Technology Comparison for Large Last-Level Caches (L^3Cs): Low-Leakage SRAM, Low Write-Energy STT-RAM, and Refresh-Optimized eDRAM

Mu-Tien Chang*, Paul Rosenfeld*, Shih-Lien Lu†, and Bruce Jacob*
*University of Maryland †Intel Corporation
*{mtchang, prosenf1, blj}@umd.edu †shih-lien.l.lu@intel.com

Abstract

Large last-level caches (L^3Cs) are frequently used to bridge the performance and power gap between processor and memory. Although traditional processors implement caches as SRAMs, technologies such as STT-RAM (MRAM), and eDRAM have been used and/or considered for the implementation of L^3Cs. Each of these technologies has inherent weaknesses: SRAM is relatively low density and has high leakage current; STT-RAM has high write latency and write energy consumption; and eDRAM requires refresh operations. As future processors are expected to have larger last-level caches, the goal of this paper is to study the trade-offs associated with using each of these technologies to implement L^3Cs.

In order to make useful comparisons between SRAM, STT-RAM, and eDRAM L^3Cs, we model them in detail and apply low power techniques to each of these technologies to address their respective weaknesses. We optimize SRAM for low leakage and optimize STT-RAM for low write energy. Moreover, we classify eDRAM refresh-reduction schemes into two categories and demonstrate the effectiveness of using dead-line prediction to eliminate unnecessary refreshes.

A comparison of these technologies through full-system simulation shows that the proposed refresh-reduction method makes eDRAM a viable, energy-efficient technology for implementing L^3Cs.

1. Introduction

Last-level cache (LLC) performance is a determining factor of both system performance and energy consumption in multi-core processors. While future processors are expected to have more cores [17], emerging workloads are also shown to be memory intensive and have large working set size [31]. As a result, the demand for large last-level caches (L^3Cs) has increased in order to improve the system performance and energy.

L^3Cs are often optimized for high density and low power. SRAMs (static random access memories) have been the mainstream memory technology for high performance processors due to their standard logic compatibility and fast access time. However, relative to alternatives, SRAM is a low-density technology that dissipates high leakage power. STT-RAMs (spin-transfer torque magnetic random access memories) and eDRAMs (embedded dynamic random access memories) are potential replacements for SRAMs in the context of L^3C due to their high density and low leakage features. For instance, eDRAM was used to implement the last-level L3 cache of the IBM Power7 processor [21].

Though they provide many benefits, both STT-RAM and eDRAM have weaknesses. For instance, STT-RAM is less reliable and requires both a long write time and a high write current to program [29]; eDRAM requires refresh operations to preserve its data integrity. In particular, as cache size increases, each refresh operation requires more energy and more lines need to be refreshed in a given time; thus, refresh becomes the main source of eDRAM power dissipation, as shown in Figure 1a [41]. Moreover, as technology scales down, increasing leakage and smaller storage capacitance result in shorter retention time, which in turn exacerbates the refresh power problem, shown in Figure 1b. Process and temperature variations also negatively affect the eDRAM data retention time. An eDRAM cache thereby requires higher refresh rate and refresh power to accommodate the worst case retention time.

In this paper, we evaluate energy-efficient L^3Cs built with SRAM, STT-RAM, and eDRAM, each optimized according to its technology. Specifically, the experimental SRAM L^3C is optimized for low leakage power, and the STT-RAM L^3C is optimized for low write energy. Moreover, we apply a practical, low-cost dynamic dead-line prediction scheme to reduce eDRAM refresh power. Refresh operations to a cache line are bypassed if the line holds data unlikely to be reused. The prediction scheme introduces only insignificant area overhead (5%), power overhead (2%), and performance degradation (1.2%), while at the same time greatly reducing the refresh power (48%) and the L^3C energy consumption (22%). Based on the workloads considered, we show that the eDRAM-based L^3C achieves 36% and 17% energy reduction compared to SRAM and STT-RAM, respectively.

The main contributions of this paper are as follows:

1. We quantitatively illustrate the energy breakdown of L^3Cs. In particular, we highlight the significance of refresh and its impact on eDRAM L^3C energy consumption.
2. We classify eDRAM refresh-reduction schemes into two categories and show that the use of dead-line prediction eliminates unnecessary refreshes.

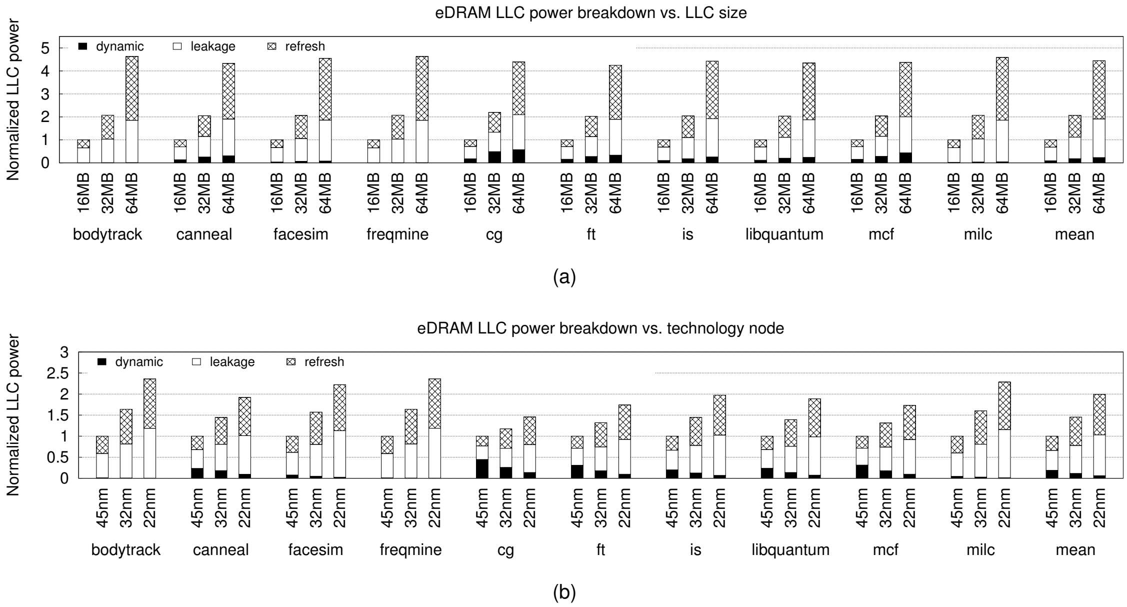


Figure 1: Embedded DRAM LLC power breakdown. (a) Refresh power vs. cache size. (b) Refresh power vs. technology node. As cache size increases, technology scales down, refresh becomes the main source of power dissipation.

3. We demonstrate the technology implications for energy-efficient L^3Cs. Specifically, we evaluate the energy, performance, and cost of L^3Cs that are power-optimized for SRAM, STT-RAM, and eDRAM. We also show the impact of cache size, technology scaling, processor frequency, and temperature. Full-system simulations indicate that with the proposed refresh-reduction method based on a low-cost dynamic dead-line prediction scheme, eDRAM becomes a viable alternative for energy-efficient L^3C architectures.

The remainder of this paper is organized as follows. Section 2 provides an overview of memory technologies for on-die caches. Section 3 summarizes refresh-reduction methods. Section 4 demonstrates the use of dead-line prediction for refresh reduction. Section 5 presents our experimental cache-modeling framework, low power L^3C implementations, and evaluation methodology. Section 6 discusses the results and analysis. Finally, Section 7 concludes this paper.

2. Memory Technologies for On-Die Caches

Memory technologies for implementing on-die LLCs include SRAM, STT-RAM, and eDRAM. These candidate technologies are fast and have high write endurance. On the other hand, memory technologies such as PCM (phase change memory), Flash, and Memristor are relatively slow and have limited endurance, making them less suitable for implementing on-die caches.

In a typical processor with a three-level cache hierarchy, latency and bandwidth are the most important design considerations for the L1 and L2 caches. Therefore these caches are usually implemented using high-performance SRAMs. L3 last-level cache designs, however, target high capacity and low power. As a result, in addition to using low-leakage SRAM to construct LLCs, STT-RAM and eDRAM are potential memory technology alternatives. Table 1 compares the technology features of SRAM, STT-RAM, and two types of eDRAM.

2.1. SRAM

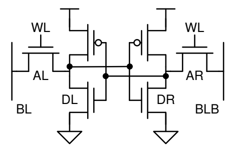
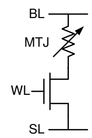
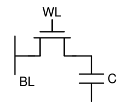
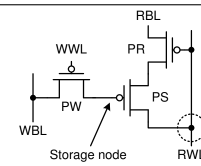
A typical 6T SRAM cell is shown in Table 1(A). When performing a read operation, the access transistors (AL and AR) are turned on, and, depending on the stored data, one of the pull-down transistors (DL or DR) creates a current path from one of the bit-lines (BL or BLB) to ground, enabling fast differential sensing operation. When performing a write operation, the cell content is written based on the bit-lines' differential voltage signal applied by the write driver.

SRAMs can be built using standard CMOS process. They also provide fast memory accesses, making them the most widely used embedded memory technology. However, due to their six-transistor implementation, SRAM cells are large in size. Subthreshold and gate leakage paths introduced by the cell structure also result in high standby power.

2.2. STT-RAM

STT-RAM is a type of magnetic RAM. An STT-RAM cell consists of a magnetic tunneling junction (MTJ) connected in series with an NMOS access transistor. A schematic of an STT-RAM cell is shown in Table 1(B), where the MTJ is denoted by the variable resistor. There are two ferromagnetic layers in the MTJ that determine the device resistance: the free layer and the reference layer. Depending on the relative magnetization directions of these two layers, the MTJ is either low-resistive (parallel) or high-resistive (anti-parallel). It is thereby used as a non-volatile storage element.

Table 1: Comparison of various memory technologies for on-die caches.

	(A) SRAM	(B) STT-RAM	eDRAM (C) 1T1C	eDRAM (D) Gain cell
Cell schematic				
Process	CMOS	CMOS + MTJ	CMOS + Cap	CMOS
Cell size (F^2)	120 - 200	6 - 50	20 - 50	60 - 100
Data storage	Latch	Magnetization	Capacitor	MOS gate
Read time	Short	Short	Short	Short
Write time	Short	Long	Short	Short
Read energy	Low	Low	Low	Low
Write energy	Low	High	Low	Low
Leakage	High	Low	Low	Low
Endurance	10^{16}	$> 10^{15}$	10^{16}	10^{16}
Retention time	-	-	< 100 us *	< 100 us *
Features	(+) Fast (-) Large area (-) Leakage	(+) Non-volatile (+) Potential to scale (-) Extra process (-) Long write time (-) High write energy (-) Poor stability	(+) Low leakage (+) Small area (-) Extra process (-) Destructive read (-) Refresh	(+) Low leakage (+) Decoupled read/write (-) Refresh

* 32 nm technology node

When performing a read operation, the access transistor is turned on, and a small voltage difference is applied between the bit-line (BL) and the source-line (SL) to sense the MTJ resistance. When performing a write operation, a high voltage difference is applied between BL and SL. The polarity of the BL-SL cross voltage is determined by the desired data to be written. Long write pulse duration and high write current amplitude are required to reverse and retain the direction of the free layer.

Though we usually refer STT-RAM as a non-volatile technology, it is also possible to trade its non-volatility (i.e., its retention time) for better write performance [39]. The retention time can be modeled as

$$t = t_0 \times e^{\Delta} \tag{1}$$

where t is the retention time, t_0 is the thermal attempt frequency, and Δ is the thermal barrier that represents the thermal stability [12]. Δ can be characterized using

$$\Delta \propto \frac{M_s H_k V}{k_B T} \tag{2}$$

where M_s is the saturation magnetization, H_k is the anisotropy field, V is the volume, k_B is the Boltzmann constant, and T is the absolute temperature. A smaller Δ allows a shorter write pulse width or a lower write current, but it also increases the probability of random STT-RAM bit-flip.

2.3. eDRAM

There are two common types of eDRAM: the 1T1C eDRAM and the gain cell eDRAM. Both of them utilize some form of capacitor to store the data. For instance, a 1T1C cell utilizes a dedicated capacitor to store its data, while a gain cell relies on the gate capacitance of its storage transistor. In this paper, we use gain cell as the basis of our eDRAM study.

Since the stored charge gradually leaks away, refresh is required for eDRAMs to prevent data loss. The refresh rate of an eDRAM circuit is determined by its data-retention time, which depends on the rate of cell leakage and the size of storage capacitance (see Table 1 for typical retention times).

2.3.1. 1T1C eDRAM

A 1T1C eDRAM cell consists of an access transistor and a capacitor (C), as shown in Table 1(C). 1T1C cells are denser than gain cells, but they require additional process steps to fabricate the cell capacitor. A cell is read by turning on the access transistor and transferring electrical charge from the storage capacitor to the bit-line (BL). Read operation is destructive because the capacitor loses its charge while it is read. Destructive read requires data write-back to restore the lost bits. The cell is written by moving charge from BL to C.

2.3.2. Gain Cell eDRAM

Gain cell memories can be built in standard CMOS technology, typically implemented with two or three transistors [36, 19, 40, 11], providing low leakage, high density, and fast memory access. This study utilizes the boosted 3T gain cell [11] as the eDRAM cell structure due to its capability to operate at high frequency while maintaining an adequate data retention time. Table 1(D) shows the schematic of the boosted 3T PMOS eDRAM gain cell. It is comprised of a write access transistor (PW), a read access transistor (PR), and a storage transistor (PS). PMOS transistors are utilized because a PMOS device has less leakage current compared to an NMOS device of the same size. Lower leakage current enables lower standby power and longer retention time.

During write access, the write bit-line (WBL) is driven to the desired voltage level by the write driver. Additionally, the write word-line (WWL) is driven to a negative voltage to avoid threshold voltage drop such that a complete data '0' can be passed through the PMOS write access transistor from WBL to the storage node.

When performing a read operation, once the read word-line (RWL) is switched from VDD to 0V, the precharged read bit-line (RBL) is pulled down slightly if a data '0' is stored in the storage node. If a data '1' is stored in the storage node, RBL remains at the precharge voltage level. The gate-to-RWL coupling capacitance of PS enables preferential boosting: when the storage node voltage is low, PS is in inversion mode, which results in a larger coupling capacitance. On the other hand, when the storage node voltage is high, PS is in weak-inversion mode, which results in a smaller coupling capacitance. Therefore, when RWL switches from VDD to 0V, a low storage node voltage is coupled down more than a high storage node voltage. The signal difference between data '0' and data '1' during a read operation is thus amplified through preferential boosting. This allows the storage node voltage to decay further before a refresh is needed, which effectively translates to a longer data retention time and better read performance.

3. Refresh Management

Refresh is required for eDRAM-based caches; unfortunately, this creates negative impacts on both performance and power. For instance, the cache bandwidth is degraded by refresh activity because normal cache accesses are stalled while the cache is being refreshed. This problem can be alleviated by organizing a cache into multiple sub-banks, allowing refresh operations and normal cache accesses to happen concurrently [27].

There are several methods to mitigate refresh penalties. They can be classified into two categories:

1. **Reducing the refresh rate by exploiting process and temperature variations.** Process and temperature variations affect the retention time of a DRAM cell. Traditionally, the refresh rate is determined by the weakest DRAM cells, i.e., those cells that have the shortest data-retention time. However, conservatively performing refresh operations based on the shortest retention time introduces significant refresh overhead. One promising approach for reducing refresh is to utilize retention-time variation and to decrease the refresh rates for blocks or rows that exhibit longer retention time [33, 25, 7, 32]. This approach requires an initial time period to characterize the retention time of each individual memory block and store the retention time information in a table.

   Another promising approach is to utilize error-correcting codes (ECC) to dynamically detect and correct bits that fail [15, 41, 43]. This approach reduces the refresh rate by disassociating failure rate from single effects of the weakest cells.
2. **Reducing the number of refresh operations by exploiting memory-access behaviors.** Reohr [37] presents several approaches for refreshing eDRAM-based caches, including periodic refresh, line-level refresh based on time stamps, and no-refresh. For instance, Liang et al. [30] showed that by adopting the line-level refresh or the no-refresh approaches with intelligent cache-replacement policies, 3T1D (three transistors one diode) eDRAM becomes a potential substitute for SRAM in the context of L1 data caches.

   The periodic refresh policy does a sweep of the cache such that all cache lines are refreshed periodically, similar to the refresh mechanism used in regular DRAM main memories. It introduces the least logic and storage overhead but provides no opportunity to reduce the number of refresh operations.

   The line-level refresh policy utilizes line-level counters to track the refresh status of each cache line. This policy is analogous to *Smart Refresh* [16], a refresh-management framework for DRAM main memories. When a line is refreshed, its counter resets to zero. There are two types of refreshes: the *implicit* refresh and the *explicit* refresh. An implicit refresh happens when the line is read, written, or loaded; an explicit refresh happens when the line-level counter signals a refresh to the data array. Therefore, if two accesses to the same cache line occur within a refresh period, the cache line is implicitly refreshed and no explicit refresh is needed.

   The no-refresh policy never refreshes the cache lines. Similar to the line-level refresh implementation, each cache line has a counter that tracks the time after an implicit refresh. When the counter reaches the retention time, the line is marked as invalid. As a result, the no-refresh policy removes refresh power completely but potentially introduces more cache misses.

Our refresh reduction method falls into the second category: we attempt to identify dead lines using a low-cost dynamic dead-line prediction scheme and, thereby, to eliminate refreshes to these lines. To our knowledge, this is the first work that uses dead-line prediction to reduce the refresh power of eDRAM-based caches. Our design can also be applied on top of variable retention time architectures or error-correcting systems.

4. Dead-Line Prediction vs. Refresh

Refresh is the main source of eDRAM-based L^3C power dissipation [41]. Previous studies have shown the effectiveness of using dead-line prediction to reduce the leakage power of SRAM-based L1 or L2 caches. However, to the best of our knowledge, no previous study has demonstrated dead-line prediction in the context of eDRAM caches. This work proposes a refresh-reduction method for eDRAM L^3Cs using a

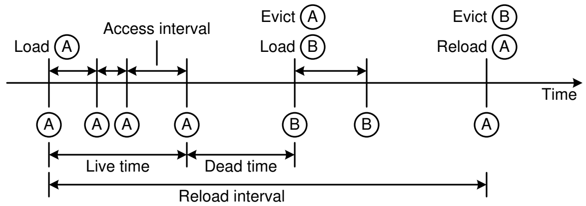


Figure 2: Generational behavior of a cache line. The generation of Ⓐ begins when it is loaded and ends when it is evicted.

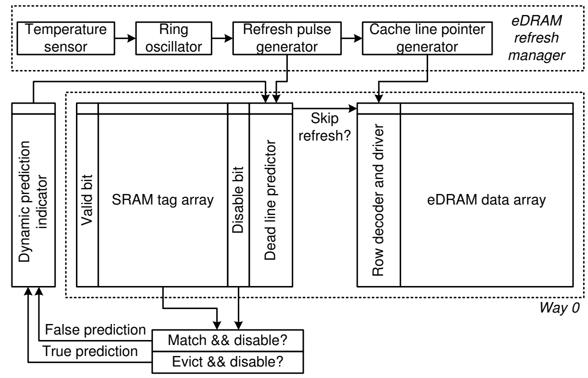


Figure 3: Proposed eDRAM cache architecture with dynamic dead-line prediction.

low-cost dynamic dead-line prediction scheme. If a cache line is predicted dead, refresh to the line is skipped, thereby minimizing refresh energy. Additionally, using dead-line prediction to reduce standby power is a more natural fit for eDRAM than SRAM. For instance, unlike SRAM-based cache lines, re-enabling eDRAM-based lines does not require wake-up time. Moreover, since hardware such as a ring oscillator and refresh pulse generator are already part of the eDRAM refresh controller, we can reuse them to support time-based dead-line predictors.

To demonstrate a limit case of refresh power savings that one can achieve by eliminating refreshes to dead lines, we characterize the average dead time of a cache line in a 32MB LLC. Based on the workloads considered, a cache line is dead 59% of the time, indicating that significant refresh power can be saved without degrading performance.

4.1. Cache Time Durations

Dead-line prediction can be used to improve cache hit rate [18] or to reduce cache standby power [23]. It uses the concept of *cache-time durations*, which are best expressed using the generational behavior of cache lines. Each generation begins with a data insertion and ends with data eviction. A cache-line generation is partitioned into two parts. The first part, *live time*, is the time where the line is actively accessed. The second part, *dead time*, is the time where the line is awaiting eviction. Additionally, the *access interval* is the time between two successful line references, while the *reload interval* is the time between two generations of the same line. An example of the generational behavior of a cache line is depicted in Figure 2.

4.2. Dead-Line Prediction

There are several state-of-the-art approaches that use dead-line prediction to save the leakage power of L1 or L2 SRAM caches. For instance, Kaxiras et al. [23] proposed two methods to implement time-based leakage control (*Cache Decay*). The first method uses a fixed decay interval, which only requires two extra bits for each cache line to track the decay interval. The *decay interval* is the elapsed time in which a cache line has not been accessed. However, since different applications may have different decay intervals, this method does not always result in optimal standby power reduction. The second method improves upon the first one by adaptively finding the optimal decay interval for each of the cache lines. It requires six extra bits for each line: two bits used as a per-line counter to identify the best decay interval, and four bits to represent the length of the interval.

One downside is that using counters to forecast decay intervals potentially results in more false predictions [6]. For example, if the period between two consecutive hits is longer than the counter's threshold, dead-line prediction is falsely considered successful. Consequently, instead of prolonging the decay interval to correct the false prediction, the interval is decreased, making the next prediction also possibly incorrect.

Zhou et al. [44] proposed *Adaptive Mode Control*, in which a global register indicates the optimal decay interval for the entire cache. It introduces only a small storage and power overhead, but it also results in non-optimal cache performance and power because not every cache line has the same decay interval.

Abella et al. [6] proposed *IATAC*, a smart predictor to turn off L2 cache lines, which uses global predictors in addition to local predictors to improve the prediction accuracy. However, this scheme requires non-negligible overhead, making it less practical for large caches. For example, a 32MB cache requires on the order of 10% storage overhead.

In addition to time-based dead-line predictors, Lai et al. [28] proposed a trace-based dead-line predictor for L1 data caches, Khan et al. [24] proposed sampling dead-line prediction for LLCs. They demonstrate that by using dead-line prediction, prefetching can be performed more effectively, hence improving cache hit ratio.

Though a number of dead-line predictors are applicable to eDRAM L^3C refresh reduction, our design inherits the concept from time-based dead-line predictors. It is easy to implement and introduces insignificant logic and storage overhead. We show that the proposed implementation effectively reduces eDRAM L^3C refresh power.

4.3. Proposed Implementation

Figure 3 shows the proposed eDRAM cache architecture with dynamic dead-line prediction. It consists of the eDRAM refresh manager, the dynamic dead-line prediction utility, the SRAM tag array, the eDRAM data array, and

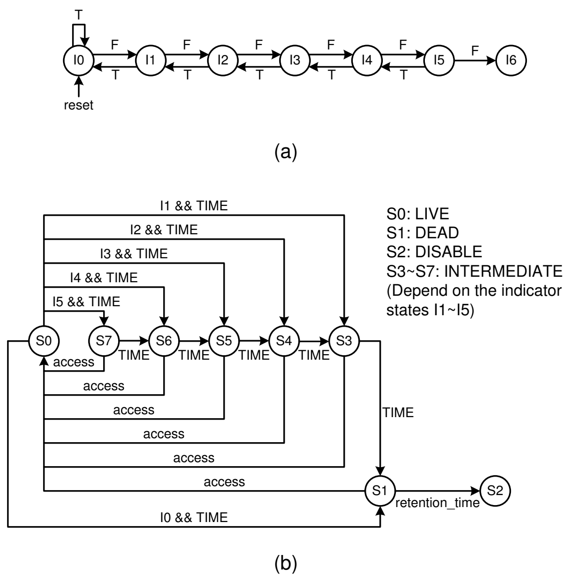


Figure 4: Proposed dynamic dead-line prediction implementation. It is comprised of the dynamic prediction indicator and the dead-line predictor. The dynamic prediction indicator determines the decay interval of the dead-line predictor. (a) State machine of the dynamic prediction indicator. When the indicator enters state I6, the associated dead-line predictors are turned off to prevent more undesired cache misses. (b) State machine of the dead-line predictor. In our implementation, TIME = 256 * retention_time.

other logic and storage necessary for caches. The temperature sensor determines the frequency of the ring oscillator: higher temperature normally results in higher frequency. This frequency then determines the rate of the refresh pulse. Additionally, each line has its time-based dead-line predictor and disable bit. The disable bit is an indicator of whether the eDRAM line holds valid data or stale data. For instance, if a line is marked as dead, its content becomes stale after a retention time period has elapsed because no refreshes were applied. Each set also has a prediction indicator, which dynamically controls the dead-line predictors based on the history of prediction.

Under normal conditions, an eDRAM cache line is periodically refreshed. However, if the associated dead-line predictor turns off the line, any refresh signal to the line is bypassed, disabling refresh. Figure 4a illustrates the state machine of the dynamic prediction indicator. A false prediction (F) indicates that the previous designated decay interval was too short, and a longer interval should be utilized instead to avoid unwanted cache misses. On the other hand, a true prediction (T) indicates that a reasonable decay interval has been reached. Finally, if many false predictions are detected, the prediction indicator switches off the dead-line predictors to prevent more undesired cache misses.

Figure 4b shows the state machine of the dead-line predictor. Anytime a line hit or a line insertion happens, the cache line returns to the $S0$ state, indicating that the line is alive.

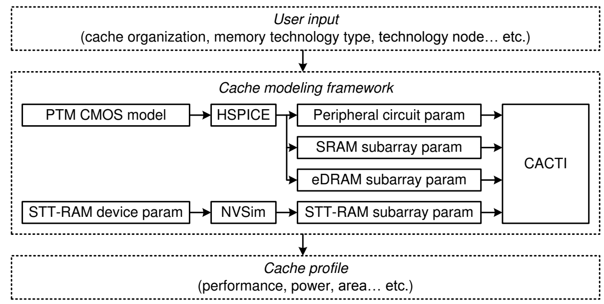


Figure 5: Cache modeling framework.

When a predetermined time period ($TIME$) has elapsed, the line transitions to one of the $S1$, $S3 \sim S7$ states, depending on the dynamic prediction indicator. For example, if the dynamic prediction indicator is in the $I1$ state, then the dead-line predictor will transition from $S0$ to $S3$ after $TIME$ has elapsed. After another $TIME$ duration, the dead-line predictor will enter $S1$, meaning that the line is predicted as dead. In other words, the line is predicted as dead if it has not been accessed for two $TIME$ durations. Additionally, since no refresh is applied to a dead line, the eDRAM cache line loses its content when its retention time expires. In this scenario, the line is written back to the main memory if it is dirty. The state of the dead-line predictor also transitions from $S1$ to $S2$. $S2$ represents a disabled line, meaning that any access to the line results in a cache miss.

The proposed dynamic dead-line prediction scheme requires four additional bits per cache line (one disable bit, three predictor bits), and three additional indicator bits per cache set. For a 32MB, 16-way cache that uses 64-byte blocks, the area overhead of the logic and storage is less than 5%, and the power overhead is less than 2%. We consider these overheads to be reasonable tradeoffs.

5. Modeling and Methodology

5.1. Cache Modeling

Our SRAM, STT-RAM, and eDRAM cache models build on top of CACTI [1], an analytical model that estimates the performance, power, and area of caches. For the peripheral circuitry, the SRAM array, and the gain cell eDRAM array, we first conduct circuit (HSPICE) simulations using the PTM CMOS models [3]. We then extract the circuit characteristics and integrate them into CACTI. For the STT-RAM cache modeling, we obtain STT-RAM array characteristics using NVSim [13] and integrate them into CACTI. The STT-RAM device parameters are projected according to [14, 20]. Currently, CACTI only models leakage power as being temperature dependent. We extend CACTI to model the effects of temperature on dynamic power, refresh power, and performance (access time, cycle time, retention time). Figure 5 shows our cache-modeling framework.

As a case study, we evaluate a high-capacity gain cell eDRAM cache against SRAM and STT-RAM equivalents (see Table 2). The high-capacity cache is a 32nm, 32MB,

Table 2: Detailed characteristics of 32 MB cache designs built with various memory technologies.

	SRAM	STT-RAM	Gain cell eDRAM
Read latency	4.45 ns	3.06 ns	4.29 ns
Write latency	4.45 ns	25.45 ns	4.29 ns
Retention time	-	1 s	20 us
Read energy	2.10 nJ/access	0.94 nJ/access	1.74 nJ/access
Write energy	2.21 nJ/access	20.25 nJ/access	1.79 nJ/access
Leakage power	131.58 mW/bank	45.28 mW/bank	49.01 mW/bank
Refresh power	0 mW	0 mW	600.41 mW
Area	80.41 mm^2	16.39 mm^2	37.38 mm^2

Temperature = 75^{o}C

Table 3: Baseline system configuration.

Processor	8-core, 2 GHz, out-of-order, 4-wide issue width
L1I (private)	32 KB, 8-way set associative, 64 B line size, 1 bank, MESI cache
L1D (private)	32 KB, 8-way set associative, 64 B line size, 1 bank, MESI cache
L2 (private)	256 KB, 8-way set associative, 64 B line size, 1 bank, MESI cache
L3 (shared)	32 MB, 16-way set associative, 64 B line size, 16 banks, write-back cache
Main memory	8 GB, 1 channel, 4 ranks/channel, 8 banks/rank

16-way cache that is partitioned into 16 banks and uses 64-byte blocks. Additionally, the cache tag and data are sequentially accessed (i.e., data array access is skipped on a tag mismatch). By skipping the data array access on a tag mismatch, a sequentially accessed cache saves dynamic power. The cache is also pipelined such that it exhibits a reasonable cycle time.

For the peripheral and global circuitry, high performance CMOS transistors are utilized. Low leakage SRAM cells are used for the data array of the SRAM cache and the tag arrays of the SRAM, STT-RAM, eDRAM caches. Unlike storage-class STT-RAM implementations that have retention times more than 10 years, the STT-RAM device presented in Table 2 has only 1 second retention time, which requires lower write current. These parameters are also used for the L3 cache in our full-system evaluation framework.

As shown in Table 2, since the interconnections play a dominant role in access time and access energy for high-capacity caches, the STT-RAM and the eDRAM caches have shorter read latencies and lower read energies compared to the SRAM cache. This is due to their smaller cell sizes and shorter wires. In particular, the STT-RAM cache has the smallest cell size and correspondingly best read performance. However, although its retention time is sacrificed for better write performance, the STT-RAM cache still has the highest write latency and energy. Finally, when comparing standby power, SRAM is the leakiest technology among the three memory designs. Both STT-RAM and eDRAM dissipate low leakage power, but the eDRAM cache suffers short retention time and high refresh power.

5.2. Low Power L^{3}C Implementations

As mentioned earlier, before comparing the three technologies, we optimize each for improved energy consumption.

5.2.1. SRAM L^{3}C

The low power SRAM L^{3}C is optimized for leakage power at various levels. At the device level, we use a low-leakage CMOS process to implement the SRAM cells. At the circuit level, we apply power gating at the line granularity. Finally, we use the proposed dynamic dead-line prediction at the architecture level: a cache line is put into sleep mode (low power mode) via power gating if it is predicted dead.

5.2.2. STT-RAM L^{3}C

The low power STT-RAM L^{3}C is optimized for write energy using the STT-RAM device optimization methodology presented in [39]. As described in Section 2.2, we can reduce the write energy by sacrificing the STT-RAM's data-retention time. Based on the average live time of a cache line in a 32MB LLC, we set the STT-RAM retention time to 1 second and further optimize the write energy according to this target retention time [20]. We did not consider STT-RAM L^{3}Cs that further reduces retention time to obtain even lower write energy consumption, because these STT-RAMs require additional buffers [20] and scrubbing mechanisms to retain cache reliability.

5.2.3. eDRAM L^{3}C

The low power eDRAM L^{3}C is optimized for refresh power using our proposed refresh reduction method: if a cache line is predicted dead, its refresh signals are skipped to save refresh power, as described in Section 4.

5.3. Baseline Configuration

Our study uses MARSS [35], a full-system simulator of multi-core x86 CPUs. MARSS is based on QEMU [9], a dynamic binary-translation system for emulating processor architectures, and PTLsim [42], a cycle-accurate x86 microarchitecture simulator. QEMU also emulates IO devices and chipsets, allowing it to boot unmodified operating systems (e.g., Linux). When simulating a program, MARSS switches from emulation mode to detailed simulation mode once the region of interest is reached.

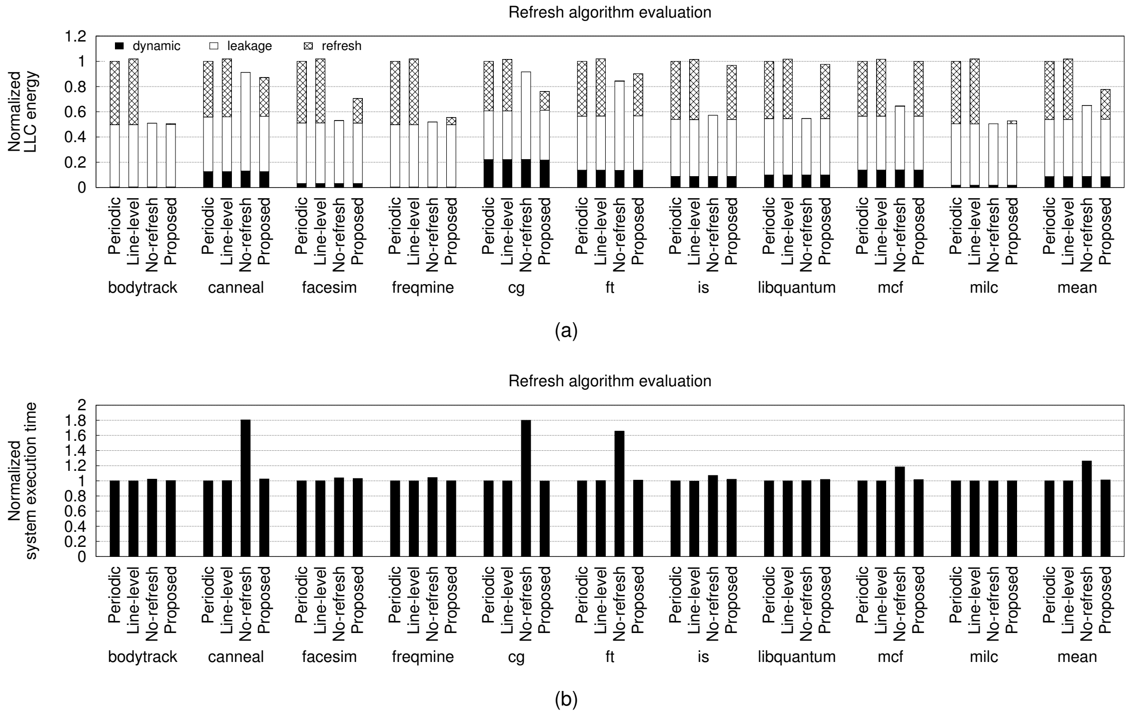


Figure 6: Refresh algorithm evaluation. (a) LLC energy breakdown normalized to periodic refresh. (b) System execution time normalized to periodic refresh. The proposed refresh algorithm effectively reduces the refresh energy with negligible performance loss.

We integrate a refresh controller into MARSS and augment the cache models with the necessary counters and statistical utilities to support the low power techniques described in Section 5.2. In addition to the parameterized cache access time, we expand MARSS with parameterized cache cycle time and refresh period. We also modify MARSS to support asymmetric cache read, write, and tag latencies. This property is required to evaluate STT-RAM caches accurately.

The baseline configuration is an 8-core, out-of-order system that operates at 2GHz, with L1 and L2 private caches, and a 32MB shared last-level L3 cache. The L1 caches are implemented using multi-port (2-read/2-write) high performance SRAMs, while the L2 caches are built with single-port high performance SRAMs. A pseudo-LRU replacement policy [8] is used for the caches. Additionally, DRAMSim2 [38], a cycle-accurate DRAM simulator, is utilized for the main memory model, integrated with MARSS. The 8GB main memory is configured as 1 channel, 4 ranks per channel, and 8 banks per rank, using Micron's DDR3 2Gb device parameters [5]. Table 3 summarizes our system configuration.

The power and performance parameters for the caches are extracted from our enhanced CACTI model. We also use HSPICE simulation based on the PTM CMOS models to calculate the power of the additional storage and logic.

5.4. Workloads

We use multi-threaded and multi-programmed workloads from the PARSEC 2.1 benchmark suite [10], the NAS parallel benchmark suite (NPB 3.3.1) [2], and the SPEC CPU 2006 benchmark suite [4] to evaluate our system. The multi-threaded workloads (PARSEC and NPB) are configured as single-process, eight-thread workloads, while the multi-programmed workloads (SPEC) are constructed using eight identical copies of single-threaded benchmarks. We use the input sets *simmedium*, *CLASS A*, *ref* for the PARSEC, NPB, SPEC benchmarks, respectively. All workloads run on top of Ubuntu 9.04 (Linux 2.6.31), executing 2.4 billion instructions in detailed simulation mode, starting at the region of interest.

6. Results and Analysis

In this section, first we demonstrate the effectiveness of the proposed refresh-reduction method by comparing it with existing refresh algorithms. Then, we evaluate L^3Cs built with SRAM, STT-RAM, and eDRAM. The evaluation includes the study of LLC energy breakdown (where the length of execution time plays a role), system performance, and die cost. We also explore the impact of LLC size, technology scaling, frequency scaling, and temperature.

6.1. Refresh Algorithm Evaluation

Figure 6 compares the eDRAM LLC energy breakdown and system execution time when using various refresh algorithms, including periodic refresh, line-level refresh (Smart Refresh), no-refresh, and the proposed refresh mechanism based on dead-line prediction. We summarize the results as follows:

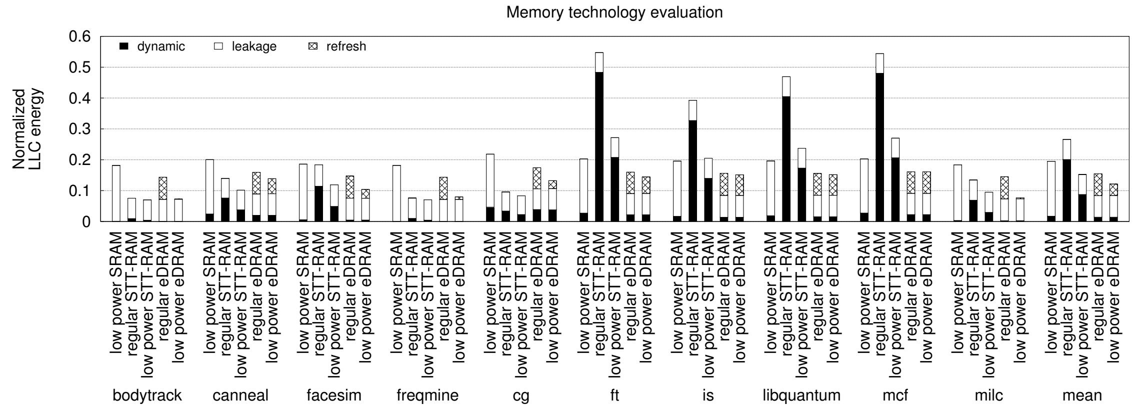


Figure 7: Normalized LLC energy breakdown with respect to various memory technologies. The results are normalized to regular SRAM (not shown). Note that regular SRAM dissipates significantly higher leakage power.

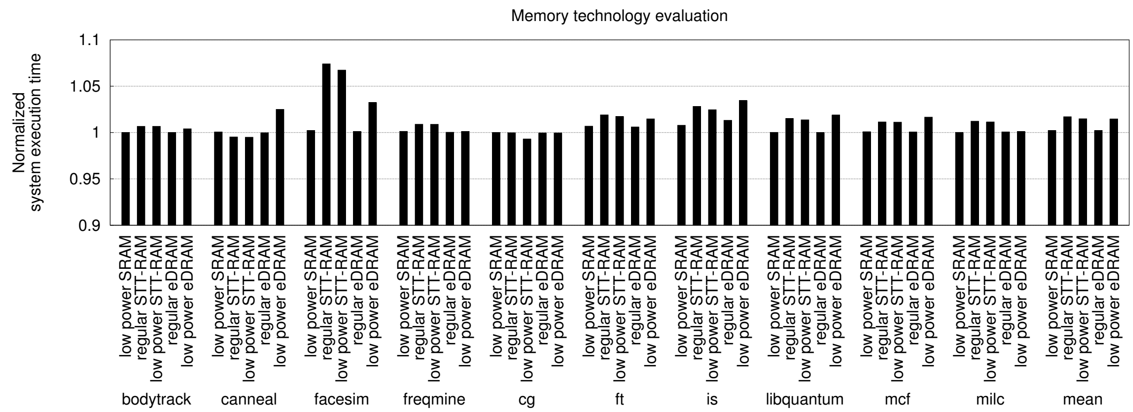


Figure 8: Normalized system execution time with respect to various memory technologies. The results are normalized to regular SRAM (not shown).

- In contrast to utilizing line-level refresh for L1 caches or Smart Refresh for commodity DRAM main memories, applying line-level refresh to LLCs results in slightly higher energy usage compared to the baseline periodic refresh. This is because line-level refresh only improves refresh under the condition that the retention time of each line is much longer than the line access interval. Line-level refresh also shortens the refresh period to accommodate the worst-case scenario in which all lines in a subarray reach the refresh threshold simultaneously. As a result, since the LLC is not as intensively accessed as the L1 caches, and the data retention time of eDRAMs is much shorter than the retention time of commodity DRAMs, line-level refresh is unlikely to reduce the number of refresh operations.
- Similar to line-level refresh, no-refresh has little opportunity to take advantage of implicit refresh. Consequently, most cache lines become invalid before they are re-referenced. Therefore, although no-refresh results in the least LLC energy consumption, it degrades the system performance by 26% on average. It also results in significantly more system energy consumption due to longer execution time and higher main memory activity for servicing the additional cache misses.
- Our proposed refresh scheme reduces refresh power significantly for benchmarks such as *bodytrack*, *facesim*, *freqmine*, *cg*, and *milc*. Based on the workloads considered, the proposed scheme reduces refresh energy by 48%, and reduces LLC energy by 22% with only 1.2% longer execution time compared to periodic refresh.

6.2. Memory Technology Evaluation

6.2.1. LLC Energy Breakdown

Figure 7 shows the normalized energy breakdown of LLCs based on SRAM, STT-RAM, and eDRAM. For each memory technology, we include the results before power-optimization and the results after applying low power techniques. For instance, 'regular' SRAM uses high performance transistors to implement the entire cache with no power gating; 'regular' STT-RAM uses storage-class STT-RAM technology, which has long retention time but requires high write energy; and 'regular' eDRAM uses the conventional periodic refresh policy. On the other hand, low power SRAM, STT-RAM, and eDRAM LLCs represent the designs described in Section 5.2. The results are summarized as follows:

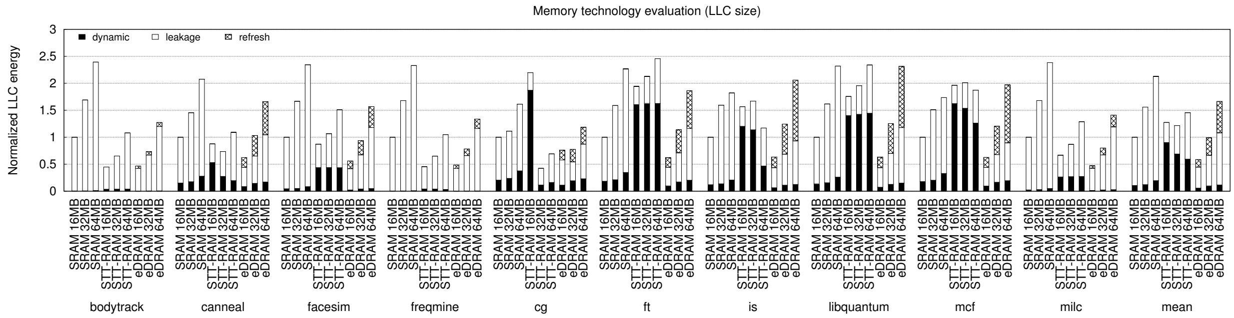


Figure 9: Normalized LLC energy breakdown with respect to different LLC sizes.

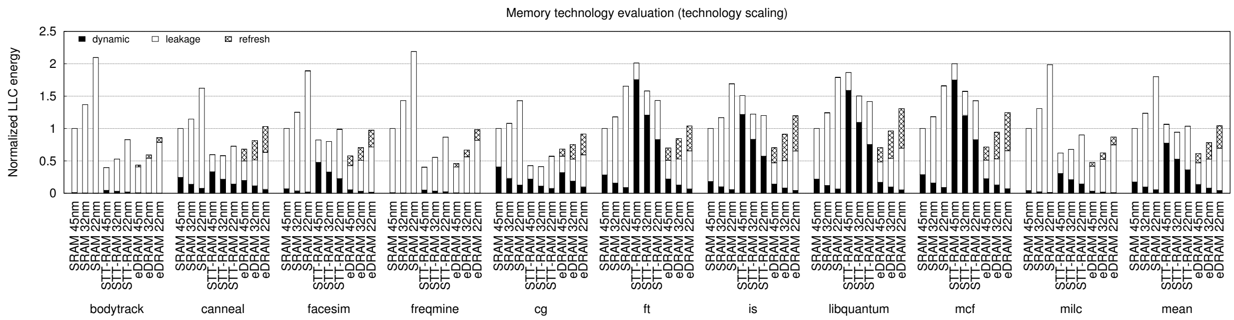


Figure 10: Normalized LLC energy breakdown with respect to various technology nodes.

- Low power LLC implementations consume much less energy compared to regular implementations. For instance, low power SRAM consumes 81% less energy than regular SRAM, and low power STT-RAM uses 48% less energy than regular STT-RAM. As a result, we show that ignoring implementation details potentially leads to wrong conclusions.
- Low power STT-RAM consumes the least energy for benchmarks that have few writes (*bodtrack*, *canneal*, *freqmine*, *cg*). However, if there are many write operations from the CPU or from main memory fetches, STT-RAM uses the most energy (*ft*, *is*, *libquantum*, *mcf*).
- For write intensive workloads, eDRAM results in the most energy-efficient LLC implementation. For workloads with low write intensity, the energy consumption of eDRAM approaches that of STT-RAM when our proposed refresh reduction method is used. Based on the workloads considered, we show that low power eDRAM reduces the LLC energy by 36% compared to low power SRAM, and reduces the LLC energy by 17% compared to low power STT-RAM.

6.2.2. System Performance

Figure 8 illustrates the normalized system execution time with respect to LLCs based on various memory implementations. We show the following:

- Regular SRAM has the best system performance on average. However, since the low power implementations also use high performance transistors for the peripheral circuitry, they are not much slower than regular implementations.
- Low power STT-RAM performs the best for read intensive workloads (*canneal* and *cg*) because it has the shortest read latency. However, although low power STT-RAM has better write performance than its regular (unoptimized) counterpart, it still performs worse overall than SRAM and eDRAM on average.

6.2.3. The Impact of Cache Size

Figure 9 shows the normalized LLC energy breakdown with respect to different LLC sizes. Key observations:

- Increasing the LLC size potentially results in lower cache miss ratio and thus shorter system execution time and fewer updates from the main memory. As a result, there are cases where a large LLC consumes less energy than a small LLC. For example, since a 16MB LLC is not large enough to hold the working set of the *cg* benchmark, the LLC is frequently updated by the main memory. Therefore, if the 16MB LLC is built with STT-RAM where write energy is high, the dynamic energy becomes significant due to the large number of cache updates.
- The leakage and refresh power increase with the size of the LLC. In particular, for the 64MB case, eDRAM consumes more energy than STT-RAM due to refresh. However, since we use the same refresh and dead-line prediction implementation for eDRAM LLCs of all sizes, the proposed refresh reduction method is not optimized for all cases.

6.2.4. The Impact of Technology Scaling

Figure 10 illustrates the normalized LLC energy breakdown with respect to various technology nodes. The results:

- As technology scales down, caches consume less active energy, but the leakage and refresh power both increase significantly. The relative dominance of active and

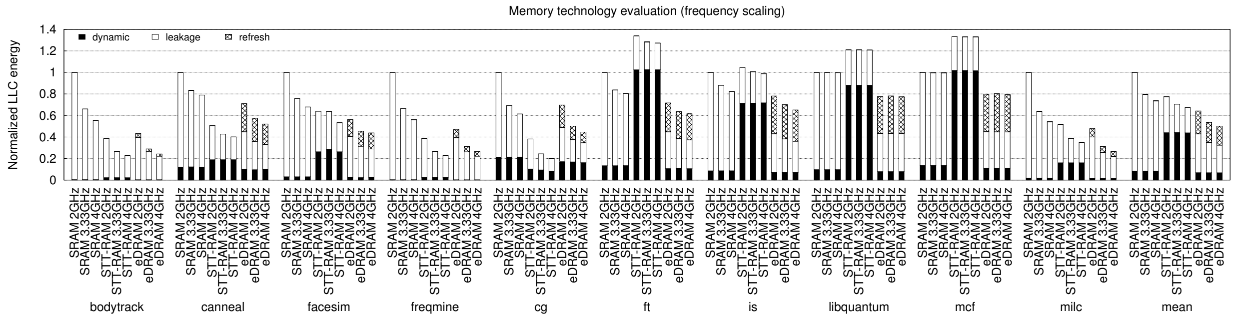


Figure 11: Normalized LLC energy breakdown with respect to various processor frequencies.

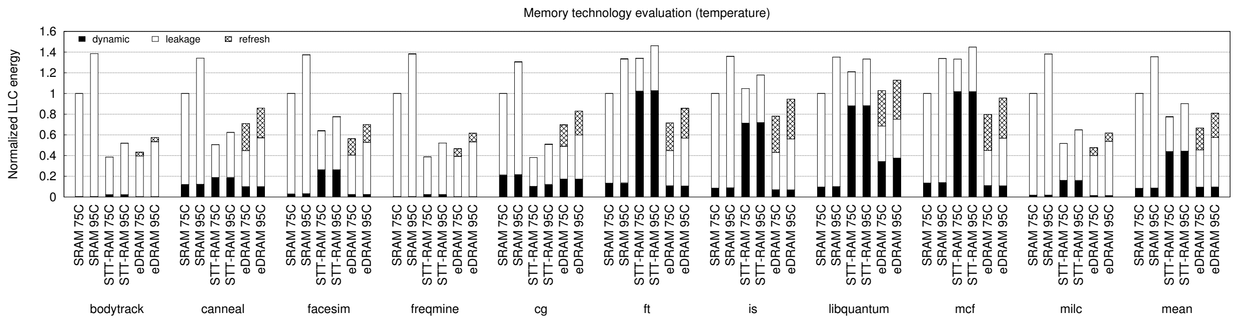


Figure 12: Normalized LLC energy breakdown with respect to different temperatures.

standby (leakage, refresh) power is an important indicator of which memory technology is a better candidate. For instance, in the 45nm technology node, STT-RAM uses the most energy, while eDRAM consumes the least. However, in the 22nm technology node, STT-RAM becomes the most energy-efficient choice due to its low standby power feature, while SRAM consumes the most energy.

- It is worth noting that, though STT-RAM is projected to perform better in the 22nm technology node, its thermal stability (reliability) also degrades significantly [22]. Researchers continue to improve the data retention and write current scaling for STT-RAM to extend its scalability [26, 34].

6.2.5. The Impact of Frequency Scaling

We also study the impact of processor frequency, shown in Figure 11. Since a high-frequency processor typically completes jobs faster than a low-frequency processor, the energy usage due to standby power is lower for the high-frequency processor. Therefore, SRAM and eDRAM appear to be more energy-efficient when running at high speed. For instance, at a 2GHz clock frequency, eDRAM uses 17% less energy than STT-RAM, whereas at a 4GHz clock frequency, the percentage of energy reduction increases to 25%.

6.2.6. The Impact of Temperature

Figure 12 shows the impact of temperature on LLC energy consumption. We summarize our observations as follows:

- Although high temperature negatively affects the dynamic, leakage, and refresh power, we show that standby power is more sensitive to temperature variation than dynamic power. As a result, at 95^{o}C, the energy gap between eDRAM and STT-RAM becomes smaller.

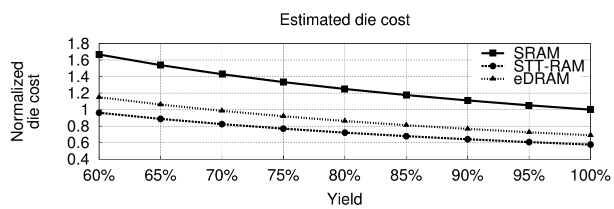


Figure 13: Estimated die cost normalized to an 8-core processor using SRAM LLC with 100% yield.

- However, the thermal stability, and thus retention time, of STT-RAM also becomes worse when temperature increases. For instance, when increasing the temperature from 75^{o}C to 95^{o}C, the average STT-RAM retention time decreases from 1 second to 0.33 second. Therefore, at high temperature, STT-RAM either requires higher write energy to prolong its retention time or requires scrubbing mechanisms to detect and correct failed bits regularly.

6.2.7. Die Cost

We estimate the die cost using

$$die\_cost = \frac{wafer\_cost}{\frac{wafer\_area}{die\_area} \times yield} \quad (3)$$

where $\frac{wafer\_area}{die\_area}$ represents the number of dies per wafer, $yield$ refers to the percentage of good dies on a wafer. We project the die area (die_area) based on the chip layout of Power7. We also assume STT-RAM introduces 5% more wafer cost due to additional fabrication processes. Figure 13 compares the die cost of an 8-core processor using either SRAM, STT-RAM, or eDRAM as its 32 MB last-level L3 cache.

7. Conclusions

Embedded DRAM, exhibiting both high density and low leakage, is a potential replacement for SRAM in the context of L^3C. However, as future processors are expected to have larger LLCs implemented using smaller technologies, refresh becomes the main source of energy consumption. In this paper, we classify eDRAM refresh reduction methods into two categories and show that, by applying a low-cost dynamic dead-line prediction scheme, refresh power is greatly reduced. Furthermore, we model SRAM, STT-RAM, eDRAM caches in detail and make an impartial comparison by applying low power techniques to each of the memory technologies. Full-system simulation results indicate that if refresh is effectively controlled, eDRAM-based L^3C becomes a viable, energy-efficient alternative for multi-core processors.

Acknowledgments

The authors would like to thank Professor Sudhanva Gurumurthi and the reviewers for their valuable inputs. This research was funded in part by Intel Labs University Research Office, the United States Department of Energy, Sandia National Laboratories, and the United States Department of Defense.

References

[1] "CACTI 6.5," http://www.hpl.hp.com/research/cacti/.

[2] "NAS Parallel Benchmarks," http://www.nas.nasa.gov/Resources/Software/npb.html.

[3] "Predictive Technology Model," http://ptm.asu.edu/.

[4] "SPEC CPU 2006," http://www.spec.org/cpu2006/.

[5] "Micron DDR3 SDRAM," http://micron.com/document_download/?documentId=424, 2010.

[6] J. Abella *et al.*, "IATAC: A Smart Predictor to Turn-Off L2 Cache Lines," *ACM Trans. Archit. Code Optim.*, vol. 2, no. 1, pp. 55–77, Mar. 2005.

[7] J. H. Ahn *et al.*, "Adaptive Self Refresh Scheme for Battery Operated High-Density Mobile DRAM Applications," in *ASSCC*, 2006.

[8] H. Al-Zoubi, A. Milenkovic, and M. Milenkovic, "Performance Evaluation of Cache Replacement Policies for the SPEC CPU2000 Benchmark Suite," in *ACMSE*, 2004.

[9] F. Bellard, "QEMU, a Fast and Portable Dynamic Translator," in *ATEC*, 2005.

[10] C. Bienia, "Benchmarking Modern Multiprocessors," Ph.D. dissertation, Princeton University, Jan. 2011.

[11] K. C. Chun *et al.*, "A 3T Gain Cell Embedded DRAM Utilizing Preferential Boosting for High Density and Low Power On-Die Caches," *IEEE J. Solid-State Circuits*, vol. 46, no. 6, pp. 1495–1505, Jun. 2011.

[12] Z. Diao *et al.*, "Spin-Transfer Torque Switching in Magnetic Tunnel Junctions and Spin-Transfer Torque Random Access Memory," *J. Physics: Condensed Matter*, vol. 19, no. 16, 2007.

[13] X. Dong *et al.*, "NVSim: A Circuit-Level Performance, Energy, and Area Model for Emerging Nonvolatile Memory," *IEEE Trans. CAD*, vol. 31, no. 7, pp. 994–1007, July 2012.

[14] A. Driskill-Smith *et al.*, "Latest Advances and Roadmap for In-Plane and Perpendicular STT-RAM," in *IMW*, 2011.

[15] P. G. Emma, W. R. Reohr, and M. Meterelliyoz, "Rethinking Refresh: Increasing Availability and Reducing Power in DRAM for Cache Applications," *IEEE Micro*, vol. 28, no. 6, pp. 47–56, Nov.–Dec. 2008.

[16] M. Ghosh and H. H. Lee, "Smart Refresh: An Enhanced Memory Controller Design for Reducing Energy in Conventional and 3D Die-Stacked DRAMs," in *MICRO*, 2007.

[17] L. Hsu *et al.*, "Exploring the Cache Design Space for Large Scale CMPs," *SIGARCH Comput. Archit. News*, vol. 33, no. 4, pp. 24–33, Nov. 2005.

[18] Z. Hu, S. Kaxiras, and M. Martonosi, "Timekeeping in the Memory System: Predicting and Optimizing Memory Behavior," in *ISCA*, 2002.

[19] N. Ikeda *et al.*, "A Novel Logic Compatible Gain Cell with Two Transistors and One Capacitor," in *Symp. VLSI Technology*, 2000.

[20] A. Jog *et al.*, "Cache Revive: Architecting Volatile STT-RAM Caches for Enhanced Performance in CMPs," in *DAC*, 2012.

[21] R. Kalla *et al.*, "Power7: IBM's Next-Generation Server Processor," *IEEE Micro*, vol. 30, no. 2, pp. 7–15, Mar.–Apr. 2010.

[22] T. Kawahara, "Scalable Spin-Transfer Torque RAM Technology for Normally-Off Computing," *IEEE Design & Test of Computers*, vol. 28, pp. 52–63, 2011.

[23] S. Kaxiras, Z. Hu, and M. Martonosi, "Cache Decay: Exploiting Generational Behavior to Reduce Cache Leakage Power," in *ISCA*, 2001.

[24] S. Khan, Y. Tian, and D. Jimenez, "Sampling Dead Block Prediction for Last-Level Caches," in *MICRO*, 2010.

[25] J. Kim and M. C. Papaefthymiou, "Block-Based Multiperiod Dynamic Memory Design for Low Data-Retention Power," *IEEE Trans. VLSI*, vol. 11, no. 6, pp. 1006–1018, Dec. 2003.

[26] W. Kim *et al.*, "Extended Scalability of Perpendicular STT-MRAM Towards Sub-20nm MTJ Node," in *IEDM*, 2011.

[27] T. Kirihata *et al.*, "An 800-MHz Embedded DRAM with a Concurrent Refresh Mode," *IEEE J. Solid-State Circuits*, vol. 40, no. 6, pp. 1377–1387, Jun. 2005.

[28] A.-C. Lai and B. Falsafi, "Selective, Accurate, and Timely Self-Invalidation using Last-Touch Prediction," in *ISCA*, 2000.

[29] H. Li *et al.*, "Performance, Power, and Reliability Tradeoffs of STT-RAM Cell Subject to Architecture-Level Requirement," *IEEE Trans. Magnetics*, vol. 47, no. 10, pp. 2356–2359, Oct. 2011.

[30] X. Liang *et al.*, "Process Variation Tolerant 3T1D-Based Cache Architectures," in *MICRO*, 2007.

[31] J. Lin *et al.*, "Understanding the Memory Behavior of Emerging Multi-core Workloads," in *ISPDC*, 2009.

[32] J. Liu *et al.*, "RAIDR: Retention-Aware Intelligent DRAM Refresh," in *ISCA*, 2012.

[33] T. Ohsawa, K. Kai, and K. Murakami, "Optimizing the DRAM Refresh Count for Merged DRAM/logic LSIs," in *ISLPED*, 1998.

[34] J. Park *et al.*, "Enhancement of Data Retention and Write Current Scaling for Sub-20nm STT-MRAM by Utilizing Dual Interfaces for Perpendicular Magnetic Anisotropy," in *VLSI Technology*, 2011.

[35] A. Patel *et al.*, "MARSSx86: A Full System Simulator for x86 CPUs," in *DAC*, 2011.

[36] W. Regitz and J. Karp, "A Three Transistor-Cell, 1024-bit, 500ns MOS RAM," in *ISSCC*, 1970.

[37] W. R. Reohr, "Memories: Exploiting Them and Developing Them," in *SOCC*, 2006.

[38] P. Rosenfeld, E. Cooper-Balis, and B. Jacob, "DRAMSim2: A Cycle Accurate Memory System Simulator," *IEEE Computer Architecture Letters*, vol. 10, no. 1, pp. 16–19, Jan. 2011.

[39] C. W. Smullen *et al.*, "Relaxing Non-Volatility for Fast and Energy-Efficient STT-RAM Caches," in *HPCA*, 2011.

[40] D. Somasekhar *et al.*, "2GHz 2Mb 2T Gain Cell Memory Macro With 128GBytes/sec Bandwidth in a 65nm Logic Process Technology," *IEEE J. Solid-State Circuits*, vol. 44, no. 1, pp. 174–185, Jan. 2009.

[41] C. Wilkerson *et al.*, "Reducing Cache Power with Low-Cost, Multi-Bit Error-Correcting Codes," in *ISCA*, 2010.

[42] M. T. Yourst, "PTLsim: A Cycle Accurate Full System x86-64 Microarchitectural Simulator," in *ISPASS*, 2007.

[43] W. Yun, K. Kang, and C. M. Kyung, "Thermal-Aware Energy Minimization of 3D-Stacked L3 Cache with Error Rate Limitation," in *ISCAS*, 2011.

[44] H. Zhou *et al.*, "Adaptive Mode Control: A Static-Power-Efficient Cache Design," in *PACT*, 2001.