Laboratorio 1: Simulación de microarquitectura

Manuel Santiago Velásquez López
Departamento de Ing. Electrónica y Telecomunicaciones
Universidad de Antioquia
Medellín, Colombia
msantiago.velasquez”udea.edu.co

Yonathan López Mejía
Departamento de Ing. Electrónica y Telecomunicaciones
Universidad de Antioquia
Medellín, Colombia
harley.lopez@udea.edu.co

I. Introducción

En este trabajo se busca analizar cómo los distintos parámetros microarquitectónicos de un procesador Cortex A76 afectan tanto su rendimiento como su consumo energético. Para ello se utilizó el simulador *Gem5*, una herramienta ampliamente usada en la investigación de microarquitecturas de procesadores. Con dicho simulador se realizó un perfilamento inicial de 6 flujos de trabajo que se consideran relativamente pesados como son decodificadores y codificadores de h264, jpg y mp3.

A partir de este perfil se construyó un espacio de diseño que considera diferentes parámetros del procesador, los cuales se modificaron sistemáticamente para evaluar su impacto en el rendimiento mediante *Gem5*, y en el consumo energético mediante la herramienta *McPAT*.

La exploración del espacio de diseño se realizó mediante un *script* que implementa el algoritmo de búsqueda metaheurística conocido como *Recocido Simulado*. Este algoritmo, inspirado en el proceso físico de enfriamiento de los metales, permite realizar una búsqueda de soluciones cercanas a la solución óptima en problemas que por su complejidad tomarían mucho tiempo en solucionarse de manera exhaustiva.

Dicho algoritmo se aplicó sobre el flujo de trabajo que presentó un mayor EDP pues dicha métrica permite evaluar un compromiso entre rendimiento y eficiencia energética. Lográndose una configuración que mejora en alrededor del 30% el EDP de la configuración inicial del procesador de prueba.

II. Objetivos

- Analizar distintos flujos de trabajo para identificar posibles cuellos de botella y oportunidades de mejora en la microarquitectura del procesador.
- Definir un espacio de diseño en el que se modifiquen parámetros microarquitecturales con el propósito de evaluar su impacto sobre el rendimiento y la eficiencia energética a lo largo de un flujo de trabajo.
- Desarrollar *scripts* que automaticen el proceso de simulación e implementen una metaheurística capaz de reducir el tiempo de búsqueda y aproximarse a soluciones cercanas al óptimo sin requerir exploraciones exhaustivas.
- Integrar las herramientas *Gem5* y *McPAT* dentro del flujo de trabajo para perfilar el consumo energético de las configuraciones exploradas y seleccionar aquellas que optimicen las métricas de rendimiento, eficiencia energética y compromiso entre ambas (EDP).

III. Marco teórico

Gem5

Gem5 es un simulador de sistemas y microarquitecturas de procesadores ampliamente utilizado en el ámbito de la investigación para evaluar el rendimiento y el comportamiento energético de distintas configuraciones de hardware. Ofrece una plataforma flexible que permite simular diversas arquitecturas de CPU, subsistemas de memoria y dispositivos de entrada/salida (I/O).

Una de sus principales ventajas es la combinación de simulación a nivel de ciclo y a nivel funcional, lo que posibilita capturar tanto el desempeño general del sistema como el comportamiento detallado de cada componente.

McPAT

McPAT (*Multicore Power, Area, and Timing*) es una herramienta de modelado analítico que estima el consumo de potencia, el área física y el tiempo de respuesta de procesadores multinúcleo. Emplea modelos parametrizados basados en características arquitectónicas —como el tamaño de las cachés, el ancho de bus o la frecuencia de reloj— para ofrecer una estimación detallada del consumo energético y del área de los distintos bloques del procesador.

Integración entre Gem5 y McPAT

El flujo de trabajo comienza con **Gem5**, que genera para cada simulación dos archivos principales:

- `stats.txt`, con las estadísticas de ejecución.
- `config.json`, con los parámetros arquitectónicos del sistema simulado.

A continuación, estos archivos se procesan mediante el script de Python `gem5toMcPAT_cortexA76.py` (proporcionado por el docente), el cual produce un archivo de salida `config.xml` compatible con **McPAT**.

Finalmente, McPAT se ejecuta indicando la ubicación del archivo `.xml` generado para cada simulación. Como resultado, la herramienta genera un informe detallado que incluye métricas de **potencia estática**, **potencia dinámica** y **área** de los componentes del procesador.

IV. Procedimiento

Instruction Class profiling

Inicialmente se evalúan los 6 *workloads* proporcionados y se extraen las siguientes métricas de su respectivo *stats.txt*:

- **Número total de operaciones simuladas o *simOps***: Esta métrica incluye el número de instrucciones simuladas junto con las microoperaciones que cada instrucción podría necesitar.
- **Porcentaje de operaciones ejecutadas del tipo IntAlu, Memwrite, Memread, Float y SIMD**: esta información está disponible en las métricas de la forma `system.cpu.commit.committedInstType_0::`, de esta manera se obtiene el porcentaje de operaciones de cada tipo respecto al total.
- **Número de veces que una unidad de procesamiento para operaciones IntALU, Memwrite, Memread, Float y SIMD estuvo ocupada mientras requería ser utilizada**: esta información está contenida en las métricas de la forma `system.cpu.statFuBusy::`

Con esta información es posible construir gráficos de barras que pueden indicar la presencia de cuellos de botella como por ejemplo: si el programa contiene un 50% de instrucciones tipo IntALU y el 40% de las veces que trata de utilizar dicha unidad esta se encuentra ocupada, es posible que exista un cuello de botella.

Parameter definitions and simulation

Una vez se tiene un perfil de cada una de los workloads se procede a escoger uno con un numero reducido de variedad de instrucciones, en este caso el dec jpg2k. Esto se hace con el fin de ver mas directamente el cambio generado por la varianza de parámetros pre-escogidos. Respecto a los parámetros a evaluar, se escogen aquellos mas relevantes respecto a las operaciones de memoria, así como el parámetro relacionado a las unidades de calculo de enteros.

Definido esto se procede a escoger un tipo de algoritmo para recorrer el espacio de búsqueda. Simulaciones individuales previas permiten estimar el tiempo de simulación del espacio de muestra y determinar inviable una búsqueda de fuerza bruta debido a limitaciones de computo. Se hace uso de un algoritmo de búsqueda de recocido simulado, permitiendo exploración del espacio de diseño con un costo computacional reducido, siendo mucho menor que el tiempo total estimado con el método de búsqueda a fuerza bruta.

Los resultados de la búsqueda del espacio de diseño da resultados concordantes a lo esperado el perfil del workload escogido. Se ve un incremento irregular de las unidades de escritura respecto a las de lectura, ademas, se ve un incremento hasta el máximo posible en el espacio de las unidades de trabajo de la ALU entera, las unidades de escritura, así como de la localidad de la memoria cache l1. El tamaño de la cache el no crece hasta el máximo posible, tampoco llega al máximo posible del espacio de búsqueda las unidades de lectura de memoria.

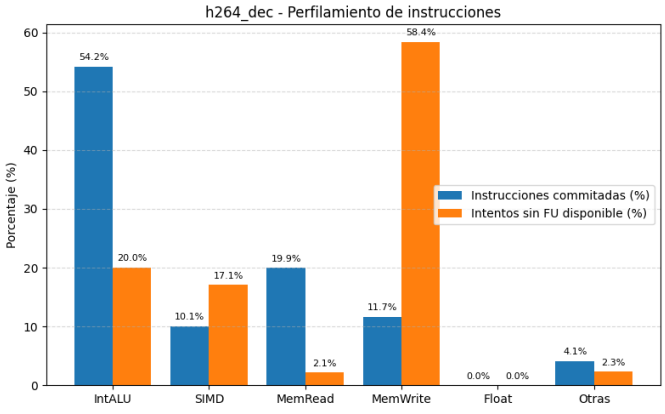

Fig. 1. Instruction class profiling para el decodificador h264

TABLE I
Métricas de rendimiento para los diferentes *workloads*

Workload	Instruction Count	CPI	CPU Time (s)	EDP
`h264_dec`	261,123,526	0.952687	0.118519	2.82469
`h264_enc`	314,280,893	1.007403	0.150775	2.66462
`jpg2k_dec`	201,827,086	1.459392	0.140710	7.05635
`jpg2k_enc`	321,263,107	1.111123	0.170106	4.41075
`mp3_dec`	230,472,915	0.905211	0.099317	2.13930
`mp3_enc`	238,353,312	1.045466	0.118678	2.36974

Respecto al manejo del algoritmo se destaca un espacio de búsqueda multivariado no linear con incrementos respecto a la métrica CPU time la cual se escoge como función objetivo. El algoritmo no hace uso de recalentamiento pero puede ser modificado por el usuario para recalentar al utilizar los últimos parámetros escogidos por el algoritmo la ultima vez que se utilizo. Para el manejo de las rondas se toman 10 por cada análisis y se generan un máximo de entre 14 y 16 vecinos para el calculo del siguiente paso. Estos cálculos de exploración se realizan en paralelo.

Para el calculo de consumo energético y de desempeño se hace uso de la herramienta McPAT. Los resultados de la anterior simulación se evalúan respecto a su desempeño respecto a consumo energético y se comparan unas con otras hasta encontrar tres valores considerados como importantes. El mínimo EDP, el mínimo CPU time, y un tradeoff que se encuentra como el producto del CPU time y el EDP.

V. Resultados

Instruction Class Profiling

Las figuras 1, 2, 3 4 5 6 muestran la manera en que se distribuyen las instrucciones y el porcentaje de tiempo que sus unidades funcionales estuvieron ocupadas en cada simulación. Por otro lado, la tabla muestra un perfilamiento adicional de cada uno de los *workloads* que además agrega el cálculo del EDP usando McPAT

Recocido simulado

Luego de definir el espacio de diseño, se corrió el recocido simulado

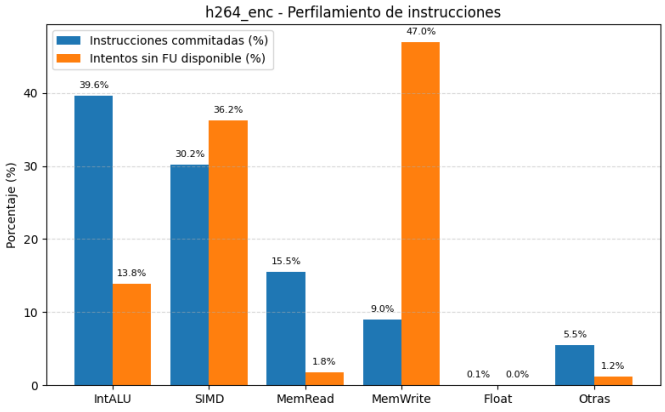

Fig. 2. Instruction class profiling para el codificador h264

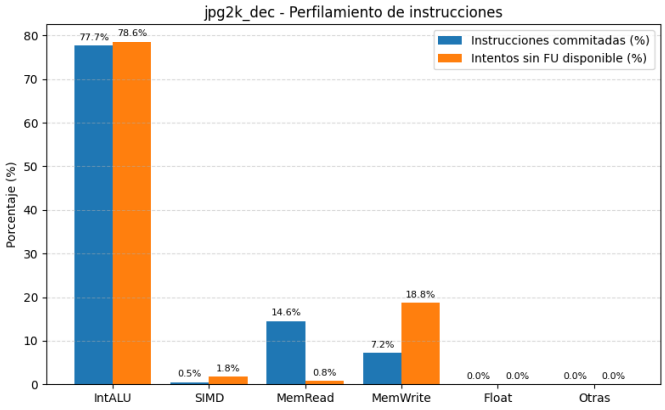

Fig. 3. Instruction class profiling para el decodificador jpg

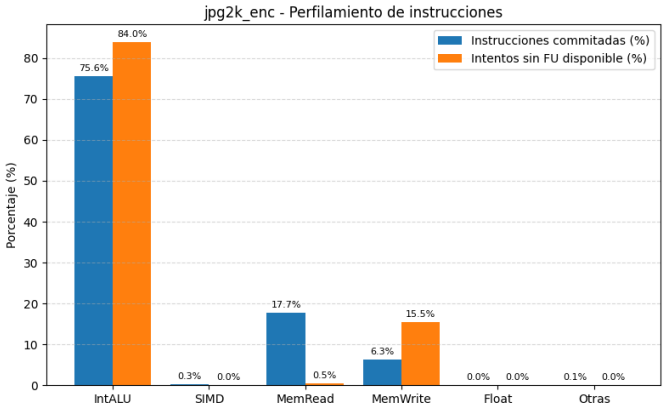

Fig. 4. Instruction class profiling para el codificador jpg

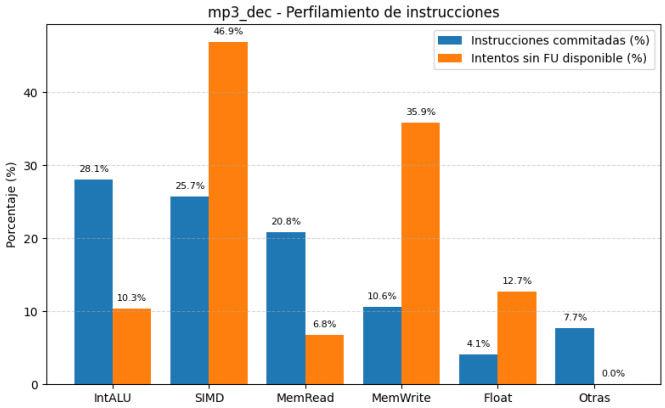

Fig. 5. Instruction class profiling para el decodificador mp3

TABLE II
COMPARACIÓN DE CONFIGURACIONES CON MEJOR EDP, TIEMPO DE CPU Y COMPROMISO EDP–TIEMPO

Caso	ALU	R	W	L1D (kB)	A	CPU Time (s)	EDP
Mejor EDP	6	5	7	64	2	0.140642	4.7995
Mejor CPU time	3	3	8	512	2	0.140139	4.9450
Mejor compromiso (EDP × CPU time)	6	3	3	64	2	0.140640	4.7996

Por otro lado, la figura muestra cómo se comporta el EDP y el CPU time en las distintas configuraciones encontradas con el recocido simulado.

VI. ANÁLISIS DE RESULTADOS

Workloads iniciales

En general, los *workloads* son intensivos en el uso de operaciones enteras, así como en accesos de lectura y escritura a memoria. Sin embargo, existen casos particulares, como el del codificador H.264, donde en el 58% de las ocasiones en que se intentó escribir en memoria esta se encontraba ocupada. Inicialmente esto podría parecer un cuello de botella; no obstante, dichas operaciones representan únicamente el 12% del total, por lo que su impacto absoluto podría no resultar relevante si se compara, por ejemplo con el 20% de las veces que la unidad de arimética entera estaba ocupada si se tiene en cuenta que dichas operaciones representan un 54% del total aproximadamente.

Por otro lado, en el codificador MP3, las operaciones de lectura de memoria ocurrieron en el 41.7% de los casos y, en el 40% de ellos, la unidad de escritura estaba ocupada, lo que sí podría representar un posible cuello de botella.

Aún así, todos los programas presentan variaciones en la distribución de instrucciones que utilizan haciendo que cualquier mejora en la microarquitectura del procesador no impacte de igual manera a todos los programas. Esta es una de las limitaciones de los procesadores de propósito general. Sin embargo, cualquier cambio debería buscar mejorar la mayor cantidad de parámetros posibles.

La Tabla I presenta las métricas de rendimiento obtenidas para seis workloads representativos de codificación y decodificación multimedia. Se muestran el número total de instrucciones ejecutadas, el CPI promedio, el tiempo total de CPU y el producto energía–retardo (EDP), utilizado como métrica de eficiencia energética.

En términos generales, los encoders presentan un mayor conteo de instrucciones que sus correspondientes decoders. Por ejemplo, h264_enc ejecuta aproximadamente un 20% más de instrucciones que h264_dec, y jpg2k_enc más de un

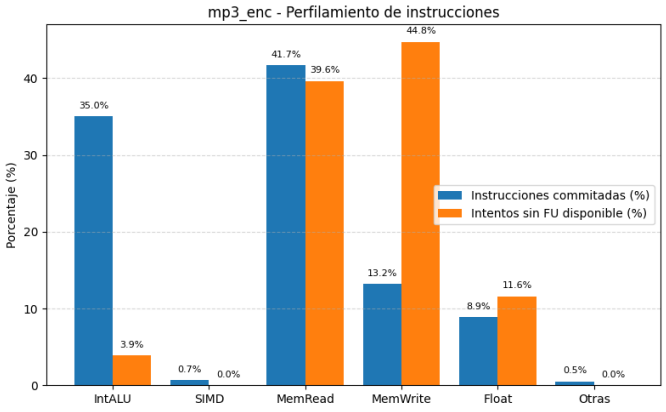

Fig. 6. Instruction class profiling para el codificador mp3

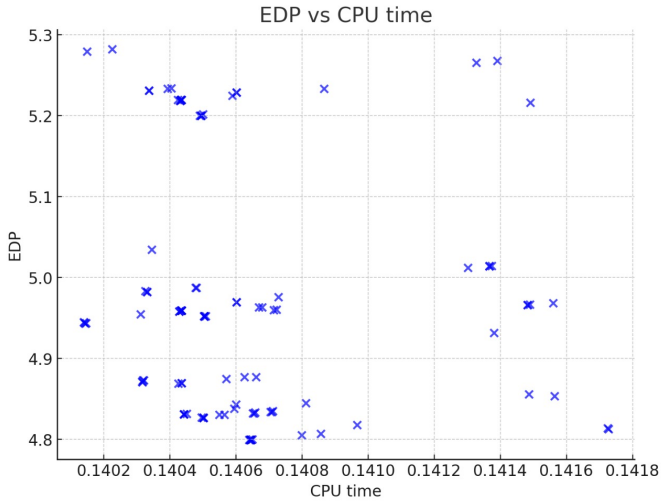

Fig. 7. Instruction class profiling para el codificador mp3

50% respecto a jpg2k_dec. Esto puede dar indicios de que las tareas de compresión son más complejas pues requieren operaciones y transformaciones sobre los datos que pueden no ser computacionalmente baratas.

En cuanto al CPI, los valores se mantienen próximos 1 (entre 0.9 y 1.46), lo muestra un aprovechamiento aceptable del procesador. Sin embargo, jpg2k_dec tiene el CPI más elevado (1.46), lo que indicaría una ejecución menos eficiente posiblemente por una mayor dependencia de memoria o menor paralelismo en las instrucciones.

El tiempo de CPU se comporta similar a las instrucciones: los encoders tienden a requerir más tiempo de ejecución que los decoders. Aún así, jpg2k_dec es una excepción pues muestra un tiempo relativamente alto pese a un menor número de instrucciones, lo cual es coherente con un CPI alto.

Finalmente, el EDP permite evaluar el equilibrio entre rendimiento y eficiencia. Los decoders de mp3 y h264 presentan los valores más bajos (2.1–2.8), con lo cual presentan una mejor eficiencia energética relativa. Por otra parte, jpg2k_dec alcanza el EDP más alto de todos (7.05), lo que la identifica como la carga más costosa en términos de energía y tiempo combinados. Siendo esta la carga que se seleccionó para realizar el recocido simulado

Recocido simulado

Como se mencionó anteriormente, se seleccionó el decodificador `jpg2k_dec` debido a su elevado valor de EDP. En la Tabla II se observa que todas las configuraciones obtenidas mediante recocido simulado reducen el EDP en aproximadamente un 30%. Esta mejora se logra principalmente al incrementar el número de unidades de cómputo y de acceso a memoria, lo que permite una mayor concurrencia en la ejecución de instrucciones que involucran operaciones aritméticas (ALU) y de escritura en memoria.

El aumento en el paralelismo reduce el CPI y, en consecuencia, el tiempo total de ejecución. Aunque estas configuraciones pueden incrementar la potencia disipada debido al mayor número de unidades activas, el impacto positivo sobre el rendimiento es más importante, resultando en una mejora general del EDP. Dado que el valor inicial del EDP era considerablemente alto, las reducciones en CPI tuvieron un efecto más significativo en el comportamiento global de la métrica.

Sin embargo, en un escenario real valdría la pena preguntarse si aumentar dichas unidades de cómputo vale la pena en un dispositivo de propósito general como el Cortex A76 teniendo en cuenta que es posible que un usuario no necesite dicho rendimiento extra en tareas de decodificación de imágenes jpg.

Por otro lado, la Figura 7 ilustra cómo las configuraciones exploradas pueden presentar métricas de rendimiento similares, pero diferir significativamente en su EDP o viceversa. Esto evidencia que mejoras en el desempeño no siempre implican una mayor eficiencia energética, y que el equilibrio entre ambas métricas depende directamente de los recursos microarquitectónicos empleados.

En la región inferior izquierda del gráfico se observan las configuraciones más favorables, que combinan bajos valores de EDP con tiempos de ejecución reducidos, representando el mejor compromiso entre rendimiento y eficiencia. En contraste, las configuraciones ubicadas en la parte superior izquierda presentan un alto rendimiento a costa de un EDP elevado, mientras que las situadas en la parte inferior derecha muestran un EDP bajo pero con un desempeño limitado. Finalmente, en la parte superior derecha se identifican tres configuraciones subóptimas, ya que presentan simultáneamente un mayor EDP y un menor rendimiento, resultando malas en ambos aspectos.

VII. Conclusiones

- La realización de este tipo de simulaciones representa un desafío significativo, ya que requiere una gran cantidad de tiempo y recursos computacionales. En este contexto, se evidencia la importancia de ejecutar múltiples simulaciones en paralelo y de emplear algoritmos metaheurísticos que, aunque no garantizan alcanzar la solución

óptima, permiten obtener resultados cercanos a ella con un costo computacional considerablemente menor.

- Las modificaciones microarquitectónicas implementadas fueron diseñadas específicamente para el flujo de trabajo definido en este estudio; por tanto, el impacto observado sobre el EDP no debe extrapolarse directamente a otros escenarios. El análisis inicial muestra que, si bien los distintos *workloads* comparten ciertas similitudes en su comportamiento de instrucciones, presentan diferencias suficientes para responder de manera distinta a las mismas modificaciones. Adicionalmente, otros parámetros no considerados en este trabajo —como los predictores de salto o las memorias caché de niveles inferiores— podrían incluirse en análisis futuros para lograr una caracterización más completa del comportamiento microarquitectónico.
- No existieron cambios significativos en el `CPU Time` como si los hubo en el EDP, aún así, esta última es una métrica más interesante porque integra el consumo de potencia y los ciclos de reloj por instrucción, permitiendo una visión más global de la relación entre rendimiento y consumo energético.