ENINVIE 2005
Encuentro de Investigación en Ingeniería Eléctrica
Zacatecas, Zac, Marzo 17 —18, 2005

Estimación del Consumo de Potencia Dinámica en un Microprocesador Superscalar

Marcos de Alba Rosano
Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Estado de México
Departamento de Ciencias Computacionales
CP-52926.
TEL: +(55)58645555, ext. 2466, correo-e: marcos.de.alba@itesm.mx

Iván Cabrera Altamirano
Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Estado de México
Departamento de Ciencias Computacionales
CP-52926.
TEL: +(55)58645555, ext. 3205, correo-e: a00455591@itesm.mx

Carlos Tadeo Ortega Otero,
Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Estado de México
Departamento de Ciencias Computacionales
CP-52926.
TEL: +(55)58645555, ext. 3205, correo-e: a00464167@itesm.mx

Abstract — The development of a microprocessor is bound by different design tradeoffs. One of them is to specify the type of applications to be executed by the microprocessor. Another is to maximize the instruction throughput per clock cycle. Another more is to optimize energy consumption. In this work we estimate the consumed dynamic power on a superscalar microprocessor.

We evaluated SPECint2000 and SPECfp2000 benchmarks to estimate disipated dynamic power. It was observed that different applications produced different average consumed dynamic power during execution-stage cycles. However, it was also observed that for different applications the average dynamic power for the entire execution of the program remained constant. This result is correlated to the processor characteristics. For the confirguration chosen in our experiments all processor components are connected to a global shared system clock. For this reason, at every clock cycle every component consumes energy.

Keywords — Dynamic power, perfornance, throughput, global clock.

I. INTRODUCCIÓN

EL DESARROLLO de nuevas tecnologías de semiconductores, de modernas técnicas de diseño de circuitos y de novedosas arquitecturas de microprocesadores ha permitido que la ley de Moore [1] se cumpla desde finales de la década de los años 60. Aún más, se han logrado alcances tecnológicos que mejoran dicha ley desde finales de la década de los 80. Debido a que los microprocesadores actuales operan a frecuencias muy altas (por ejemplo, el Pentium 4 opera a una frecuencia de 3.6 GHz) el seguir manteniendo un desarrollo tecnológico al ritmo impuesto por la ley de Moore se ha vuelto un reto cada día más difícil para los diseñadores de microprocesadores.
Por otro lado, la potencia dinámica consumida se expresa por la ecuación [2]:

$$P = CV^2_{dd}\alpha f$$

donde P es la potencia dinámica, C es la capacitancia de la carga, V_{dd} el voltaje de alimentación, α un factor de actividad entre 0 y 1 y f la frecuencia de operación. Se observa que la potencia dinámica es directamente proporcional a la frecuencia, por lo que para cada nueva generación de microprocesadores el consumo se duplicaría considerando que el producto $CV^2_{dd}\alpha$ se mantuviera constante. Debido a que los consumidores de procesadores y las nuevas legislaciones exigen reducir el consumo de procesadores, los diseñadores deben reducir la potencia dinámica por medio de diversas técnicas. De forma inmediata se puede proponer reducir la magnitud del voltaje de operación V_{dd} en un 30 %, manteniendo constantes el valor de la capacitancia de la carga y el factor de actividad. Sin embargo, a algunos componentes dentro del chip del microprocesador no se les puede reducir el voltaje de alimentación debido a limitaciones físicas en tecnologías recientes, al costo y al mantenimiento de la estabilidad de los mismos (por ejemplo, si se desea reducir el voltaje de alimentación de las celdas de memoria, éstas podrían perder el dato almacenado si no se rediseñan nuevas celdas). El proceso de diseño y fabricación de nuevas celdas de memoria no lleva el mismo ritmo que el de otros componentes del microprocesador. Aunque podría proponerse la utilización de celdas de memoria con un menor número de transistores por bit, su rapidez sería penalizada. Por lo tanto, el voltaje de alimentación no puede ser la única fuente para reducir el consumo dinámico. Otras ténicas incluyen el diseño de circuitos dinámicos (cuyo funcionamiento es activado o desactivado en tiempo de ejecución) en etapas del microprocesador que operan sólo en ciertos periodos predeterminados en la ejecución de instrucciones. Otros mecanismos consisten en proponer nuevas arquitecturas cuya filosofía fundamental consiste en “apagar dinámicamente y por completo” secciones específicas del microprocesador [3]. Técnicas adicionales incluyen apagar dinámicamente el sistema de reloj [4], el reducir de forma dinámica la magnitud del voltaje de alimentación [5], el construir microprocesadores cuyos componentes operan a diversas frecuencias [6] y otras más proponen reducir el consumo dinámico a partir de optimización de los códigos a ejecutar por medio de técnicas de compilación [7].

En este trabajo se analiza la potencia dinámica en un microprocesador superescalar con el objeto de identificar regiones del mismo que deben ser mejoradas para reducir la potencia dinámica disipada.

II. TRABAJO PREVIO

El consumo de potencia dinámica es un factor muy importante que se toma en cuenta en el diseño de un microprocesador. Existen diversas técnicas para reducir dicho consumo.

La técnica de conmutación determinística del reloj propuesta en [4] consiste en predecir el grado de utilización de diversos bloques del microprocesador (por ejemplo, *drivers* de las líneas de palabra en una memoria cache de datos, o *latches* que unen etapas del *pipeline*, entre otros) y de

esta forma saber por adelantado durante que periodos puede deshabilitarse la señal de reloj que se les suministra, reduciendo el consumo de potencia dinámica en hasta un 20 % sin tener penalización en el rendimiento del microprocesador.

El escalado dinámico del voltaje de alimentación en redes de interconección del microprocesador es una técnica utilizada para reducir el consumo de potencia dinámica en un alto porcentaje % [5]. Esta técnica consiste en reducir de forma dinámica la amplitud del voltaje de alimentación cuando se detecta que puede bajarse la frecuencia de operación en ciertos bloques del procesador. A través de la detección dinámica del grado de utilización de los diversos bloques del procesador se puede determinar cuales pueden operar a una menor frecuencia. Además de reducirse el consumo de potencia dinámica por operar a menor frecuencia, se reduce el consumo porque de forma dinámica se alimenta un voltaje de alimentación menor a los bloques del procesador que reducen su velocidad de operación sin mantener su estabilidad.

La técnica de conmutación del voltaje de alimentación [8] consiste en apagar de forma dinámica regiones de la memoria cache de datos que no son utilizadas. Por medio de métodos de predicción se puede determinar durante que periodos ciertas regiones no serán utilizadas. También de forma predeterminada se puede decidir apagar ciertas regiones durante un número de ciclos de operación. Este tipo de técnicas proporciona reducción del consumo de la potencia dinámica y estática en hasta un 62 % con un impacto mínimo en rendimiento.

La utilización de múltiples relojes dentro de un microprocesador [6] consiste en identificar cual es la frecuencia de operación óptima para cada bloque del procesador. Debido a que dentro de un microprocesador hay etapas productoras de resultados y otras consumidoras de los mismos es necesario tener colas donde se guardan los resultados. El acceso a y porcentaje de ocupación de esas colas es dependiente de las características de los programas, por ello la frecuencia de operación de estas es más lenta que la de otros bloques del procesador, como las unidades de ejecución, de esta forma el consumo de potencia dinámica se reduce grandemente cuando se utilizan diversos relojes dentro del mismo microprocesador. El proceso de sincronización de los mismos es otro factor de diseño.

Durante el periodo de ejecución de un programa los diversos niveles de memoria cache son utilizados. Cuando se accesa a una línea, ésta se usa frecuentemente por un período de tiempo que se conoce como su tiempo de vida y después deja de usarse por otro período de tiempo que se conoce como su tiempo muerto antes de ser desechada. Detectando de forma dinámica los periodos muertos de las líneas de la memoria cache, estas se pueden apagar por adelantado reduciendo hasta cuatro veces el consumo de potencia estática [9,10].

Otra técnica más agresiva para reducir el consumo de potencia dinámica consiste en habilitar y deshabilitar bloques completos del microprocesador. Algunas veces, por ejemplo, unidades de ejecución, sean de enteros o de punto flotante, son apagadas para reducir el consumo. A este tipo de técnicas se les conoce como procesadores *cluster* [2]. Típicamente se construye el microprocesador con un *front-end* compartido por varios *back-ends*. El número de back-ends se determina dependiendo la aplicación que se ejecute, de esta forma el consumo de potencia dinámica se reduce varias veces porque únicamente se utiliza el hardware necesario para cada aplicación.

Otras técnicas para minimizar el consumo de potencia se centran en la planeación óptima de instrucciones en tiempo de compilación [7]. Estas buscan acomodar las instrucciones de manera que las líneas de las memorias cache sean accesadas en paralelo, permitiendo mejor rendimiento y menor consumo. Se han observado reducciones en el consumo de potencia dinámica de hasta un 66 % con dichas técnicas.

La combinación de técnicas de software (en tiempo de compilación) y de hardware podrían proveer mayores resultados.

En este trabajo nos concentramos en estimar el consumo de potencia dinámica en un

microprocesador superescalar para identificar cuales son los bloques que más consumen.

III. METODOLOGÍA

Para la realización del estudio se utilizó un simulador de microprocesadores a nivel microarquitectura [14]. En la tabla 1 se muestran las características del procesador superescalar con ejecución fuera de orden propuesto. La arquitectura utilizada para nuestro estudio fue Alpha [15].

TABLA 1. CONFIGURACIÓN DE UN MICROPROCESADOR SUPERESCALAR FUERA DE ORDEN.

Ancho de la etapa de fetch (en instrucciones)	4
Penalidad en fallo del predictor de saltos	3 ciclos
Tiempo de acceso del predictor de saltos	1
Tipo del predictor de saltos	Bimodal
Tamaño de la tabla del predictor de saltos	2048
Configuración de la tabla del segundo nivel del predictor de saltos	1 1024 8 0
Tamaño de la pila de call/return	8
Tamaño de la tabla de direcciones del predictor de saltos y asociatividad	512 4
Ancho de la etapa de decodificación	4
Ancho de la etapa de ejecución	4
Ancho de la etapa de liberación	4
Tamaño de la ventana de instrucciones	16
Tamano de la ventana de load/store	8
Memoria cache de datos de primer nivel	16KB, 4-way
Memoria cache de datos de segundo nivel	256KB, 4-way
Memoria cache de instrucciones	16KB, direct
Latencias de memoria	18, 2 (ciclos)
Ancho de bus de memoria (bytes)	8
TLB de instrucciones	4KB
TLB de datos	4KB
Penalidad en fallo de TLB	30 ciclos
Número de ALUs de enteros	4
Número de multiplicadores de enteros	1
Número de puertos de memoria	2
Número de ALUs de punto flotante	4
Número de multiplicadores de punto flotante	1

Para modelar el consumo de potencia dinámica se utilizó el modelo propuesto en [2]. Tal modelo proporciona parámetros para estimar el consumo de potencia dinámica. En tiempo de ejecución se actualizan contadores de actividad de cada uno de los bloques del procesador y estos se multiplican por sus factores de potencia. Al final de la ejecución se tiene la potencia dinámica consumida por cada bloque durante todo el programa.

Se utilizaron las aplicaciones SPECint2000: bzip2, crafty, gap, gcc, gzip, mcf, parser, perlbmk, twolf, vpr y SPECfp2000: ammp, applu, art, equake, galgel, lucas, mesa, mgrid, swim [11] en nuestros experimentos. Para todas las aplicaciones se ejecutaron 100 millones de instrucciones y se usaron las entradas *ref*.

IV. ANÁLISIS DE LA POTENCIA DINÁMICA CONSUMIDA

En la tabla 2 se muestra el consumo de potencia dinámica promedio por ciclo durante toda la ejecución. En cada ciclo de reloj se actualizan los contadores de potencia de todos los bloques para producir un resultado que se obtiene dividiendo la cantidad total de potencia consumida en cada bloque por el número total de ciclos consumidos en la ejecución del programa. Este resultado es el mismo en todas las aplicaciones porque los contadores de potencia agregan una cantidad constante predefinida a cada bloque en cada ciclo de reloj.

Las figuras 1 a 6 muestran los resultados de consumo de potencia dinámica promedio por ciclo de acceso en los diferentes bloques. En este cálculo se mide el consumo cuando se accesa a cada bloque, un acceso se consistuye por una secuencia que consume uno o más ciclos dependiendo del bloque. Los bloques analizados consumen tres ciclos, los resultados muestran el consumo en cada uno de ellos.

TABLA 2. POTENCIA DINÁMICA PROMEDIO POR CICLO DURANTE TODA LA EJECUCIÓN.

Bloque	Potencia dinámica promedio por ciclo (W)
rename	0.418
bpred	4.5231
window	2.2043
lsq	0.9632
regfile	3.5725
icache	2.4769
dcache	6.0849
dcache2	4.2091
alu	18.9412
falu	14.281
resultbus	2.2975
clock	26.0975
fetch	7.0001
dispatch	0.418
issue	34.7002
total	71.7882

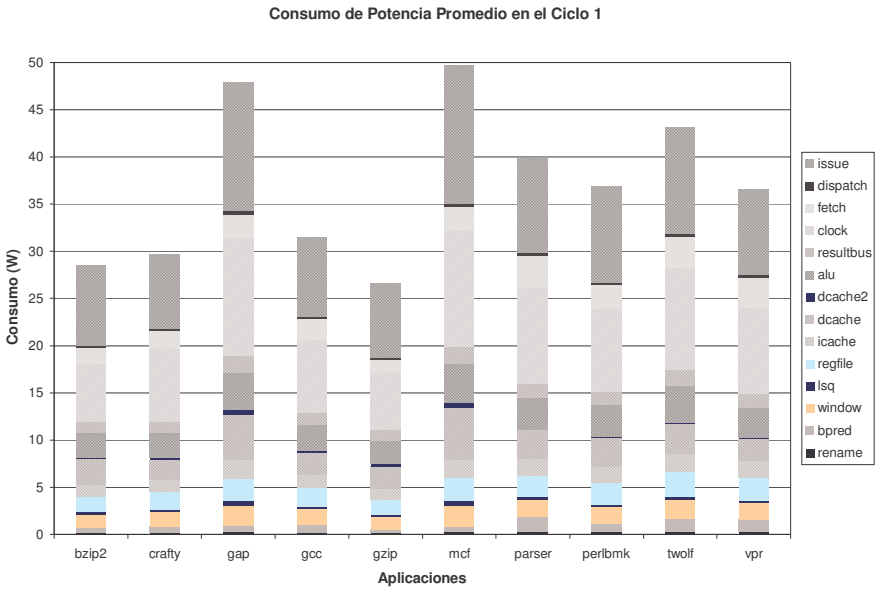

Figura 1. Consumo de potencia promedio en el ciclo 1 para las aplicaciones SPECint2000.

En la figura 1 se muestra el consumo de potencia dinámica en las aplicaciones SPECint2000. Se reportan consumos de los buffers de interconección del *pipeline*: *issue*, *dispatch* y *fetch*, el reloj, el bus de transferencia de resultados, las memorias cache de nivel 1 y 2, el banco de registros, la cola de *load/store*, la cola de instrucciones, el predictor de saltos y la tabla de renombramiento de registros. Se observa que en la mayoría de las aplicaciones, entre un 25 y un 30 % de la potencia es consumida por el buffer de *issue* y una cantidad similar es consumida por el circuito de reloj. A diferencia del resultado de la tabla 2, los valores mostrados en la figura 1 son diferentes para cada aplicación, porque los accesos a los diversos componentes son característicos de cada programa.

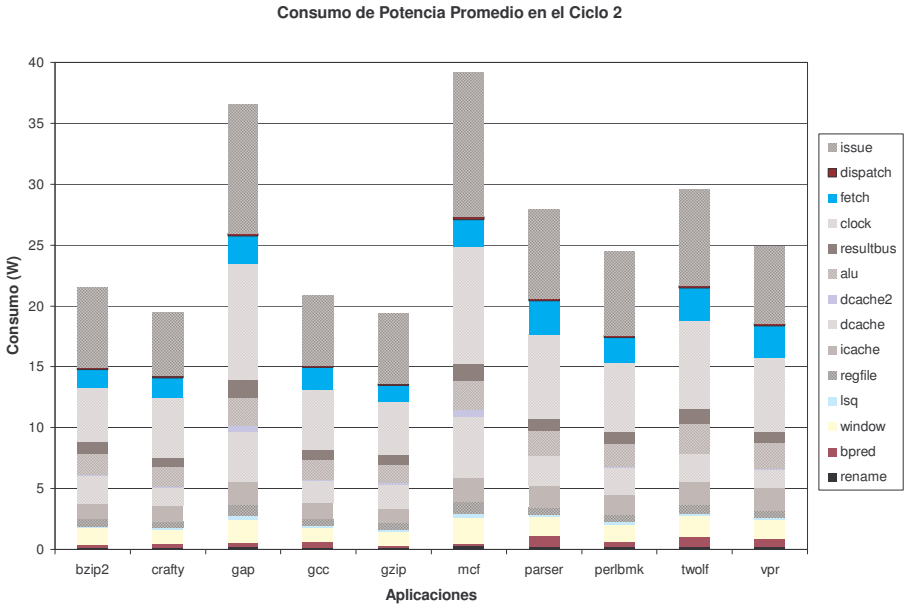

Figura 2. Consumo de potencia promedio en el ciclo 2 para las aplicaciones SPECint2000.

En la figura 2 se muestra el consumo de potencia de los mismos componentes para las mismas aplicaciones, pero en el ciclo 2 de un acceso. A diferencia de los resultados de la figura anterior, aquí se muestra una reducción global en el consumo entre un 30 % y un 40 %, esto se debe a que un número de accesos acertados sólo toma un ciclo.

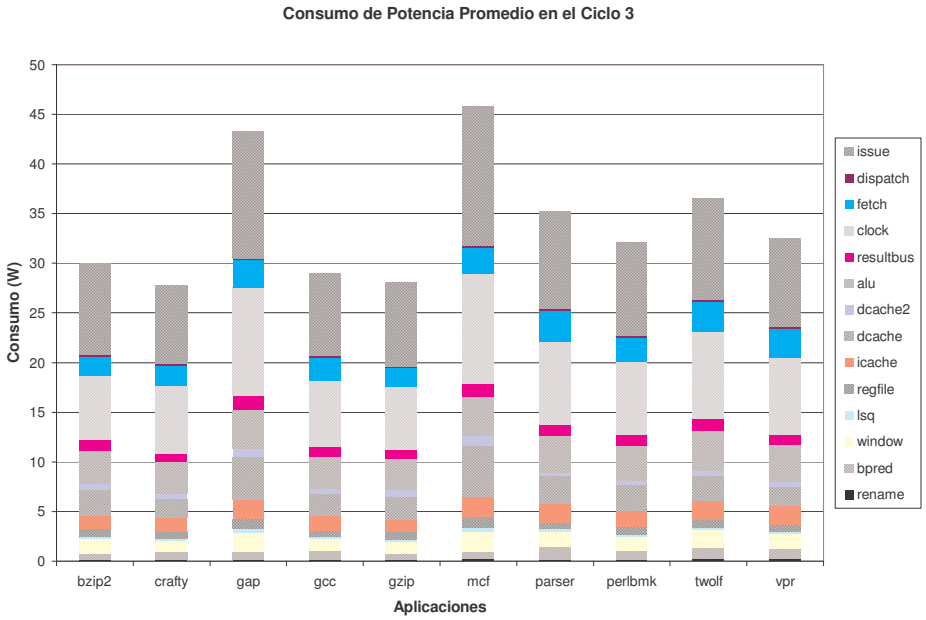

Figura 3. Consumo de potencia promedio en el ciclo 3 para aplicaciones SPECint2000.

En la figura 3 se muestran resultados similares a los de la figura anterior pero para el tercer ciclo. A diferencia del ciclo2, en este se observa un incremento de consumo. Tal incremento es causado por componentes que producen latencias mayores y por fallos en caches, tablas y predictores que originan un incremento en el número de accesos a otros niveles en la jerarquía que tienen latencias mayores.

Las figuras 4 a 6 muestran los resultados del consumo dinámico promedio en las aplicacione SPECfp2000 para los ciclos 1,2 y 3, respectivamente. Cabe señalar una importante reducción en el consumo dinámico de las aplicaciones de punto flotante. Como su código es más regular, se produce un mucho menor número de cambios de contexto y fallos en los diferentes niveles de caché.

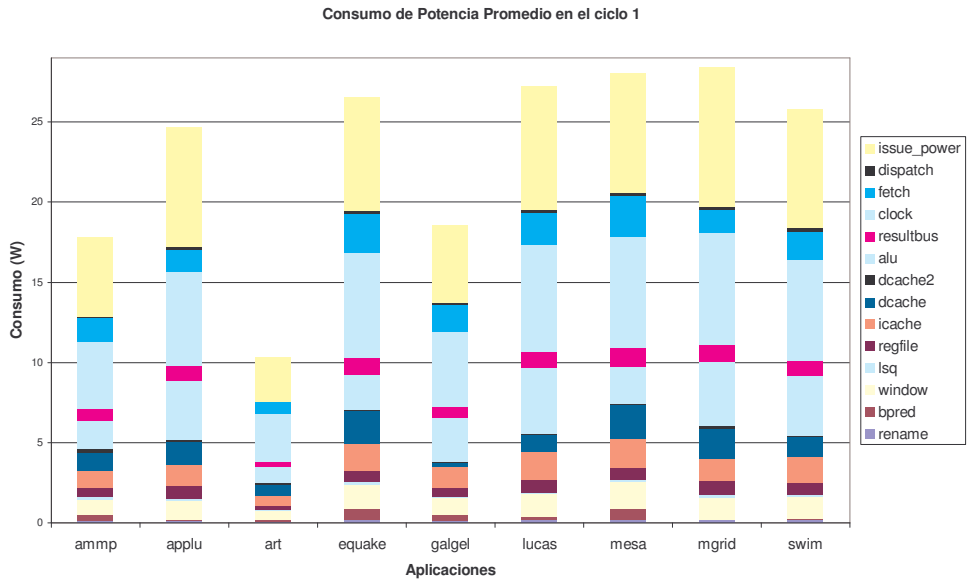

Figura 4. Consumo de potencia promedio en el ciclo 1 para las aplicaciones SPECfp2000.

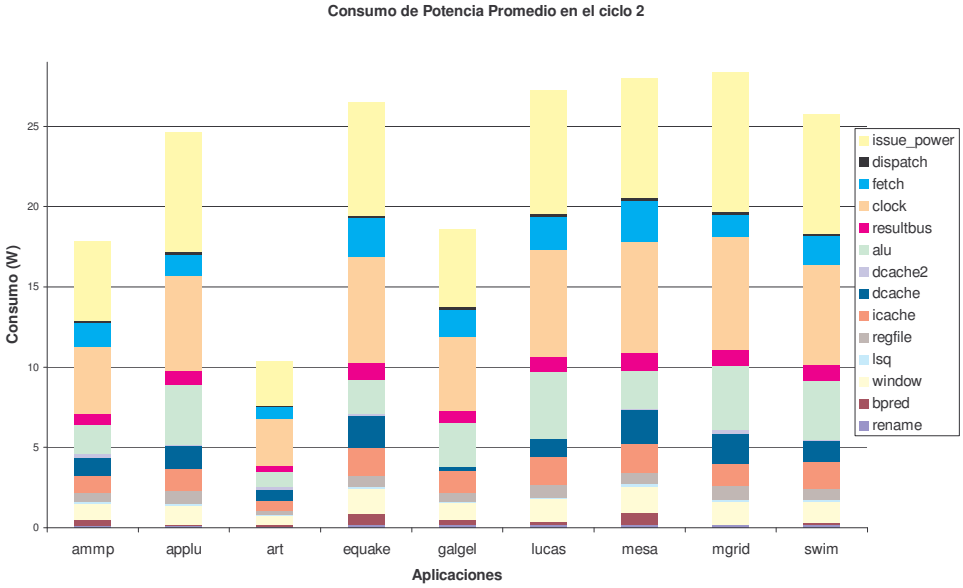

Figura 5. Consumo de potencia promedio en el ciclo 2 para las aplicaciones SPECfp2000.

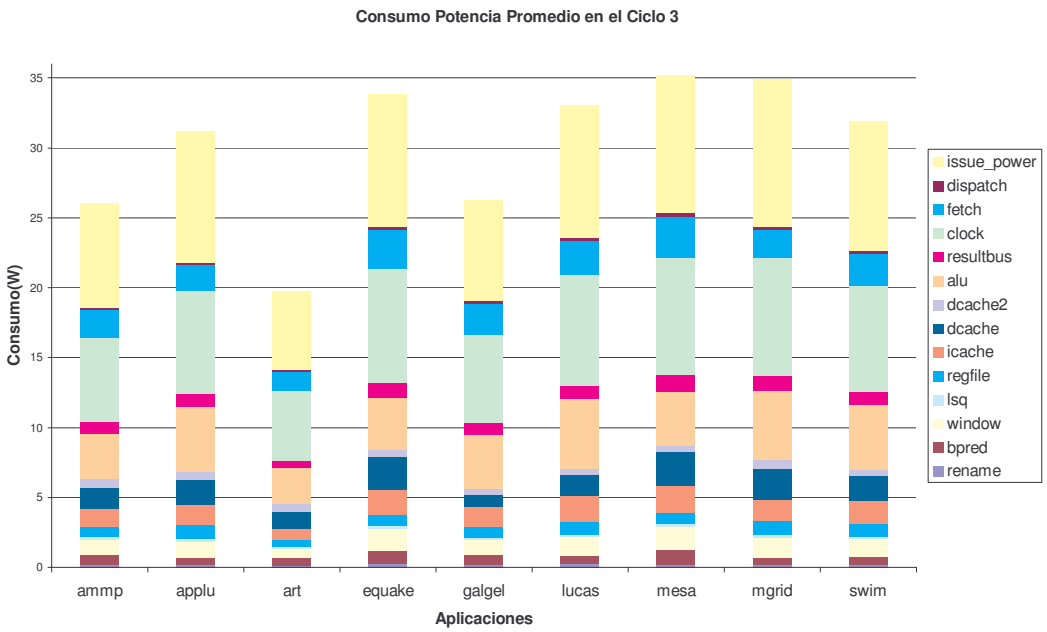

Figura 6. Consumo de potencia promedio en el ciclo 3 para las aplicaciones SPECfp2000.

V. CONCLUSIONES Y TRABAJO FUTURO

Los resultados obtenidos demuestran que más de la mitad de la potencia dinámica consumida se concentra en los buffers de interconección del pipeline y en el reloj. Por esto en un trabajo futuro se pretende proponer técnicas a nivel arquitectura para reducir el consumo en esos bloques. Se observó que al menos el 5 % de las instrucciones se reejecutan debido a fallos de predicción de saltos. Una mejor planeación dinámica de instrucciones reduciría el número de transacciones en el buffer de pipeline de issue, también podría utilizarse una tabla pequeña que contenga los valores más recientemente usados de donde puedan ser reutilizados sin tener que ser enviados por el buffer del pipeline. Para reducir el consumo causado por el reloj, se deben identificar secciones de ejecución del programa en los que se puede reducir la frecuencia de operación y construir una arquitectura con dominio de relojes multiples.

RECONOCIMIENTOS

Este trabajo es parte de un proyecto del Seminario de Investigación de Sistemas Digitales del ITESM CEM.

REFERENCIAS

[1] R. Schaller. “More´s law: past, present and future”,IEEE Spectrum, 1997.
[2] D. Brooks, V. Tiwari, M. Martonosi , “Wattch: A Framework for Architectural –Level Power Analysis and Optimizations”, in Proc. of the 27th International Sympsium on Computer Architecture. June 2000.
[3] A. Baniasadi, A. Moshovos, “Asymmetric-Frequency clustering: a power-aware back-end for high-performance microprocessors”, In Proc. of the 2002 international symposium on Low power electronics and design. Monterey, California, USA. pp. 255-258.
[4] Q. Wu, M. Pedram, X. Wu, “Clock-gating and its Application to Low Power Design of Sequential Circuits”. IEEE Custom Integrated Circuits Conference, 1997.
[5] T. Pering, T. Burd, R. Brodersen, “The simulation and evaluation of dynamic voltage scaling algorithms”, In Proc. of the 1998 Symposium on Low Power and Electronics.
[6] E. Talpes, D. Marculescu, “A critical analysis of application-adaptive multiple clock processors”, Proceedings of the 2003 international symposium on Low power electronics and design, Seoul, Korea, pp 278-281.
[7] W-T. Shiue, “Retargetable compilation for low power”, Proceedings of the ninth international symposium on Hardware/software codesign, 2001.

[8] M. Powell, S-H. Yang, B. Falsafi, K. Roy, T.N. Vijaykumar, “Gated-Vdd: a circuit technique to reduce leakage in deep-submicron cache memories”, Proceedings of the 2000 international symposium on Low power electronics and design, 2000.
[9] S. Kaxiras, Z. Hu, M. Martonosi, “Cache decay: exploiting generational behavior to reduce cache leakage power”, Proceedings of the 28th annual international symposium on Computer architecture, Volume 29 Issue 2.
[10] K. Flautner, NS Kim, S. Martin, D. Blaauw, T. Mudge, “Drowsy Caches: Simple Techniques for Reducing Leakage Power”, In Proc. of the 29th International Symposium on Computer Architecture. 2002.
[11] Standard Performance Evaluation Corporation. www.spec.org.
[12] L. Benini, G. de Micheli, “System-Level power optimization: techniques and tools”, In ACM Transactions on Design Automation of Electronic Systems (TODAES), Volume 5 , Issue 2 (April 2000), pp 115 – 192.
[13] L. Li, V. Degalahal, N. Vijaykrishnan, M.Kandemir, M. J. Irwin , “Power optimizations for cache memory: Soft error and energy consumption interactions: a data cache perspective”, Proceedings of the 2004 international symposium on Low power electronics and design , 2004.
[14] D. Burger and T. M. Austin and S. Bennett, "Evaluating Future Microprocessors: The SimpleScalar Tool Set", CS-TR-1996-1308, University of Wisconsin, Madison, 1996.
[15] R.E. Kessler, "The Alpha 21264 Microprocessor", IEEE Micro, March-April, 1999, pp. 24-36.