Near-memory Caching for Improved Energy Consumption

Nevine AbouGhazaleh, Bruce Childers, Daniel Mossé, Rami Melhem
Department of Computer Science
University of Pittsburgh
(nevine, childers, mosse, melhem)@cs.pitt.edu

Abstract

Main memory has become one of the largest contributors to overall energy consumption and offers many opportunities for power/energy reduction. In this paper, we propose a Power-Aware Cached-DRAM (PA-CDRAM) organization that integrates a moderately sized cache directly into a memory module. We use this near-memory cache to turn a memory bank off immediately after it is accessed to reduce power consumption. We modify the structure of cached DRAM (CDRAM) with the goal of reducing energy consumption while retaining the performance advantage for which CDRAM was originally proposed. We evaluate the approach using a cycle accurate processor and memory simulator. Our results show that PA-CDRAM achieves up to 84% (28% on average) improvement in the energy-delay product and up to 76% (19% on average) savings in energy when compared to a time-out power management technique.

1 Introduction

Energy consumption is a limiting constraint for both embedded and high-performance systems. In embedded systems, the lifetime of a device is limited by the rate of energy dissipation from its battery. On the other hand, energy consumption in high-performance systems increases thermal dissipation, thus requiring more cooling resources and accordingly increasing the system's maintenance overhead. For the majority of these systems, the memory subsystem consumes a large portion of the overall energy dissipation, which motivates the need for efficient memory power management.

Memory has a huge internal bandwidth compared to its external bus bandwidth [1]. To exploit the wide internal bus, *cached DRAM* (CDRAM) adds an SRAM cache to the DRAM array on the memory chip [2] as shown in Figure 1. Such a *near-memory cache* acts as an extra memory hierarchy level, whose fast latency improves the average memory access time and thus improves system performance, provided that the near-memory cache is appropriately configured.

In this paper, we explore the energy saving obtained by placing SRAM caches closer to the memory, rather than closer to the CPU. Our evaluation focuses on the memory's energy consumption as well as the system's overall performance. We integrate a moderately sized cache within the chip boundary of a power-aware multi-banked memory. We call this organization *power-aware cached DRAM, (PA-CDRAM)*. In addition to improving performance, PA-CDRAM significantly reduces energy consumption in caches and in main memory. Cache energy is reduced because (1) using small caches distributed to the memory chips reduces the cache access energy compared to using a large non-distributed cache, and (2) near-memory caches allow the access of relatively large blocks from memory, which is not affordable with *far-from-memory* caches. Memory energy consumption is reduced by having longer memory idle period during which DRAM banks can be powered off. PA-CDRAM improves the original CDRAM by tackling the interplay of the cache and memory organizations to optimize the memory's performance and energy consumption.

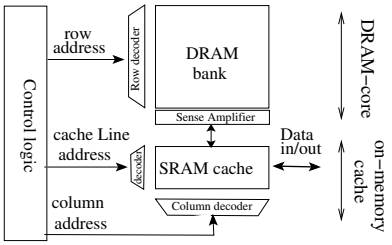

Figure 1: Functional block diagram of a CDRAM

2 PA-CDRAM

CDRAM was originally proposed to improve system performance; however, it was not designed as a replacement for power-aware memory. In comparison with traditional memory hierarchy[1], CDRAM has a good performance improvement over traditional memory (reaches 50%); however, total memory energy consumption suffers dramatically (increase can reach 1.5x to 3.0x) [4]. This increase is due to the extra energy consumed in accessing the near-memory caches and transferring more data from the DRAM-core at large block sizes.

The energy penalty of CDRAM can be overcome by improving the miss rate of the near-memory cache and using power management for the DRAM-core. As we will show, we do not only improve the energy penalty of CDRAM, but also significantly improve the *overall* memory energy in comparison to a traditional power-aware memory hierarchy.

[1]CDRAM uses near-memory cache configuration as in [3].

Because DRAM power management typically relies on *idleness* to select power states, an improved miss rate in the near-memory cache increases the amount of idleness in the DRAM-core, leading to more effective power management. One way to improve miss rate is to increase the capacity of the near-memory cache. However, the total energy of the whole memory hierarchy is increased due to greater overall cache capacity. Instead, we propose to re-allocate existing cache capacity from the memory hierarchy's lowest cache level to near-memory cache. For example, it may be possible to allocate the capacity of the L3 cache to CDRAM's near-memory cache. The L3 cache could then be eliminated, possibly without harming application performance. Moving cache capacity to the near-memory cache has three advantages. First, the near-memory caches are distributed among the memory chips, which leads to lower energy consumption because the individual caches are smaller than one monolithic cache. Second, large data transfers are possible from the DRAM-core to the near-memory cache (i.e., there is more memory bandwidth, which makes large block sizes feasible). Finally, the near-memory caches can filter accesses to the DRAM-core. Such filtering increases idleness and lets the DRAM-core stay in a low power state for longer periods.

To build a power-aware cached DRAM, there are two main challenges that must be addressed: (1) how to manage the DRAM-core's power and (2) what is the best configuration for the near-memory cache to balance energy and performance. We describe each of these challenges and our way of addressing them below.

2.1 Memory power management

With a near-memory cache, we propose applying aggressive power management in the DRAM-core. During a chip's idle time, the memory controller can immediately transition the DRAM-core to the sleep state after servicing all outstanding requests. This is equivalent to the use of a timeout policy with an idle threshold of zero seconds. Although a zero-threshold policy increases the total inactive time, it can degrade performance and increase the total energy consumption when too many requests are directed to a memory chip. The extra delay and energy overheads are due to the transitional cost between power states.

In PA-CDRAM, we avoid this problem by choosing the near-memory cache configuration in a way that increases the hit rate while reducing the DRAM-core's energy consumption. When most data requests are serviced as cache hits in the near-memory cache, the inter-arrival time between requests that reach the DRAM-core increases, making it cost effective to immediately deactivate banks after servicing outstanding requests. We choose to keep the near-memory cache active all the time to avoid delays that may be caused by on-demand activation of the cache at each request.

2.2 Memory energy trade-off

To reduce the memory's energy consumption, we need to consider the effect of the near-memory cache configuration on the energy consumption of both the near-memory cache and the DRAM-core. The two factors that affect the cache energy consumption and access latency —for a given cache size and fixed number of cache subbanks —are the *associativity* and the *block size* [5].

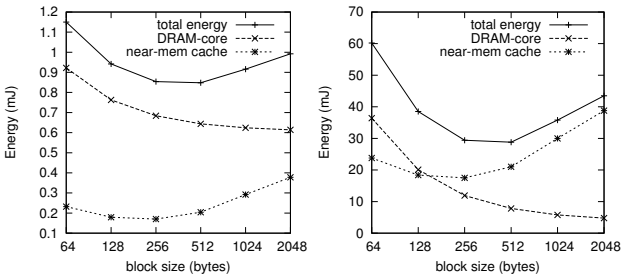

Figure 2: The effect of varying the cache block size on the memory energy consumption for *bzip* (left) and *mcf* (right).

The cache associtivity directly affects miss rate. Increasing associtivity reduces cache miss rate and vice versa. One goal of PA-CDRAM is to keep the near-memory cache miss rate as low as possible because it directly influences memory energy consumption in two ways. First, the higher the miss rate, the more activity in the DRAM in terms of transitioning from sleep to active state, performing address decoding, and transfer of data. Second, the lower the miss rate, the longer the DRAM-core idle time. To keep the miss rate at a minimum, we use fully associative caches to eliminate any conflict misses. We argue that in most cases, performance improvement and energy saving from reducing near-memory miss rates can outweigh extra delay and energy consumed in accessing higher associative caches.

Choosing a near-memory cache block size creates a trade-off between the near-memory cache and DRAM-core energy consumption. Small cache blocks have the advantage of fast hit time and low energy per access. However, smaller blocks imply frequent accesses and consequently increasing the DRAM-core energy due to the increased activity. . The increase in the DRAM-core energy rises as a result of increasing the memory activity (such as power-state transitions, address decoding, and data transfer). Conversely, larger near-memory cache block sizes reduce the DRAM-activity but increases the near-memory cache energy consumption and latency due to accessing these large blocks.

This trade-off is illustrated in Figure 2. The figure show variation of the energy consumption in near-memory cache and DRAM-core with the cache block size. Energy values are obtained using a simplified energy model. The model estimates the near-memory cache energy consumption, E_{cache}, and DRAM-core, E_{Dcore} as a function of the number of near-memory cache accesses (c_access) and DRAM-core accesses (d_access), respectively. We estimate $E_{cache} = E_{c\_access} \cdot \#c\_access$ and $E_{Dcore} = E_{d\_access} \cdot \#d\_access + E_{trans} \cdot \#trans + P_{idle} \cdot T_{idle}$. The cache access energy (E_{c_access}) is obtained from the Cacti tool [5][2], while the DRAM-core's energy per access (E_{d_access}), the power state transition energy (E_{trans}), and the idle power (P_{idle}) are specifications of an RDRAM memory chip [6].

[2]Cache energy values are obtained for a 256KB cache

To simplify the energy memory estimation, we assume an application with the following behavior: (1) has high spatial locality; that is, all data accessed from the DRAM-core are read/written by an application, (2) each cache block is read only once from the DRAM-core, and (3) uses immediate deactivation of DRAM-core after each access. Thus, $\#d\_access = \frac{\#c\_access}{blocksize}$ and $\#trans = 2 \cdot \#d\_access$. T_{idle} is an application's execution time minus time spent accessing data from the DRAM-core and transitioning between power states.

From this model, given the number of L2 misses and the approximate execution time for an application, we can roughly estimate the memory energy consumption at different block sizes. We use simplescalar [7] to estimate the two input parameters. In Figure 2, we show estimated energy for two of the SPEC2000 benchmark: *bzip* and *mcf* as an example of CPU and memory intensive applications, respectively. In *bzip*, the DRAM-core idle energy dominates the PA-CDRAM energy, while in *mcf*, the frequent accesses to the near-memory caches makes the cache energy dominate the total energy at large block sizes. From the figure, we see that the trade-off between the near-memory cache and DRAM-core energy consumption creates a sweet spot between block sizes 256 and 512 bytes. Note that, ideal block size varies in applications with different degree of locality and memory traffic intensity. However, from our simulation, in most of the applications, the minimum energy-delay product can be achieved at, or within a slight margin of, one of these two block sizes.

From this section, we conclude that for the given cache size, the near-memory cache should be fully associative and have a block size of either 256 or 512 bytes. For the DRAM-core, setting the chip to the sleep state after servicing outstanding requests is expected to save the memory energy consumption. The implementation described in the next section uses such configuration.

3 PA-CDRAM implementation

Our PA-CDRAM design modifies the original RDRAM design for power efficiency. Beside the addition of the near-memory cache, some alterations are needed in the main components of the RDRAM, namely: the DRAM-core, and the control logic.

Near-memory cache: We add a fully associative cache (depicted as dark blocks in Figure 3) with its data divided array into two sections. Since the original RDRAM design has a divided data bus, each section of the cache is connected to one of two internal data busses (DQA and DQB). Each cache section stores half of each cache block. We keep the write buffers in the original RDRAM before the sense amplifiers. The write buffers are used to store replaced dirty blocks from the near-memory cache to be written to the DRAM-core. The power state of this cache is independent of the power state of the other chip components. In the nap state, the RDRAM internal clock is periodically synchronized with the external clock [6]. Thus, the near-memory cache is accessible even when the DRAM is in the nap state.

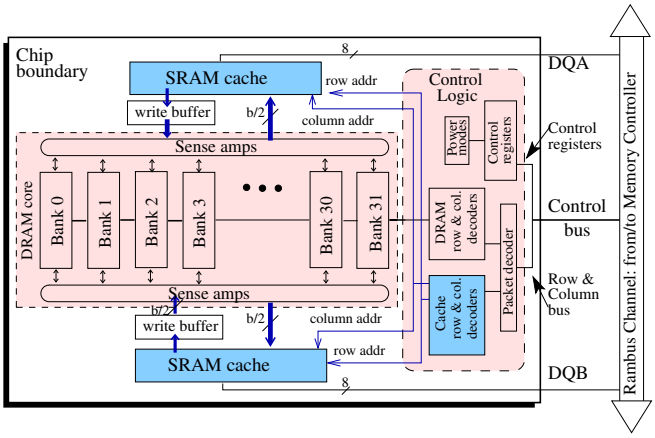

Figure 3: Functional block diagram of a PA-CDRAM.

DRAM-core: To accommodate the large transfer sizes between the DRAM-core and the near-memory cache, wider internal data busses are required to connect the sense amplifiers with the near-memory cache. The width of each bus connecting the DRAM and near-memory caches is $b/2$ bits, where b is the size of a cache block. The busses DQA and DQB remain at an 8-bit width.

Control logic: Extra cache row and column decoders are added in the chip's control logic for decoding cache addresses. In addition, the existing RDRAM packet decoder in each chip is modified to decode new cache commands. The new cache commands indicate whether the access is a hit/miss in the cache, and whether replacement is needed. We take advantage of the Rambus bus to define the cache commands without the need to add extra control pins in memory chips.

4 Evaluation

We evaluate PA-CDRAM using Simplescalar architecture simulator [7] and an integrated RDRAM memory simulator [8]. Simulations are performed using a set of applications from the SPEC2000 benchmark suite. Our study evaluates PA-CDRAM against a base case that employs traditional power saving policies implemented by Rambus. The memory controller uses a linear memory page allocation as used in [9, 10] with a timeout of 1000 cycles that achieves the best average energy-delay product across all simulated applications. The cache hierarchy consists of 32KB instruction and data L1 caches, 256 KB L2 cache, and 2 MB L3 cache. Memory consists of eight 32 MB RDRAM chips. In PA-CDRAM we replace the L3 cache with eight near-memory caches integrated into PA-CDRAM chips, such that the total near-memory cache capacity equals the L3 capacity. We use interleaved memory mapping to make use of all the near-memory caches. We compute the energy consumption in the DRAM-core, caches and busses. We use Cacti 3.0 [5] estimation of the cache access energy and delay. For the memory chips we use the specification of the RDRAM 256Mb/1066MHz/32 split bank architecture [6]. We use bus models in [11] to estimate the memory bus energy. From our simulation with

different block size we found that near-memory caches with 512B block size yields the least normalized energy-delay averaged over all applications. We use this block size to obtain the results. To analyze the benefits of PA-CDRAM on energy and performance independently, we evaluate energy-delay product, execution time and energy consumption of twelve benchmarks. Figure 4 shows these metrics normalized to the base case.

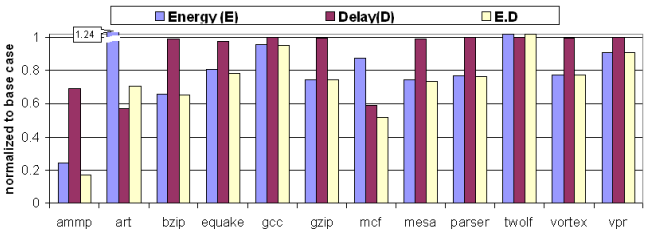

Figure 4: PA-CDRAM energy-delay break down.

For most applications, there is no significant improvement in the delay over the base case due to two reasons. First, in most of the applications, L2 can service a large number of the memory requests. Second, most of the CPU stalls resulting from L3 misses can be masked by the execution of other independent instructions in the pipeline. Three exceptions in Figure 4 are *ammp, art* and *mcf*. For these benchmarks, the near-memory miss rate significantly improved. In addition, these applications are characterized by heavy memory traffic; thus, a reduction in average memory access time significantly improved performance. From this we conclude that: near-memory caching well suits memory intensive applications.

From Figure 4, we also see that energy savings alone reached up to 76%. All applications exhibit energy savings except for *art* and *twolf*. These two applications exhibit poor spatial locality. This poor locality– combined with the large block size– causes higher energy consumption due to accessing data from the DRAM-core that is never requested by the applications. Thus, near-memory caching can perform better in applications with high spatial locality.

5 Related work

To exploit the large internal bandwidth of memory chips, an extra on-chip cache was introduced by Mitsubishi [12]. Hsu et al. [2] evaluated the performance of CDRAM in vector supercomputers. To improve the CDRAM performance, Kedem et al. [13] proposed a cached-DRAM with wide cache line ranging from 4KB to 8KB interleaved across multiple DRAM banks. Whereas Hegde et al [3] proposed using variable width cache lines that fit the application access pattern to save energy consumed in unnecessary traffic between the DRAM-core and the near-memory cache. Past work did not explore the energy savings over power-aware memory.

Lebeck et al [9] proposed the use of a power-aware allocation policy where data is allocated sequentially in each bank to increase a bank idle periods. An implementation of the memory power manager in the Linux operating system [10] allocates memory pages to banks based on the running applications.

6 Conclusion

In this paper, we explore the energy efficiency of *near-memory caches* rather than conventional cache hierarchies, where the L3 cache is "closer" to the CPU. PA-CDRAM can be used as an alternative to traditional power-aware memories to conserve energy and improve performance. PA-CDRAM reduces the memory's energy consumption by (1) bringing the cache closer to the memory to exploit the high internal memory bandwidth, and (2) distributing the external cache into smaller caches that have a low access energy and latency, and (3) increasing the DRAM-core idle periods due to the low miss rates of near-memory caches. Our evaluation showed that, when compared to traditional memory using a time-out power management, PA-CDRAM saves up to 76% energy consumption (19% on average). Moreover, PA-CDRAM reduces the energy-delay product by up to 84% (28% on average), where the highest gains are for memory-intensive applications with high spatial locality.

References

[1] D. Elliott, W. Snelgrove and M. Stumm, "Computational RAM: A Memory-SIMD Hybrid and its Application to DSP", in *Custom Integrated Circuits Conference*, 1992.

[2] W. Hsu and J. Smith, "Performance of cached DRAM organizations in vector supercomputers", in *Proc. Intl. Symp. on Computer Architecture*, pp. 327–336, 1993.

[3] A. Hegde, N. Vijaykrishnan, M. Kandemir and M.J. Irwin, "VL-CDRAM: variable line sized cached DRAMs", in *Proc. of the Intl. Symp. on Hardware/software codesign & system synthesis*, pp. 132–137, 2003.

[4] N. AbouGhazaleh, B. Childers, D. Mosse' and R. Melhem, "Energy Conservation in Memory Hierarchies using Power-Aware Cached-DRAM", in *Proc. of the Dagstuhl Seminar on Power-aware Computing Systems*. Dagstuhl Research Online Publication Server, April, 03–08 2005.

[5] P. Shivakumar and N. Jouppi, "CACTI 3.0: An Integrated Cache Timing, Power, and Area Model", Technical Report 2001.2, Compaq research labs, 2001.

[6] "Rambus", 2005, http://www.rambus.com/products.

[7] "Simplescalar simulator/PISA Tool Set version 3.0d", http://www.simplescalar.com.

[8] "SDRAM and RDRAM modeling for Simplescalar simulator", 2003, http:// www.tik.ee.ethz.ch/~ip3/software.

[9] A. Lebeck, X. Fan, H. Zeng and C. Ellis, "Power aware page allocation", in *Proc. of the Intl. conf. on Architectural support for programming languages and OS*, 2000.

[10] H. Huang, P. Pillai and K. Shin, "Design and Implementation of Power-Aware Virtual Memory", in *USENIX Annual Technical Conf.*, pp. 57–70, 2003.

[11] I. Kadayif, T. Chinoda, M. Kandemir, N. Vijaykirsnan, M. J. Irwin and A. Sivasubramaniam, "vEC: virtual energy counters", in *Workshop on Program analysis for software tools and engineering*, pp. 28–31, 2001.

[12] B. Davis, *Moderan DRAM Architectures*, PhD thesis, University of Michigan, Ann Arbor, 2000.

[13] R. Koganti and G. Kedem, "WCDRAM: A Fully Associative Integrated Cached-DRAM with Wide Cache Lines", Technical report, Duke University, CS dept., 1997.