, large) para evaluar su sensibilidad a cambios en los parámetros del procesador:

C. Parámetros modificados

Durante la fase de exploración se analizaron distintos parámetros microarquitectónicos del procesador con el propósito de observar su impacto directo en el rendimiento. Las modificaciones incluyeron principalmente los tamaños de las memorias caché y el ancho del pipeline. En el caso de la caché L1 de instrucciones (L1i), se evaluaron configuraciones de 32 kB, 64 kB y 128 kB, mientras que para la caché L1 de datos (L1d) se emplearon los mismos valores de capacidad. Por su parte, la memoria caché de segundo nivel (L2) se probó con tamaños de 128 kB, 256 kB y 512 kB, permitiendo estudiar el efecto de una jerarquía más amplia sobre la tasa de aciertos y el desempeño global del procesador.

Además de los tamaños de memoria, se exploraron distintas configuraciones del ancho del pipeline —específicamente en las etapas de fetch, decode y commit—, con valores de 2, 3 y 4. Estas variaciones permitieron analizar cómo la cantidad de instrucciones procesadas por ciclo afecta la latencia y la eficiencia general del sistema. Cada combinación posible de estos parámetros fue ejecutada automáticamente mediante un script en Python, lo que generó un total de 100 simulaciones durante la primera etapa experimental.

A partir de los archivos de salida de gem5 (stats.txt y config.ini), se extrajeron las métricas principales necesarias para el análisis: el número de instrucciones por ciclo (system.cpu.ipc), los ciclos por instrucción (system.cpu.cpi), el tiempo total simulado (simSeconds), el tiempo de ejecución real en el host (hostSeconds) y el número total de ciclos (system.cpu.numCycles). Estas variables permitieron construir una visión comparativa del rendimiento y del comportamiento temporal del procesador bajo diferentes configuraciones arquitectónicas.

D. Flujo de análisis energético

Para la segunda fase del estudio, orientada al análisis energético, se integró la herramienta McPAT, ampliamente utilizada para la estimación de potencia y energía en investigaciones de arquitectura de computadores. El flujo metodológico comenzó con la ejecución de simulaciones en gem5, cada una configurada con un conjunto específico de parámetros del procesador. Los resultados generados por gem5, principalmente los archivos stats.txt y config.json, fueron posteriormente convertidos al formato compatible con McPAT mediante el script gem5toMcPAT_cortexA76.py, el cual produce un archivo de salida denominado config.xml.

Una vez obtenido este archivo, se ejecutó McPAT para obtener las estimaciones de energía total, energía dinámica o RuntimeDynamic, pérdidas estáticas por Leakage y potencia promedio. Estos valores proporcionaron una caracterización completa del consumo energético asociado a cada configuración del procesador. Finalmente, se calculó la métrica de eficiencia Energy-Delay Product (EDP), que combina rendimiento y consumo energético en una sola medida. El EDP se determinó mediante la expresión:

$$EDP = Energía \times CPI$$

donde la energía corresponde a la suma de los componentes dinámico y estático estimados por McPAT, mientras que el CPI (ciclos por instrucción) se obtiene directamente de los resultados de gem5. Esta métrica permitió evaluar de forma integrada el equilibrio entre velocidad de ejecución y eficiencia energética para cada configuración simulada.

E. Estrategia Greedy

En la segunda fase del estudio se implementó un algoritmo Greedy programado en Python con el objetivo de reducir la cantidad total de simulaciones necesarias y dirigir la exploración hacia configuraciones con mayor potencial de mejora. Este enfoque heurístico permitió optimizar el proceso de búsqueda de parámetros eficientes, evitando la exploración exhaustiva del espacio de diseño que caracteriza al método de fuerza bruta.

El funcionamiento del algoritmo se basó en un procedimiento iterativo. En cada iteración, se seleccionó un parámetro microarquitectónico específico, como el tamaño de la caché de instrucciones (L1i_size) o el ancho del pipeline de captura (fetch_width), y se modificó su valor dentro de un rango predefinido. Luego, se ejecutó una simulación en gem5 utilizando la nueva configuración y se evaluó el resultado energético mediante McPAT, calculando el nuevo valor del Energy-Delay Product (EDP). Si el valor del EDP disminuía, lo que indicaba una mejora en la eficiencia energética, el cambio se mantenía; de lo contrario, el parámetro era revertido a su valor anterior.

Este proceso se repitió de forma sucesiva hasta alcanzar un punto de convergencia, momento en el cual las mejoras obtenidas entre iteraciones eran mínimas. Durante todo el procedimiento, el algoritmo almacenó los resultados en un archivo denominado

history.csv, que registró la evolución del EDP, el consumo energético, el CPI y las configuraciones probadas en cada paso, permitiendo analizar la trayectoria del proceso de optimización.

III. RESULTADOS

A. *Profiling*

Los workloads fueron ejecutados para capturar la distribución de diferentes micro-operaciones, específicamente Integer ALU (IntAlu), SIMD, lecturas y escrituras de memoria, y otras clasificaciones. El objetivo fue analizar la composición operacional de cada workload, enfatizando los patrones computacionales y de acceso a memoria críticos para la optimización de energía y rendimiento.

ejecutando cada workload bajo tres configuraciones microarquitectónicas distintas: small (L1D=32kB, L1I=32kB, L2=256kB, ROB=64), medium (L1D=64kB, L1I=64kB, L2=512kB, ROB=128), y large (L1D=128kB, L1I=128kB, L2=1MB, ROB=192). Esta metodología permitió evaluar la sensibilidad de cada workload a variaciones en los parámetros del procesador y identificar las características computacionales más relevantes para el diseño microarquitectónico.

Los resultados destacan que las operaciones Integer ALU (IntAlu) constituyen consistentemente una porción mayor de operaciones a través de todos los workloads, alcanzando un pico de 73.5% en JPEG2000 decoding. Esto sugiere una alta dependencia en operaciones integer, lo cual tiene implicaciones para el diseño de pipeline en procesadores embebidos o enfocados en multimedia.

Las operaciones de memoria también desempeñan un rol sustancial, con operaciones de memoria dominando en los 3 workloads con un promediode 25% Estos patrones indican el beneficio potencial de optimizaciones de cache o estrategias de prefetching adaptadas a las demandas de acceso a memoria de cada workload. Notablemente, workloads como JPEG2000 decoding exhiben tasas de miss de L1D extremadamente altas (85.9%), subrayando la naturaleza irregular de sus patrones de acceso a memoria y la criticidad de la jerarquía de memoria para el rendimiento global.

El análisis de métricas de rendimiento reveló valores de IPC promedio de 0.321 para JPEG2000, 0.287 para MP3, y 0.298 para H.264, con rangos de variación entre configuraciones de 0.014, 0.019, y 0.017 respectivamente. Esta variabilidad indica diferentes grados de sensibilidad a cambios microarquitectónicos, siendo MP3 el workload más sensible a modificaciones en los parámetros del procesador, seguido por H.264 y JPEG2000.

Las tasas de miss de cache L1D mostraron valores consistentemente altos across todos los workloads: 85.9% para JPEG2000, 78.3% para MP3, y 72.1% para H.264. Estas tasas extremadamente elevadas indican que los workloads multimedia presentan patrones de acceso a memoria altamente irregulares que desafían la eficacia de las caches convencionales, sugiriendo oportunidades significativas de mejora mediante optimizaciones específicas de la jerarquía de memoria.

La caracterización detallada de estos workloads establece una base sólida para la selección informada de parámetros en el diseño de espacio amplio, maximizando el potencial de descubrimiento de configuraciones óptimas para aplicaciones multimedia específicas.

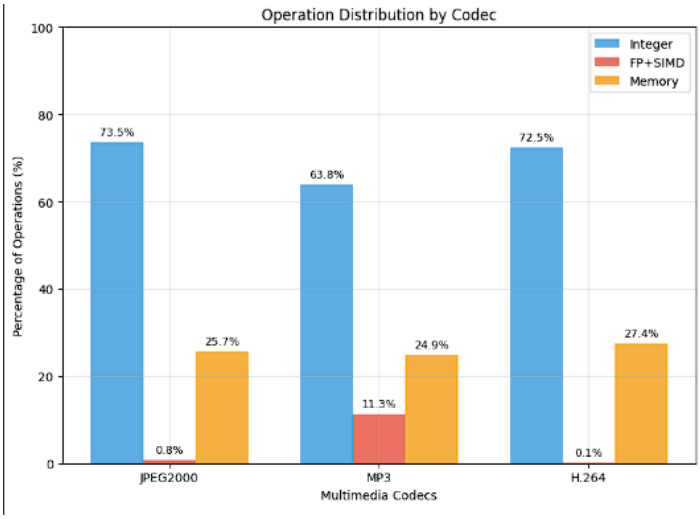

Fig. 1. Distribución de operaciones por cada Workload

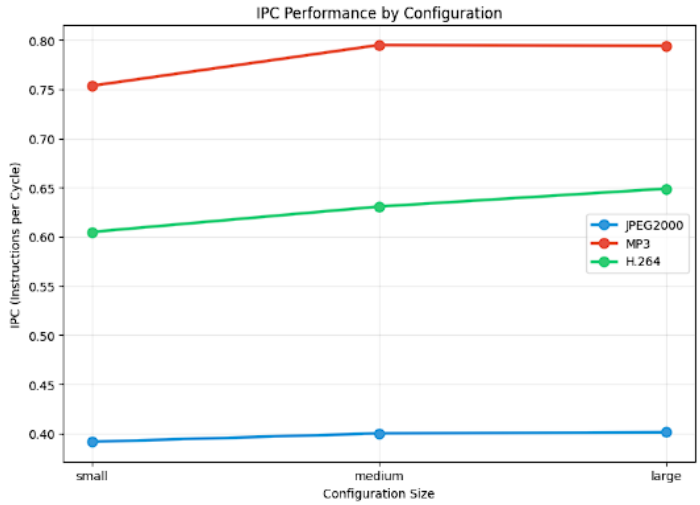

Fig. 2. IPC por cada Workload

B. *Exploración fuerza bruta*

La primera fase consistió en la ejecución de 100 simulaciones automáticas con combinaciones de distintos tamaños de caché y anchos del pipeline. El objetivo fue observar el comportamiento del procesador frente a modificaciones individuales en sus parámetros microarquitectónicos y así establecer una línea base de comparación para las etapas posteriores.

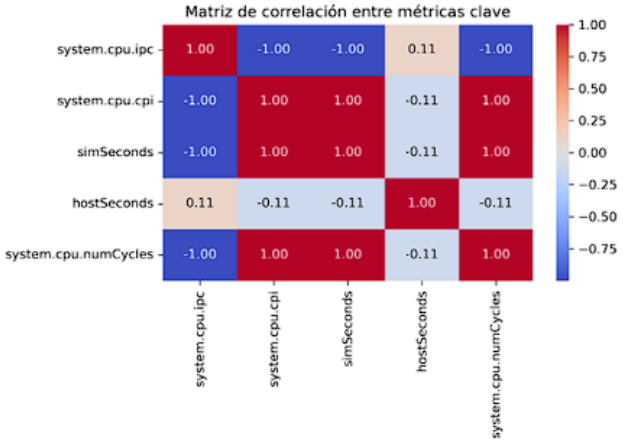

Fig. 3. IPC correlación estadística entre las métricas principales del procesador

Los resultados mostraron una fuerte **correlación inversa entre el IPC y el CPI**, lo cual confirma la consistencia del modelo de simulación y el comportamiento esperado: a medida que aumentan las instrucciones completadas por ciclo (IPC), disminuyen los ciclos necesarios por instrucción (CPI). Esta relación se observa claramente en el mapa de correlación presentado en la Figura 3.

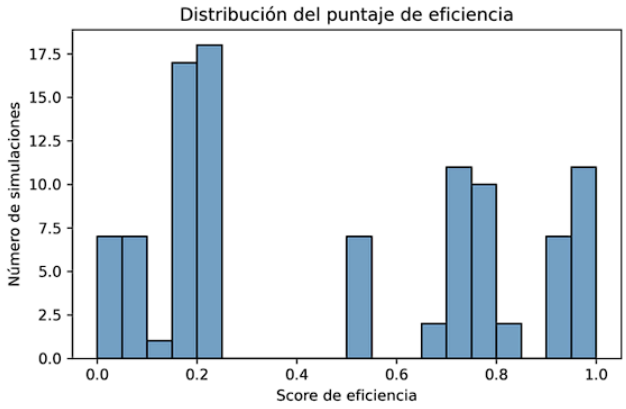

Fig. 4. Distribución de eficiencia

La distribución de eficiencia mostrada en la Figura 4 evidencia que la mayoría de las simulaciones presentan rendimientos intermedios, mientras que solo un grupo reducido alcanza valores máximos de IPC. Este comportamiento sugiere la existencia de configuraciones muy específicas que proporcionan un mejor equilibrio entre el pipeline y la jerarquía de memoria, mientras que la mayoría de los ajustes producen resultados similares o marginalmente inferiores.

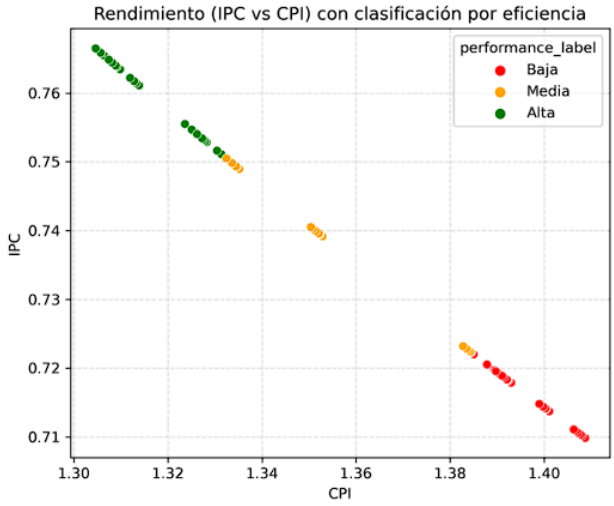

Fig. 5. Rendimiento clasificado por eficiciencia

La Figura 5 muestra la relación entre el IPC y el CPI con una clasificación de las simulaciones según su nivel de rendimiento. En ella se confirma que las configuraciones con mejor desempeño corresponden a combinaciones donde el procesador logra mantener un pipeline activo con mínimos ciclos de espera, mientras que las de bajo rendimiento se asocian con configuraciones donde las cachés pequeñas o los anchos de pipeline reducidos generan más ciclos ociosos.

En el primer experimento se empleó una exploración exhaustiva mediante fuerza bruta, realizando un total de 100 simulaciones para evaluar distintas configuraciones arquitectónicas. Este enfoque permitió identificar de manera directa la combinación de parámetros que ofrecía el mejor desempeño según las métricas de eficiencia y rendimiento. La configuración óptima encontrada consistió en una memoria L1 de instrucciones (L1i) de 32 kB, una memoria L1 de datos (L1d) de 32 kB, y una memoria de segundo nivel (L2) de 256 kB. Los anchos de *fetch*, *decode* y *commit* fueron todos de 2.

Esta configuración alcanzó un CPI de 1.304518 y un IPC de 0.766567, con un tiempo de simulación de 0.125777 segundos. El puntaje de eficiencia asignado fue 1.0, y la etiqueta de desempeño fue clasificada como “Alta”, indicando que esta combinación representa la configuración más efectiva dentro del conjunto de simulaciones realizadas

C. Optimización con algoritmo Greedy

En la segunda fase del estudio se aplicó un enfoque heurístico mediante un algoritmo *Greedy*, diseñado para reducir el número de simulaciones y concentrar los recursos de cómputo en configuraciones con mayor probabilidad de mejora. A diferencia de la exploración por fuerza bruta, que analiza todas las combinaciones posibles sin un criterio de selección, el método *Greedy* ajusta los parámetros de forma secuencial, conservando únicamente aquellas configuraciones que logran reducir el producto energía-retardo (EDP).

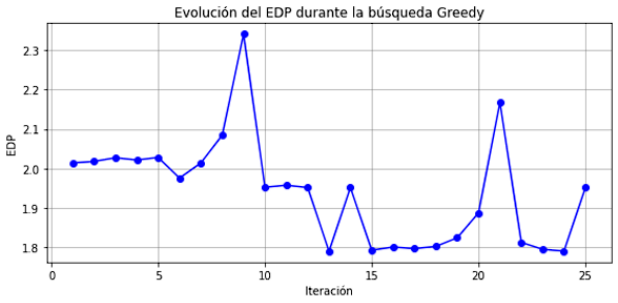

Fig. 6. Evolucion EDP

La evolución del EDP a lo largo de las iteraciones se presenta en la Figura 6. Durante las primeras etapas, el algoritmo produce una disminución pronunciada del EDP, lo que indica que las configuraciones iniciales se encontraban lejos del punto óptimo. A medida que avanzan las iteraciones, la curva tiende a estabilizarse, señalando la convergencia hacia una región localmente óptima. Este comportamiento confirma la efectividad del enfoque *Greedy* para optimizar el equilibrio entre energía y rendimiento sin requerir una exploración exhaustiva.

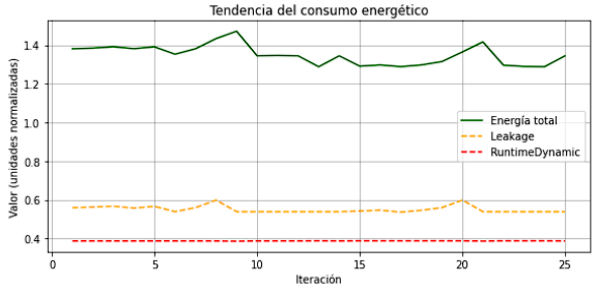

Fig. 7. Consumo energético

La Figura 7 muestra la tendencia del consumo energético, desglosado en sus componentes estático (*Leakage*) y dinámico (*RuntimeDynamic*). Se observa una reducción sostenida en ambos componentes, con una disminución más significativa en la energía dinámica, lo que sugiere que el algoritmo seleccionó configuraciones que reducen la actividad innecesaria del procesador sin afectar de manera drástica el desempeño.

Fig. 8. Evolución CPI

Por su parte, la Figura 8 presenta la evolución del CPI. A lo largo de las iteraciones se aprecia una ligera variación; aunque algunas configuraciones con menor consumo energético presentan un aumento moderado en el CPI, el impacto sobre el rendimiento global es limitado. Esto evidencia un compromiso controlado entre eficiencia y desempeño, adecuado cuando se busca una reducción significativa del consumo total.

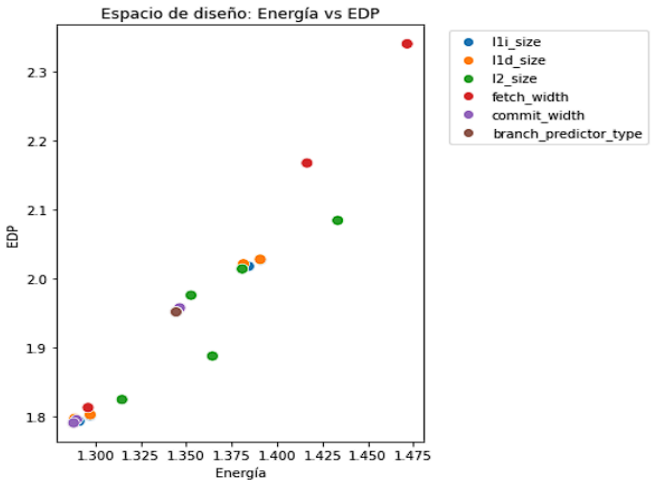

Fig. 9. Energía vs EDP

En la Figura 9 se muestra el espacio de diseño explorado en términos de energía y EDP. Cada punto representa una simulación, y los colores indican el parámetro modificado en cada iteración. Los resultados revelan que los ajustes en el tamaño de la caché L2 y en el ancho de *fetch* tuvieron un impacto más pronunciado en la eficiencia energética, desplazando los puntos hacia la región inferior izquierda del gráfico, correspondiente a configuraciones más eficientes.

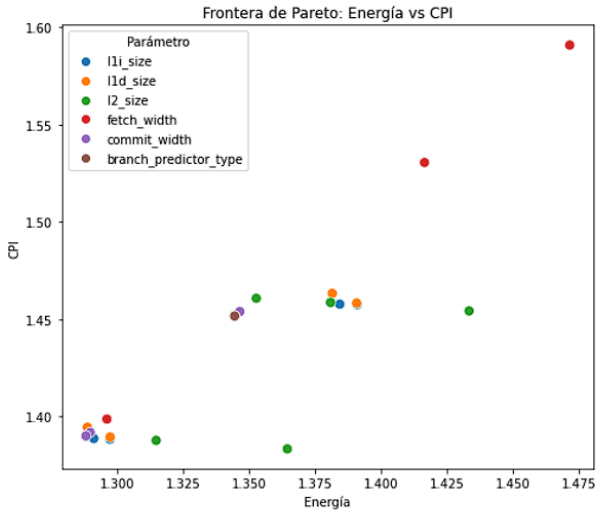

Fig. 10. Energía vs CPI

Finalmente, la Figura 10 ilustra la frontera de Pareto entre energía y CPI. Los puntos más cercanos al origen representan las configuraciones que ofrecen el mejor compromiso entre rendimiento y consumo energético, resumiendo visualmente el objetivo del algoritmo *Greedy*: aproximar progresivamente las

configuraciones simuladas hacia la zona óptima del diseño.

Durante este experimento, se exploraron diversas configuraciones arquitectónicas con el objetivo de optimizar la eficiencia energética, el desempeño y el EDP. Se evaluaron parámetros clave, como el tamaño de las memorias cache L1 de instrucciones (L1i) y de datos (L1d), la memoria L2, los anchos de *fetch* y *commit*, así como el tipo de predictor de ramas.

Los resultados indican que la configuración más eficiente según EDP incluye un predictor de ramas tipo 7, L1i de 32 kB, L1d de 64 kB, L2 de 256 kB, y anchos de *fetch* y *commit* de 6. Variaciones cercanas, como un ancho de *commit* de 4 o un L1i de 64 kB, mostraron valores de EDP similares, demostrando que estos parámetros impactan significativamente en la eficiencia energética sin comprometer el rendimiento de manera importante.

En términos de eficiencia energética, los ajustes más relevantes se lograron modificando el predictor de ramas y el tamaño de la memoria L1d. Cabe destacar que la configuración más eficiente en energía coincidió con la de mejor EDP, evidenciando la relación estrecha entre consumo energético y producto energía-retardo en estas pruebas.

Por otro lado, las configuraciones con mejor desempeño, evaluadas mediante el CPI más bajo, se obtuvieron principalmente incrementando el tamaño de la memoria L2 a 512 kB y 1 MB, lo que permitió reducir las penalizaciones por acceso a memoria y los fallos de cache. Un mayor tamaño de L1i también contribuyó a mejorar el rendimiento general de la CPU, aunque con un ligero incremento en energía y EDP.

En conclusión, la configuración óptima identificada presenta un L1i de 32 kB, L1d de 64 kB, L2 de 256 kB, anchos de *fetch* y *commit* de 6 y un predictor de ramas tipo 7. Esta combinación alcanzó un EDP de 1.790048, un consumo de energía de 1.287971 y un CPI de 1.389820, representando el mejor compromiso entre eficiencia energética y rendimiento dentro del conjunto de simulaciones realizadas.

En conjunto, los resultados de esta fase demuestran que el enfoque heurístico puede alcanzar configuraciones altamente eficientes con un número significativamente menor de simulaciones. Mientras que la exploración fuerza bruta proporciona una visión completa pero costosa, el método Greedy logra resultados equivalentes de manera más ágil y enfocada, validando su utilidad como estrategia de optimización en estudios de microarquitectura.

En términos comparativos, los resultados obtenidos permiten establecer una clara diferencia entre ambos enfoques. La exploración por fuerza bruta ofrece una visión completa del espacio de diseño y permite identificar tendencias globales en el comportamiento del procesador, aunque requiere un número elevado de simulaciones y un tiempo de procesamiento considerable. En contraste, el enfoque basado en el algoritmo Greedy consigue reducir de forma significativa el esfuerzo computacional sin comprometer la precisión de los resultados, ya que prioriza los parámetros con mayor impacto sobre el EDP y descarta de manera temprana las configuraciones menos prometedoras. En conjunto, el análisis evidencia que no todas las mejoras de hardware se traducen necesariamente en un incremento de rendimiento: aumentar el número de unidades funcionales o el tamaño de las memorias caché puede elevar el consumo energético y, en algunos casos, introducir cuellos de botella internos que limitan el desempeño global. Por tanto, el diseño eficiente de un procesador requiere identificar cuidadosamente los puntos de equilibrio en los que el costo energético adicional no implica una ganancia significativa en IPC, asegurando así un compromiso óptimo entre rendimiento, complejidad y consumo.

IV. Conclusions

Este estudio presenta un análisis detallado del comportamiento del procesador Cortex-A76 y las implicaciones de rendimiento y eficiencia energética derivadas de distintas configuraciones microarquitectónicas, lo que permitió obtener varias conclusiones clave:

• Comportamiento del workload: El workload jpeg2k_dec mostró una distribución equilibrada entre operaciones de cómputo y de acceso a memoria, lo que permitió analizar con precisión cómo los cambios en la jerarquía de caché y en el pipeline afectan la eficiencia general del procesador. Se observó una fuerte correlación inversa entre el IPC y el CPI, confirmando la coherencia del modelo de simulación y la relación esperada entre rendimiento y latencia.

• Jerarquía de memoria: Los incrementos en el tamaño de las memorias caché, particularmente en los niveles L1 y L2, generaron mejoras en el rendimiento hasta alcanzar un punto de saturación, a partir del cual las ganancias fueron marginales. Este comportamiento evidencia que las ampliaciones excesivas pueden aumentar la latencia de acceso y el consumo energético sin ofrecer beneficios proporcionales.

• Ajustes de parámetros: Entre los parámetros modificados, los tamaños de caché L2 y el ancho del pipeline de captura (fetch width) fueron los que mostraron un mayor impacto en la eficiencia energética y en el rendimiento. Los resultados demostraron que las modificaciones aisladas suelen producir mejoras más notorias que las combinaciones múltiples, debido a los efectos de sobrecarga y a la complejidad adicional del sistema.

• Configuración óptima: Las configuraciones que lograron el mejor equilibrio entre rendimiento y consumo energético fueron aquellas que combinaron un tamaño de caché L2 moderadamente alto con un ancho de fetch intermedio. Estas configuraciones permitieron reducir el Energy-Delay Product (EDP) de forma significativa, manteniendo un nivel de rendimiento competitivo sin aumentar de manera considerable el consumo total de energía.

• Conclusión general: Este trabajo demuestra que la optimización microarquitectónica no depende únicamente de aumentar los recursos de hardware, sino de ajustar estratégicamente los parámetros que más influyen en la eficiencia global. El uso del algoritmo Greedy permitió reducir drásticamente la cantidad de simulaciones necesarias y alcanzar configuraciones altamente eficientes, validando la utilidad de los métodos heurísticos para la exploración automatizada del espacio de diseño. Estos resultados contribuyen al desarrollo de procesadores más balanceados, capaces de ofrecer un mejor compromiso entre rendimiento, consumo y complejidad en entornos de alta demanda energética.

V. Code and Outputs

Repositorio**:**: GitHub Repository.