Laboratorio 1
Simulación de microarquitectura

Jose Alejandro Olivo Petit
Ingeniería Electrónica
Universidad de Antioquia
Medellin, Colombia
Email: jose.olivop@udea.edu.co

Juan Diego Cabrera Moncada
Ingeniería Electrónica
Universidad de Antioquia
Medellin, Colombia
Email: juan.cabrera@udea.edu.co

***Abstract*—Este trabajo presenta el análisis de la gestión de memoria y el rendimiento en programas en C++ utilizando el conjunto de herramientas Valgrind, específicamente Callgrind y Massif. Callgrind permitió generar perfiles de llamadas e identificar las funciones más costosas computacionalmente, mientras que Massif ofreció una visión detallada del uso de memoria en heap y pila durante la ejecución. Los resultados muestran el impacto de una adecuada gestión de memoria dinámica: los programas sin liberación de memoria presentan un crecimiento continuo del heap, mientras que aquellos con liberación correcta de recursos exhiben un consumo de memoria estable y predecible. Estos hallazgos resaltan la importancia de integrar herramientas de perfilado en el ciclo de desarrollo de software para garantizar eficiencia, robustez y sostenibilidad en aplicaciones críticas.**

I. Introducción

El diseño y evaluación de microarquitecturas de procesadores modernos requiere comprender cómo las decisiones en la organización interna afectan el rendimiento, el consumo energético y el aprovechamiento de recursos. En este laboratorio se utilizó el simulador gem5 para modelar un procesador ARM Cortex-A76 y estudiar el efecto de modificar distintos parámetros microarquitectónicos, tales como el tamaño y la latencia de los niveles de caché, el ancho del pipeline, la cantidad de unidades funcionales, el tamaño del Reorder Buffer (ROB) y diferentes estrategias de predicción de saltos. Además, se complementó el análisis con el uso de McPAT para estimar métricas energéticas, permitiendo evaluar la eficiencia de las configuraciones obtenidas.

Con el objetivo de explorar casos representativos dentro del espacio de diseño, se realizaron tres etapas de análisis: (i) un estudio exploratorio para determinar la sensibilidad del desempeño frente a modificaciones individuales; (ii) una búsqueda exhaustiva por rejilla (grid search) en los parámetros más influyentes; y (iii) un proceso de optimización mediante Simulated Annealing para encontrar configuraciones con un equilibrio entre rendimiento y consumo energético. Finalmente, se construyó una gráfica de energía vs. desempeño para comparar las configuraciones evaluadas y discutir los compromisos inherentes (trade-offs) entre reducir tiempo de ejecución y limitar el gasto energético.

II. Instruction-class profilling

Se realizó el *instruction-class profiling* para dos *workloads*: `jpeg2k_dec` (Figura 1) y `h264_dec` (Figura 2). Se observa que ambos comparten ciertas características, como el alto predominio de instrucciones enteras y el uso casi nulo de instrucciones vectoriales o de punto flotante. Sin embargo, existen diferencias en el comportamiento de acceso a memoria, especialmente en la proporción entre instrucciones de carga (*load*) y almacenamiento (*store*). Debido a limitaciones de tiempo y a que el análisis del primer *workload* se inició con anterioridad, la mayoría de los estudios microarquitectónicos se realizaron sobre `jpeg2k_dec`.

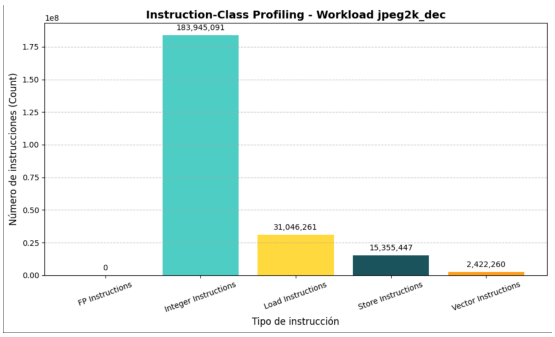

Fig. 1. Distribución de clases de instrucciones para el `workload jpeg2k_dec`.

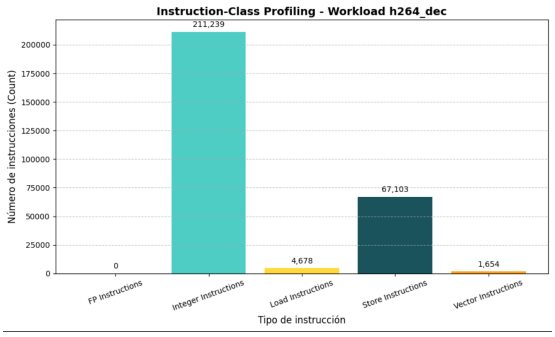

Fig. 2. Distribución de clases de instrucciones para el `workload h264_dec`.

III. Primer análisis

A partir del análisis del archivo `stats.txt`, se evaluaron diversas métricas de desempeño, entre ellas: *IPC*, *CPI*, *SimSeconds* y *BranchMispreds* orientadas a mejorar el desempeño del procesador. Se definieron varios *parámetros de diseño* a modificar, incluyendo el predictor de saltos, las memorias caché de primer, segundo y tercer nivel (*L1 instruction cache*, *L1 data cache*, *L2 cache* y *L3 cache*), el número de entradas del *BTB*, la cantidad de unidades aritmético-lógicas (*ALU units*), el ancho del *pipeline* y los tamaños de las colas internas (*queue sizes*).

El procedimiento experimental consistió en realizar una búsqueda en rejilla (*grid search*), variando inicialmente el tipo de predictor de saltos y, posteriormente, cada parámetro individualmente, manteniendo en cada iteración el valor óptimo obtenido en la variación anterior. Sin embargo, el proceso no se comportó de manera completamente correcta, ya que a partir de cierto punto algunos valores no se actualizaron adecuadamente con las configuraciones óptimas previas. A pesar de ello, fue posible observar la tendencia de las métricas de desempeño frente a las modificaciones de los parámetros de diseño, y también se evaluaron algunas configuraciones predefinidas como referencia.

TABLE I
RESULTADOS DE DESEMPEÑO AL VARIAR DIFERENTES PARÁMETROS MICROARQUITECTÓNICOS.

Parámetro	Configuración	SimSec	CPI	IPC
ALU Units	alu_4	0.141229	1.4648	0.6827
	alu_3	0.141233	1.4648	0.6827
	alu_2	0.141381	1.4664	0.6820
	alu_1	0.142379	1.4767	0.6772
Pipeline Width	pipe_8	0.140178	1.4539	0.6878
	pipe_6	0.140294	1.4551	0.6872
	pipe_mixed_narrow	0.140656	1.4588	0.6855
	pipe_4	0.141381	1.4664	0.6820
	pipe_mixed_wide	0.141388	1.4664	0.6819
	pipe_2	0.154309	1.6004	0.6248
L1 Instruction Cache	l1i_256kB	0.141353	1.4661	0.6821
	l1i_128kB	0.141362	1.4662	0.6821
	l1i_64kB	0.141381	1.4664	0.6820
	l1i_32kB	0.141493	1.4675	0.6814
ROB / LSQ Size	lsq_large	0.141362	1.4662	0.6821
	rob_256	0.141363	1.4662	0.6821
	rob_64	0.141421	1.4668	0.6818
	lsq_small	0.141555	1.4682	0.6811
L1 Data Cache	l1d_256kB	0.141218	1.4647	0.6828
	l1d_128kB	0.141326	1.4658	0.6822
	l1d_64kB	0.141381	1.4664	0.6820
	l1d_32kB	0.141857	1.4713	0.6797
Optimized Configurations	opt_low_latency	0.112983	1.1718	0.8534
	opt_ultimate	0.138805	1.4396	0.6946
	opt_memory_hierarchy	0.139679	1.4487	0.6903
	opt_aggressive	0.139900	1.4510	0.6892
	opt_wide_pipeline	0.140646	1.4587	0.6855
	opt_balanced	0.141021	1.4626	0.6837
	opt_best_cache	0.141155	1.4640	0.6831
	opt_execution_units	0.141170	1.4642	0.6830
	opt_branch_hardware	0.141450	1.4671	0.6816
	opt_efficiency	0.154162	1.5989	0.6254
L2 Cache	l2_1MB	0.140784	1.4602	0.6849
	l2_512kB	0.141212	1.4646	0.6828
	l2_256kB	0.141381	1.4664	0.6820
	l2_128kB	0.141465	1.4672	0.6816
L3 Cache	l3_4MB	0.140311	1.4553	0.6872
	l3_2MB	0.141381	1.4664	0.6820
	l3_1MB	0.141839	1.4711	0.6798
BTB Entries	btb_8192	0.141239	1.4649	0.6826
	btb_4096	0.141381	1.4664	0.6820
	btb_16384	0.141450	1.4671	0.6816
	btb_2048	0.141491	1.4675	0.6814
	btb_32768	0.141521	1.4678	0.6813
	btb_1024	0.143045	1.4836	0.6740

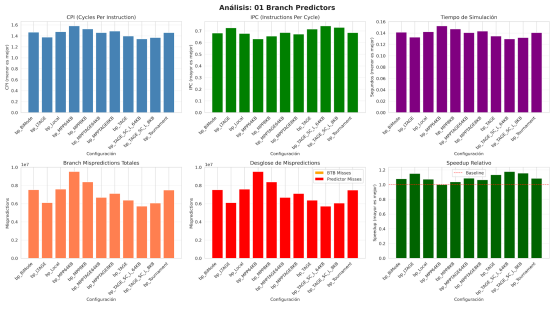

Fig. 3. Comparación de predictores de salto considerando `CPI`, `IPC`, tiempo de simulación, tasa total de `branch misprediction`, desglose de fallos y *speedup* relativo.

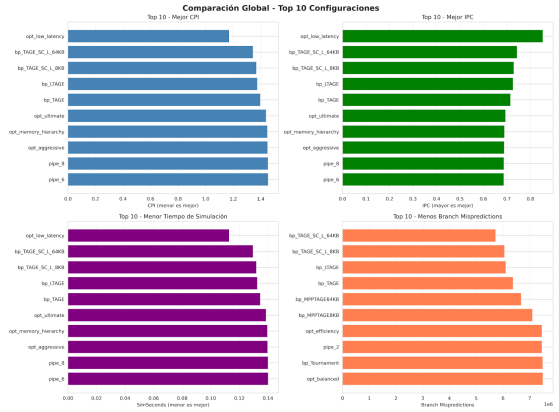

Fig. 4. Comparación de las 10 mejores configuraciones en términos de `CPI`, `IPC`, tiempo de simulación y tasa de `branch misprediction`.

IV. Segundo análisis

En el segundo análisis, a partir de lo hecho en el analisis anterior, se realizó un grid search con los parámetros para los cuales se consideró que la arquitectura era más propensa a verse afectada en términos de desempeño.

En primer lugar, se evaluaron distintos tipos avanzados de predictores de salto, específicamente LTAGE, TAGE_SC_L_64KB y TAGE_SC_L_8KB. Estos predictores se escogieron debido a su capacidad para disminuir la tasa de predicciones erróneas, lo que reduce la cantidad de burbujas en el pipeline y, en consecuencia, mejora el IPC. Además, se modificó el número de entradas del BTB (Branch Target Buffer), incrementando su tamaño para permitir almacenar un mayor historial de saltos y minimizar fallos en bifurcaciones indirectas.

En el subsistema de memoria se variaron los tamaños de las cachés L1 de instrucciones y datos (32 kB, 64 kB y 128 kB), la caché L2 (128 kB hasta 1 MB) y la caché L3 compartida (1 MB hasta 4 MB). El objetivo de este barrido fue reducir la latencia asociada a los accesos a memoria, disminuyendo así la proporción de ciclos en los que el procesador permanece inactivo esperando datos. Un menor número de fallos en caché impacta directamente en la reducción del CPI, mejorando también el tiempo total de ejecución.

Se exploraron también parámetros que influyen en el paralelismo y la capacidad de procesamiento del pipeline. Entre ellos, el número de unidades funcionales ALU enteras se incrementó para permitir ejecutar más operaciones en paralelo. Asimismo, se variaron las entradas del Reorder Buffer (ROB), lo que permitió mantener más instrucciones en vuelo y reducir la frecuencia con la que el procesador se ve obligado a detenerse por dependencias. Finalmente, se modificó el ancho del pipeline (fetch, decode, issue y commit), probando configuraciones de 4, 6 y 8 instrucciones por ciclo. Un mayor ancho de pipeline incrementa el número máximo de instrucciones procesadas por ciclo, aumentando el IPC cuando el flujo de instrucciones y los recursos internos lo permiten.

Omitiendo el top 1 mostrado en las gráficas dadas a continuación, pues se trató de una ejecución defectuosa del algoritmo, se logra observar que, a comparación de los resultados obtenidos en el primer análisis, en general, el tiempo de ejecución se logró reducir en cierta medida. Excepto para el caso de la implementación 'opt_low_latency' del primer análisis, para el cual se alcanzó un tiempo de ejecución incluso menor, con 0.113 segundos aproximadamente.

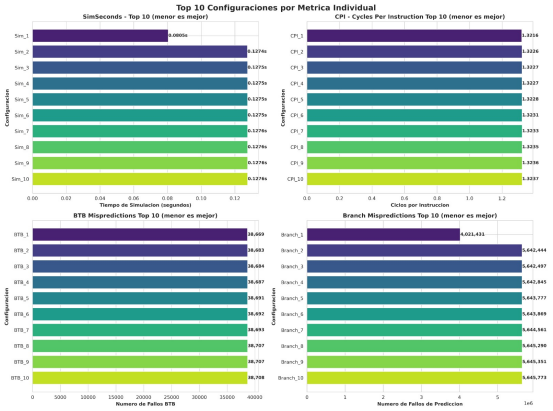

Fig. 5. Gráficas de análisis de resultados con la ejecución de un algoritmo de Gridsearch con selección de parámetros.

V. Tercer análisis

Para este análisis se empleó el algoritmo de *Simulated Annealing* considerando todas las variables de diseño. Este método, inspirado en el proceso físico de recocido de metales, se utilizó como estrategia de optimización para la búsqueda de configuraciones microarquitectónicas eficientes. Su principal ventaja radica en su capacidad de explorar el espacio de diseño de manera estocástica, aceptando ocasionalmente soluciones de menor desempeño con el fin de evitar mínimos locales y favorecer la exploración global. Durante las iteraciones iniciales, la "temperatura" del sistema es elevada, lo que incrementa la probabilidad de aceptar configuraciones subóptimas; a medida que la temperatura disminuye, el algoritmo se vuelve más selectivo y converge hacia una solución estable.

Como punto de partida se utilizó la configuración predeterminada de la microarquitectura, ejecutando el `workload h264_dec`, seleccionado por su similitud con `jpeg2k_dec` y por presentar un tiempo de ejecución considerablemente menor. Posteriormente, con el resultado obtenido de esta primera ejecución del algoritmo, se realiza una segunda ejecución del mismo ahora con el workload de `jpeg2k_dec`.

La temperatura inicial se determinó a partir de los experimentos del primer análisis, calculando primero la desviación estándar de las diferencias en la función objetivo y luego aplicando la fórmula de desviación de aceptación $k\sigma$, donde σ representa la desviación estándar y $k = -3/\ln(p)$. Se utilizó un esquema de enfriamiento geométrico, elegido por su eficiencia computacional, con un rango estandarizado para el parámetro α.

Así, la solución final obtenida en la primera ejecución del algoritmo se empleó como punto inicial para una segunda iteración. En esta segunda fase, la temperatura inicial se

TABLE II
TOP 10 CONFIGURACIONES DEL SEGUNDO ANÁLISIS CON PARÁMETROS RELEVANTES Y MÉTRICAS DE DESEMPEÑO.

SimID / Config	L1I	L1D	L2	L3	BTB	#ALUs	ROB	Ancho Pipeline	SimSec	CPI	IPC
699 (Rank 1)	32kB	32kB	512kB	4MB	8192	3	192	8	**0.080517**	1.42119	0.703636
696 (Rank 2)	32kB	32kB	512kB	4MB	8192	3	128	8	0.127420	1.32155	0.756687
480 (Rank 3)	32kB	32kB	256kB	4MB	16384	3	128	8	0.127522	1.32261	0.756080
486 (Rank 4)	32kB	32kB	256kB	4MB	16384	3	256	8	0.127527	1.32267	0.756048
483 (Rank 5)	32kB	32kB	256kB	4MB	16384	3	192	8	0.127534	1.32273	0.756010
687 (Rank 6)	32kB	32kB	512kB	4MB	8192	2	128	8	0.127543	1.32283	0.755954
690 (Rank 7)	32kB	32kB	512kB	4MB	8192	2	192	8	0.127573	1.32314	0.755779
693 (Rank 8)	32kB	32kB	512kB	4MB	8192	2	256	8	0.127587	1.32329	0.755691
456 (Rank 9)	32kB	32kB	256kB	4MB	8192	3	192	8	0.127611	1.32354	0.755549
459 (Rank 10)	32kB	32kB	256kB	4MB	8192	3	256	8	0.127616	1.32358	0.755524

calculó con la misma fórmula anterior, pero multiplicada por un factor de 100, dado que la probabilidad de aceptación de soluciones con un mayor valor de función objetivo se define según la distribución de Boltzmann:

$$P(\Delta f) = e^{-\frac{\Delta f}{T}}$$

donde Δf representa la diferencia entre los valores de la función objetivo y T la temperatura actual del sistema. Este ajuste permitió ampliar la exploración global del espacio de búsqueda durante las etapas iniciales del proceso de optimización y mejorar la probabilidad de convergencia hacia configuraciones más eficientes. Esto permite desarrollar una arquitectura de procesador que no sólo tenga la posibilidad de resultar en mejoras para un sólo tipo de workload y, en últimas, tender a mejoras generalizadas del procesador independientes del workload.

Cabe destacar que la función objetivo a minimizar correspondió a la suma de (EDP + Energía * 1.5 + SimSeconds * K_1) * K_2 donde K_1 corresponde a una constante que modifica la relevancia del tiempo de ejecución dentro de la función objetivo, con la finalidad de balancear adecuadamente esta métrica a las otras dos de acuerdo a la carga de trabajo seleccionada (h264 ó jpeg2k). Mientras que K_2 corresponde a una constante utilizada para un reajuste de la magnitud de los valores de la función, de tal modo que sean aptos para la implementación del algoritmo metaheurístico de Simulated Annealing, específicamente, la aplicación de la fórmula de probabilidad de aceptación explicada previamente, la cual brinda valores de probabilidad muy reducidos si se define un valor pequeño para la temperatura inicial o si la magnitud de la diferencia entre las funciones objetivo actual y de la solución vecina es reducida.

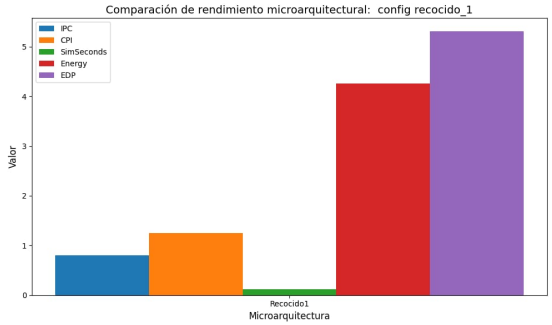

Fig. 6. Gráfica de análisis de la primera ejecución del algoritmo Simulated Annealing.

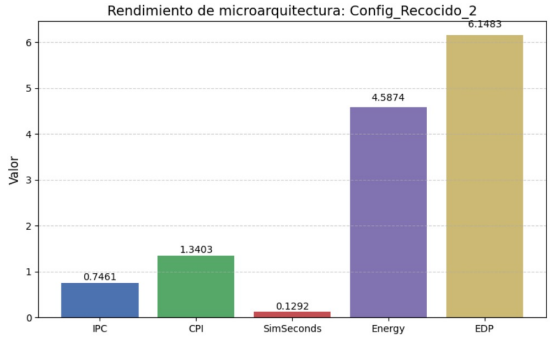

Fig. 7. Gráfica de análisis de la segunda ejecución del algoritmo Simulated Annealing.

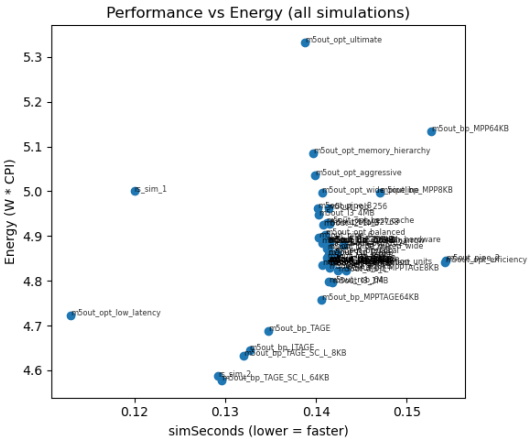

Fig. 8. Gráfica de energía vs. desempeño.

La configuración con latencias reducidas en las memorias caché mostró el menor tiempo de ejecución y menor consumo energético en la gráfica, pero este resultado no es físicamente realista. En una microarquitectura real, la latencia de acceso a una memoria caché está fuertemente condicionada por su tamaño físico, su asociatividad, entre otros factores físicos que comprometen la modificación de este parámetro. Por ende, reducir la latencia sin modificar la estructura subyacente es un supuesto simplificado del simulador y, en ocasiones, no representa una implementación viable en hardware.

VI. Resumen de resultados de desempeño vs. Recursos

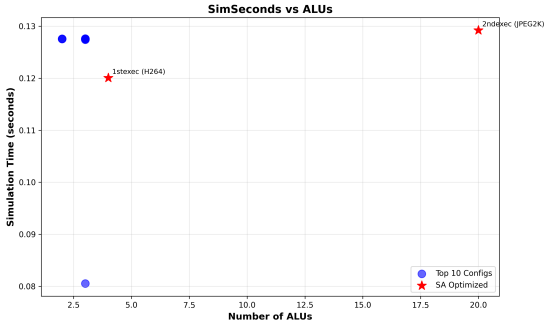

Fig. 9. Gráfica de tiempo de ejecución vs Número de Integer Functional ALU Units.

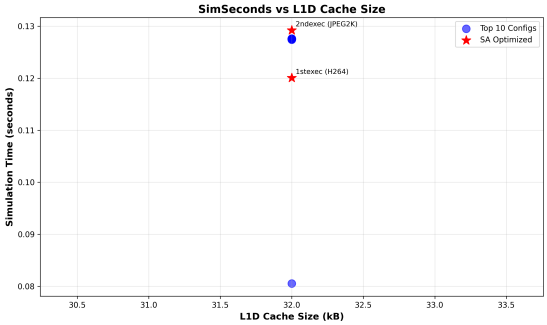

Fig. 10. Gráfica de tiempo de ejecución vs. L1Data.

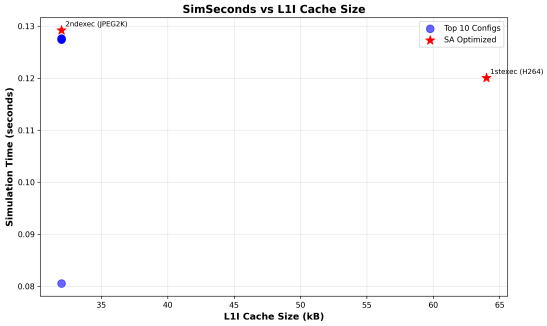

Fig. 11. Gráfica de tiempo de ejecución vs L1Instructions.

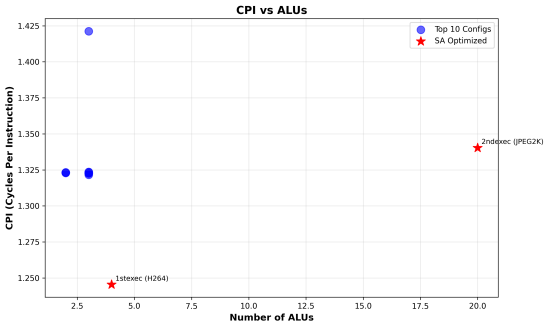

Fig. 12. Gráfica de CPI vs Número de Integer Functional ALU Units.

El análisis de las métricas de rendimiento en función de los recursos arquitectónicos revela patrones significativos que contradicen la intuición de que mayores recursos garantizan mejor desempeño. En las gráficas de SimSeconds vs ALUs y CPI vs ALUs, se observa que la configuración optimizada mediante Simulated Annealing para JPEG2K, con 20 unidades ALU, presenta un tiempo de simulación de aproximadamente 0.135 segundos y un CPI de 1.35, mientras que la configuración para H264, utilizando únicamente 4 ALUs, alcanza 0.12 segundos con un CPI superior de 1.25. Este comportamiento evidencia la presencia de rendimientos decrecientes y posibles cuellos de botella en otros componentes del pipeline, sugiriendo que el exceso de ALUs en JPEG2K no puede ser aprovechado eficientemente debido a limitaciones en la capacidad de fetch, decode o en el tamaño del reorder buffer

(ROB=72). Adicionalmente, las configuraciones del Top 10 demuestran consistencia al concentrarse en el rango de 2 a 3 ALUs con tiempos de simulación similares, excepto por una configuración destacada con 3 ALUs que logra 0.08 segundos que puede llegar a ser un fallo en la ejecución de la simulación, lo cual indica que el balance de la arquitectura supera ampliamente la maximización individual de recursos.

De acuerdo a los resultados obtenidos, se tiene entonces que la mejor configuración en términos de desempeño y EDP corresponde a la optimización low latency implementada en el primer análisis y el mejor en términos de energía fue la solución resultante de la primera ejecución del algoritmo Simulated Annealing.

Conclusiones

En el primer análisis se identificaron los componentes del procesador con mayor sensibilidad en el rendimiento. Los resultados mostraron que el subsistema de memoria (en particular el tamaño y latencia de las cachés L1 y L2) y el predictor de saltos son determinantes para reducir ciclos de espera y mejorar el flujo de instrucciones en el pipeline. También se observó que aumentar la cantidad de unidades funcionales y el ancho del pipeline puede elevar el IPC, aunque con beneficios decrecientes a partir de cierto punto.

Los resultados obtenidos evidencian que la eficiencia del procesador no depende únicamente de incrementar la cantidad de recursos, sino de la coherencia entre ellos y del tipo de *workload* ejecutado. La variación de parámetros mostró que los beneficios en desempeño aparecen cuando existe un equilibrio entre capacidad de ejecución (ALUs, ancho de pipeline), capacidad de reordenamiento (ROB) y tiempo de acceso a la memoria. Además, los predictores de saltos avanzados demostraron ser fundamentales para sostener un flujo continuo de instrucciones, reduciendo penalizaciones por bifurcaciones. La optimización mediante Simulated Annealing confirmó que la arquitectura presenta regiones de rendimiento estable, donde pequeñas modificaciones no generan mejoras significativas, destacando la naturaleza multifactorial y dependiente del contexto en el diseño microarquitectónico.

Finalmente, el uso del método de Simulated Annealing en el tercer análisis demostró ser una estrategia efectiva para navegar espacios de diseño amplios, permitiendo escapar de mínimos locales y encontrar configuraciones balanceadas entre desempeño y consumo energético. Este enfoque evidenció que no existe una única configuración óptima, sino un conjunto de soluciones que representan distintos compromisos entre velocidad, eficiencia y costo energético.

VII. Enlaces importantes

A continuación se adjunta el enlace al repositorio de GitHub donde se encuentra más información sobre las simulaciones ejecutadas, además de gráficas desarrolladas que dan mayor detalle a los resutados de simulación de las configuraciones de procesador seleccionadas: https://github.com/GenCabMon/CortexA76_processorAnalysis.git

También disponibles en formato de Google Drive: https://drive.google.com/drive/folders/1xjSiZRN0ZRJP6e_NKKsIUfmKiNGpF88g?usp=sharing

Bibliografía

References

[1] A. Tanenbaum, Structured Computer Organization, Prentice- Hall International, 4th Edition, 1999.

[2] E.-G. Talbi, Metaheuristics: From Design to Implementation, Wiley Sons, Hoboken, New Jersey, 2009.

[3] Gem5: Gem5 documentation. (2025, 11 de octubre). gem5: The gem5 simulator system. https://www.gem5.org/documentation/

[4] Michel Dubois, Murali Annavaram, Per Stenstrom, Parallel Computer Organization and Design, Cambridge University Press, 2012.