ارائه یک تکنیک ترکیبی برای کاهش مصرف انرژی در حافظه نهان به همراه افزایش کارایی

آزاده زمانی فر
دانشجوی کارشناسی ارشد، مهندسی کامپیوتر
دانشگاه علم و صنعت
az_zamanifar@comp.iust.ac.ir

میترا نصری
دانشجوی کارشناسی ارشد، مهندسی کامپیوتر
دانشگاه علم و صنعت
Mitra_nasri@comp.iust.ac.ir

محسن سریانی
استادیار دانشکده کامپیوتر
دانشگاه علم و صنعت
soryani@iust.ac.ir

چکیده: از مهمترین مسائل مطرح در تولید سیستم‌های موبایل، حسگرهای بی‌سیم و سایر پردازنده‌های نهفته، میزان مصرف انرژی در تراشه‌ی پردازنده و در حافظه‌ی نهان آن است این درحالی است که بسیاری از تلاش‌هایی که تاکنون برای غلبه بر اتلاف توان در این حافظه انجام شده‌اند، منجر به کاهش کارایی آن گردیده‌اند.

در این مقاله، با به کار بردن یکی از روش‌های کاهش نشتی توان در حافظه‌ی نهان، در کنار اِعمال تغییر اندک در کاربری ساختار LSQ، هم میزان مصرف انرژی پویا و ایستا در حافظه‌ی نهان کم شده است و هم کاهش کارایی ناشی از روش اول، توسط روش دوم جبران گردیده است. نتایج حاصل از آزمایشات روی شبیه‌ساز HotLeakage برای حافظه‌ی نهان داده، نشان‌دهنده کاهش۴,۵% زمان اجرا در کنار کاهش ۷۲,۵% درصدی نشتی انرژی در این حافظه بوده اند.

واژه های کلیدی: نشتی توان، LSQ، حافظه نهان، حافظه خوابیده، حافظه نیمه‌بیدار.

۱- مقدمه

با پیشرفت فن‌آوری در حوزه سیستم‌های نهفته و موبایل، نیاز به ابداع روش‌های جدیدی برای کنترل مصرف انرژی در سیستم به وجود امده است. از آنجا که اندازه‌ی تراشه‌ها رو به کاهش می‌رود، میزان نشتی انرژی حافظه‌ی نهان، کسر قابل توجهی از مصرف انرژی در پردازنده را به خود اختصاص می‌دهد [4].

مصرف انرژی در حافظه نهان بر دو نوع است؛ پویا (در زمان دسترسی به سلول‌های حافظه) و ایستا (برای نگهداری مقادیر درون سلول‌ها در طول زمان). بسیاری از تلاش‌های انجام شده برای کاهش نشتی توان در حافظه نهان، مبتنی بر کاهش مصرف انرژی ایستا، از طریق خاموش کردن (خواباندن[1]) [4] یا نیمه بیدار[2] [2] نگه داشتن سلول‌های حافظه بوده‌اند.

اولین روش ارائه شده برای خاموش کردن انتخابی خانه‌های حافظه‌ی نهان به هدف کاهش مصرف انرژی، توسط [10] ارائه گردید اما از آن‌جا که جریمه‌ی زمانی این روش بالاست، کارایی به نسبت زیادی، پایین می‌آید. پس از این روش، تا اواخر سال ۲۰۰۳، راه‌های اختصاصی‌تری توسط افرادی چون [2, 4, 11, 12] مطرح گردیده‌اند که در آن‌ها، خانه‌های مشخصی در هر مجموعه (Set)، خاموش می‌شوند. مبنای اصلی تمامی این تحقیقات، خاموش کردن یا به حالت نیمه بیدار بردن خانه‌هایی از حافظه نهان است که احتمال می‌رود بعد از این مورد استفاده نخواهند بود. اما تمامی این متدها همچنان با جریمه‌ی کاهش کارایی و مصرف انرژی در بیدار کردن داده‌ها یا دسترسی مجدد به حافظه‌ی نهان مواجه هستند.

در نهایت، در مقاله‌ای که توسط [6] در سال ۲۰۰۵ ارائه شد، مدلی برای تخمین حداکثر صرفه‌جویی ممکن در مصرف انرژی ایستا در انواع روش‌های حافظه خوابیده و نیمه بیدار، برای شرایطی که دانش کامل در مورد برنامه‌ها در دست است، مطرح گردید.

از دیگر سو، برای کاهش مصرف انرژی پویا، روش Cached-LSQ [7] ارائه شد که راه دیگری برای جلوگیری از اتلاف انرژی پویا از طریق کم کردن نیاز به دسترسی به حافظه نهان است. این روش پایه‌ی روش Efficient Cached LSQ در مقاله حاضر می‌باشد. در این تکنیک، علاوه بر دستورات بازنشسته نشده‌ی Store موجود در بافر LSQ، دستورات Load و نیز Store بازنشسته نیز در این بافر باقی می‌مانند تا در دستورات بعدی Load، بدون نیاز به دسترسی به حافظه نهان، مقدار داده را بازگردانند. تفاوت عمده‌ی EC-LSQ با C-LSQ در استفاده‌ی بهینه از فضای بافر LSQ می‌باشد. همچنین در مقاله حاضر برای کاهش مصرف انرژی ایستا در حافظه نهان داده، از روش نیمه‌بیدار استفاده خواهد شد. که نتایج بیانگر تاثیر ۷۲,۵% آن در کاهش مصرف این نوع انرژی برای حافظه داده می‌باشند.

بخش‌های بعدی مقاله عبارتند از مروری بر صورت مسئله، شرح روش نیمه بیدار در کاهش مصرف انرژی ایستا و روش EC-LSQ برای افزایش کارایی سیستم. در نهایت نیز تنظیمات مربوط به آزمایش‌های انجام شده، شرح آزمایش‌ها و حاصل آن‌ها و سپس نتیجه‌گیری، ذکر خواهند گردید.

۲- انتخاب روشی برای کاهش مصرف انرژی ایستا

همان طور که پیش از این عنوان گردید، حافظه خوابیده، روشی است که در آن سلول حافظه خاموش شده و مصرف انرژی آن تقریبا به صفر می‌رسد. این روش معمولا منجر به از دست دادن داده می‌گردد و جریمه‌ی بازیابی داده از حافظه‌ی سطح بعدی را خواهد داشت. در روش خوابیده سلول‌هایی از حافظه نهان که قرار نیست تا مدتی بعد (Off Time)، به کار برده شوند یا مورد استفاده مجدد قرار گیرند، خاموش می‌شوند.

حافظه نیمه بیدار، روشی است که در آن سلول حافظه، مقدار کمتری از انرژی حالت طبیعی خود را مصرف می‌کند اما نسبت به حالت خوابیده، جریمه‌ی کمتری برای بازیابی داده خواهد داشت. همچنین حافظه نیمه بیدار مقدار درون سلول‌ها را حفظ می‌کند. در نتیجه به هنگام فعال‌سازی مجدد miss تولید نمی‌شود.

در مقاله حاضر، برای انتخاب یک روش مناسب برای کاهش مصرف انرژی ایستا، چهار الگوریتم جدول ۱ ارائه شدند. تفاوت میان Fast Decay و Lazy Decay در زمان Off Time آن‌ها نهفته است. هرچه این زمان بیشتر باشد، حافظه دیرتر به حالت خاموش می‌رود. واحد این زمان بر اساس پیشنهاد مقاله‌ی [5]، کیلو سیکل است.

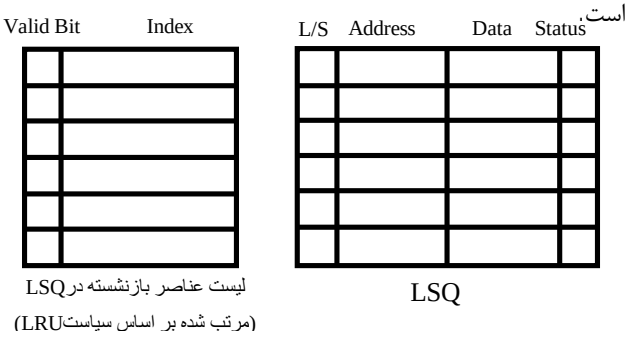
جدول(۱)- چهار الگوریتم پیشنهاد شده برای کاهش مصرف انرژی ایستا در حافظه نهان.

عنوان روش	Off Time	وضعیت حافظه پس از فعال شدن مکانیسم
Fast Decay	4KCycle	از دست دادن محتواي سلول‌ها. جريمه‌ي انتقال محتوا از حافظه سطح بعدي به اين خانه.
Lazy Decay	16KCycle	
Fast Drowsy	4KCycle	حفظ محتواي سلول‌ها براي بيدارسازي مجدد. جريمه‌ي زماني براي بيدارسازي.
Lazy Drowsy	16KCycle	

۳- Efficient CLSQ

برای بهبود میزان موازی‌سازی دستورات در پردازنده‌های امروزی، استفاده از بافرهای Load/Store با اندازه‌های بزرگتر، بسیار متداول شده است. با این حال، متوسط فضای اشغال شده در LSQ در طول زمان، در حدود ۵ تا ۱۰% است [7]. همین امر باعث به وجود آمدن فضای ذخیره‌سازی قابل توجهی در قسمت‌های استفاده نشده‌ی این بافر می‌گردد. روش حاضر، از فضاهای خالی‌ای که به طور موقت در این بافر تولید می‌شوند برای ذخیره‌ی داده‌ها استفاده می‌کند. در واقع دستورات load مقدار خود را در این بافر می‌یابند در نتیجه مراجعه به حافظه‌ی نهان کم می‌شود. انرژی مورد نیاز برای نگهداری از بافر LSQ و همچنین دسترسی load به آن، بسیار کم‌تر از دسترسی به حافظه‌ی نهان است. در واقع، با این کار، فقط به اندازه‌ی فضاهای جدیدی که در LSQ برای ذخیره داده‌ها استفاده می‌کنیم به میزان مصرف انرژی سیستم اضافه می‌شود.

شکل ۱، روش پیشنهادی را به نحو دقیق‌تری نمایش می‌دهد. در سمت چپ این شکل، ساختار نگهداری اندیس‌ها برای الگوریتم LRU به عنوان سیاست جایگزینی در لیست، ارائه شده است. در سمت راست نیز طبق روش [7] یک بیت L/S به معنی Load یا Store بودن دستور در LSQ، یک فیلد آدرس معادل آدرس حافظه مورد دسترسی، فیلد داده، حاوی داده‌ای که اخیرا در یک دستور Load خوانده شده یا قرار است در یک Store نوشته شود و نیز یک بیت وضعیت قرار گرفته است.



شکل(۱) ساختار پیشنهادی برای Efficient CLSQ

ایراد عمده‌ای که بر CLSQ، وارد می‌دانیم آن است که در شرایطی که با بافر پر سرعت و کوچکی (در حدود ۶۴ سلول داده) مانند LSQ، سروکار داریم، روش ساده‌ی FIFO به عنوان الگوریتم جایگزینی داده‌ها در این ساختار، باعث کاهش کارایی آن می‌گردد. در واقع، اثر یک الگوریتم جایگزینی بهتر در این بافر کوچک، قابل تامل است همچنان‌که این مطلب را در بخش نتایج آزمایش‌ها، در نمودار شکل ۵ مشاهده می‌نمایید.

۴- تنظیمات آزمایش

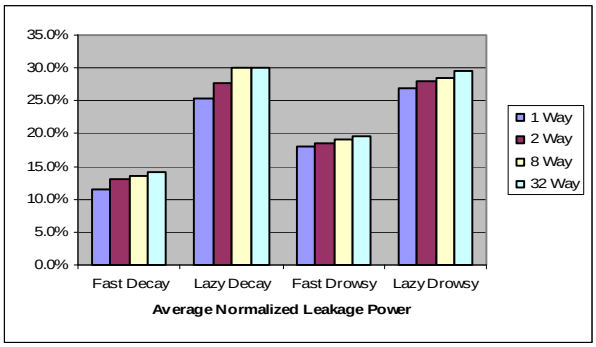
نتایج آزمایش، از اجرای برنامه‌های SPECint2000 در شبیه‌ساز HotLeakage نسخه‌ی 1.0.0.0 [13] به دست آمده‌اند. این شبیه‌ساز، مجموعه‌ای از فایل‌های مرجع است که روی Wattch 2.0 [9] سوار می‌گردد. موتور اصلی Wattch 2.0، Simplescalar 3.0 [1] است و پردازنده‌ی شبیه‌سازی شده در تمام آزمایشات، Alpha 21264 [14] می‌باشد برای اطلاعات بیشتر، پارامترهای آزمایش در جدول ۲ ارائه شده‌اند. همچنین به جای اجرای بخشی از هر برنامه، تمام قسمت‌های آن، اجرا گردیده‌اند زیرا میزان محلّیت برنامه در سرتاسر قسمت‌های آن، یکسان نیست [3].

جدول(۲) پارامترهای انجام آزمایش در شبیه‌ساز Hot Leakage.

Confing-Parameter	Value
Instruction Window	RUU size = 64, LSQ size = 32
Issue Width	4 instructions per cycle.
Functional Units	1 IntMult, 4 IntALU, 1 FpMult, 4 FpALU, 2 Memory Ports
Data Cache Level 1	Total Size 64 KB, 64Byte blocks, 1/2/8/32 ways, 1cycle latency.
Instruction Cache Level 1	Total Size 64 KB, 64Byte blocks, 1/2/8/32 ways, 1cycle latency.
Cache Level 2	Total Size 1 MB, 64Byte blocks, 8-way LRU, 6cycle latency.
Memory	100cycle Latency
Decay Extra Miss Energy Ratio	0.3

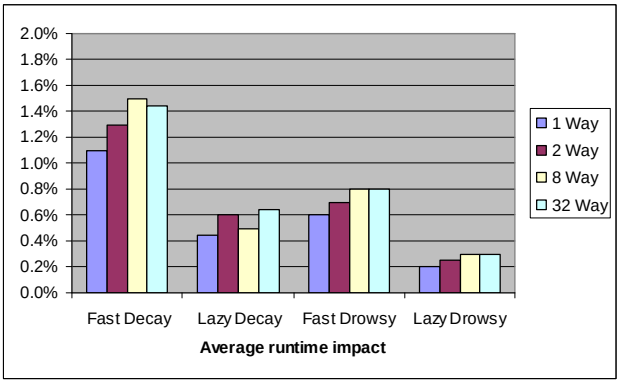
۵- نتایج آزمایش‌ها

برای مقایسه کارکرد روش پیشنهادی در این مقاله، ابتدا دو روش حافظه خوابیده و نیمه‌بیدار در اجرای برنامه‌های SPECint2000 مقایسه شده‌اند. تمام آزمایشات انجام شده روی حافظه داده و به آن دلیل است که تاثیر الگوریتم‌های حافظه خوابیده و نیمه بیدار، در حافظه داده بسیار واضح‌تر است. در شکل ۲، تفاوت اثر الگوریتم‌های جدول ۱ در میزان مصرف انرژی ایستا مشاهده می‌شود. این نمودار بر اساس مصرف انرژی در حالت طبیعی، نرمال شده است.



شکل (۲) تفاوت اثر الگوریتم‌ها بر متوسط نرمال شده‌ی مصرف انرژی.

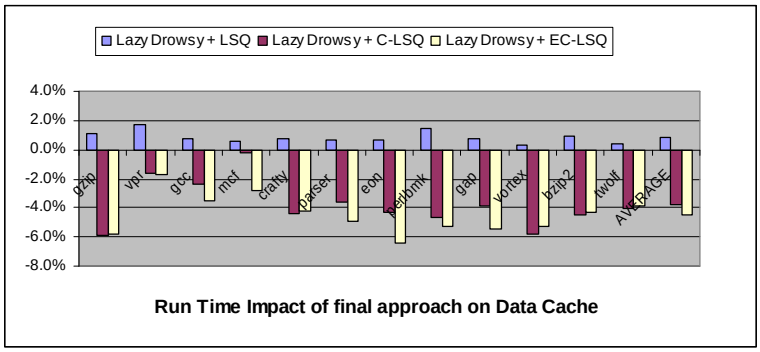
برای انتخاب بهترین گزینه، باید به نمودار شکل ۳ نیز توجه نمود. در این نمودار، اثر این الگوریتم‌ها بر متوسط زمان اجرا نمایش داده شده است. همان‌طور که مشخص است هر چهار الگوریتم، متوسط زمان اجرای برنامه‌های SPECint2000 را چند درصد افزایش داده‌اند. میزان این افزایش در روش Lazy Drowsy نسبت به همه کمتر بوده است.



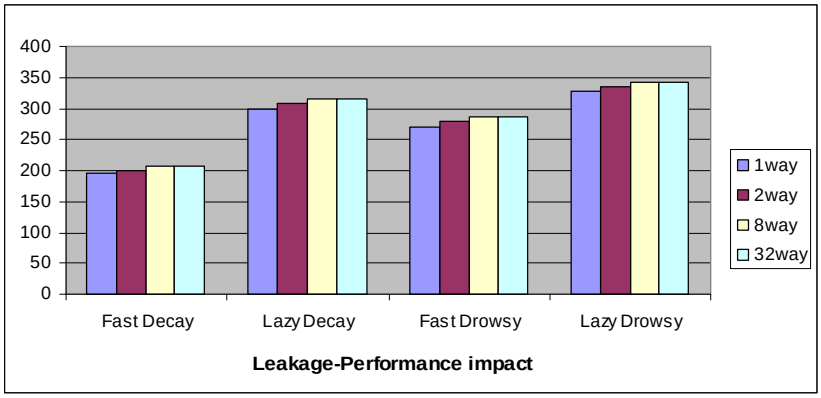
شکل (۳) تفاوت اثر الگوریتم‌ها بر متوسط زمان اجرا.

در نمودار شکل ۴ کارایی ترکیب روش lazy Drowsy (که به عنوان روش اصلی کاهش مصرف انرژی ایستا انتخاب شد)، با هر یک از روش‌های LSQ و C-LSQ و EC-LSQ نشان داده شده است. درصدهای منفی به معنی کاهش زمان اجرای برنامه‌ها توسط الگوریتم‌های ارائه شده هستند. مشاهده می‌شود که ترکیب هر دو روش C-LSQ و EC-LSQ با Lazy Drowsy منجر به افزایش کارایی می‌گردد. اما میزان این افزایش در روش EC-LSQ بیشتر است.

در شکل ۵ میزان افزایش کارایی و کاهش توان به عنوان معیار بهینه بودن تکنیک‌های مختلف در نظر گرفته شده است. به همین منظور، این دو عامل در یکدیگر ضرب شده‌اند، یعنی هر ستون از نمودار، برابر با حاصل ضرب کارایی بدست امده از روش در قدر مطلق میزان کاهش مصرف انرژی در آن روش است. همان طور که مشاهده می‌شود، تکنیک Lazy Drowsy از دیگر روش‌ها بهینه‌تر می‌باشد.



شکل (۴) تفاوت اثر استفاده از انواع مختلف LSQ در متوسط زمان اجرا. با تکنیک Lazy Drowsy



شکل (۵) تفاوت اثر استفاده از تکنیک‌های مختلف درکارایی و نشتی توان

۶- نتیجه گیری

این مقاله، یک معماری برای کاهش مصرف انرژی (ایستا) در حافظه نهان را ارائه داده که مبتنی بر ترکیب روش حافظه نیمه بیدار با ساختار تکامل یافته‌ی پیشنهادی برای LSQ، بدون کاهش کارایی می‌باشد. نتایج حاصل از آزمایش‌ها نشان داد که زمانی که روش Lazy Drowsy با EC-LSQ ترکیب شود، زمان اجرای نهایی برنامه‌ها به مقدار ۴٫۵% درصد کاهش می‌یابد در عین اینکه میزان انرژی مصرف شده در حافظه نهان نیز ۷۲٫۵% کم می‌شود.

گام‌های آتی برای بهبود مکانیسم Drowsy، استفاده از روش‌های پیشگویانه می‌باشد به بیان دیگر که به جز روش‌های مبتنی بر زمان، روش‌های مبتنی بر شرایط نیز در تصمیم‌گیری برای خاموش کردن سلول‌های حافظه دخیل باشند

۷- مراجع

[1] Austin, T., Larson, E., E., *"Simplescalar: An Infrastructure for Computer System Modeling"*, IEEE Computer, Vol. 35, No. 2, pp. 59–67, 2002.

[2] Flaunter, K., Kim, N. S. et al., *"Drowsy Caches: Simple Techniques for Reducing Leakage Power"*, In Proceedings of the 29th Annual International Symposium on Computer Architecture, IEEE Computer Society, pp. 148–157, 2002.

[3] Hennessy, J. L. Patterson, D., A., Computer Architecture: A Quantities Approach, 3rd ed., Morgan Kaufmann, San Mates, CA, 2003.

[4] Kaxiras, S., Hu, Z., Martonosi, M., *"Cache Decay: Exploiting Generational Behavior to Reduce Cache Leakage"*, In Proceedings of the 28th annual International Symposium on Computer Architecture, ACM Press, New York, pp. 240–251, 2001.

[5] Kaxiras, S., Xekalakis, P., Keramidas, G., "*A Simple Mechanism to Adapt Leakage-Control Policies to Temperature*", In Proceedings of 2005 International Symposium on Low Power Electronics and Design, pp. 54–59, 2005.

[6] Y., Sherwood, T., Kastner, "*On the Limits of Leakage Power Reduction in Caches*", 11th International Symposium on High-Performance Computer Architecture, 2005.

[7] Nicolaescu, D., Veidenbaum, A., Nicolau A., *"A Reducing Data Cache Energy Consumption via Cached Load/Store Queue"*, In Proceedings of the 2003 International Symposium on Low Power Electronics and Design, ACM Press, pp. 252–257, 2003.

[8] Standard Performance Evaluation Corporation, SPEC CPU2000 v1.1, September 2000, http://www.spec.org/cpu/CINT2000/.

[9] Brooks D., Tiwari V., Martonosi M.," Wattch: a Framework for Architectural-Level Power Analysis and Optimizations", In ISCA, pp. 83-94, 2000.

[10] Albonesi, D. H., "*Selective Cacheways: On-Demand Cache Resource Allocation*", In Proceedings of the 33rd annual ACM/IEEE International Symposium on Micro architecture, IEEE Computer Society, pp. 248–259, 1999.

[11] Hanson, H., Hrishikesh, M., Agarwal, V., Keckler, S., Burger, D., *"Static Energy Reduction Techniques for Microprocessor Caches*", In Proceedings of 2001 International Conference on Computer Design, IEEE Computer Society, pp. 276–283, 2001.

[12] Kim, N. S., Flaunter, K., Blaauw, D., Mudge, T., *"Drowsy Instruction Caches: Leakage Power Reduction using Dynamic Voltage Scaling and Cache Sub-bank Prediction"*, In Proceedings of the 35th annual ACM/IEEE International Symposium on Micro architecture, IEEE Computer Society Press, pp. 219–230, 2002.

[13] Zhang, Yan, Parikh, Dharmesh, Sankaranarayanan Karthik, Skadron, Kevin, Stan, Mircea,"HotLeakage:A Temperature-Aware Model of Sub Threshold and Gate Leakage for Architects", Technical Report, University of Virginia Dept. of Computer Science, 2003.

[14] Kessler, R.," *The Alpha 21264 Microprocessor*", IEEE Micro 19, 2, pp.24–36, 1999.

زیر نویس‌ها

[1] Decay: به حالتی از سلول‌های حافظه گفته می‌شود که مصرف انرژی در آن صفر است.

[2] Drowsy: به حالتی از سلول‌های حافظه گفته می‌شود که مصرف انرژی در آن‌ها تقریبا نصف است.