Retention-Aware Placement in DRAM (RAPID): Software Methods for Quasi-Non-Volatile DRAM

Ravi K. Venkatesan, Stephen Herr, Eric Rotenberg
ECE Department, Center for Embedded Systems Research, North Carolina State University
{rkvenkat, sdherr, ericro}@ece.ncsu.edu

Abstract

Measurements of an off-the-shelf DRAM chip confirm that different cells retain information for different amounts of time. This result extends to DRAM rows, or pages (retention time of a page is defined as the shortest retention time among its constituent cells). Currently, a single worst-case refresh period is selected based on the page with the shortest retention time. Even with refresh optimized for room temperature, the worst page limits the safe refresh period to no longer than 500 ms. Yet, 99% and 85% of pages have retention times above 3 seconds and 10 seconds, respectively.

We propose Retention-Aware Placement in DRAM (RAPID), novel software approaches that can exploit off-the-shelf DRAMs to reduce refresh power to vanishingly small levels approaching non-volatile memory. The key idea is to favor longer-retention pages over shorter-retention pages when allocating DRAM pages. This allows selecting a single refresh period that depends on the shortest-retention page among populated pages, instead of the shortest-retention page overall. We explore three versions of RAPID and observe refresh energy savings of 83%, 93%, and 95%, relative to the best temperature-compensated refresh. RAPID with off-the-shelf DRAM also approaches the energy levels of idealized techniques that require custom DRAM support.

1. Introduction

Dynamic Random Access Memory (DRAM) is predicted to displace Static Random Access Memory (SRAM) as the primary volatile memory in next-generation mobile devices, as future applications increase memory requirements [22]. For example, Micron recently introduced DRAMs targeting mobile devices, called CellularRAM™ [3]. Nodes in sensor networks with unpredictable recharging capability, e.g., environmental energy harvesting [8,14], could also benefit from deep storage capacity, enabling uninterrupted data gathering while waiting for favorable recharging circumstances for wireless transmission.

However, DRAM refresh (needed to preserve stored information) continuously drains battery energy even in standby operation. Detailed power analysis of the prototype ITSY computer [20] shows that, in the deepest standby mode that still preserves the DRAM contents ("sleep"), DRAM consumes a third of total power and this fraction will increase with more DRAM and even lower power conservation modes.

DRAM refresh power can be reduced by exploiting variations in retention times among different memory cells, rows (pages), or arbitrary blocks of cells within a single DRAM chip. Process variations cause leakage to differ from one cell to another. Therefore, different cells retain information for longer or shorter periods of time [7]. Currently, a single worst-case refresh period (typically 64 milliseconds) is selected based on the cell with the shortest retention time. The worst-case refresh period also accounts for worst-case temperature (leakage increases with temperature). Limited support for temperature-compensated refresh (TCR) is now being deployed in some DRAMs [3]. However, TCR does not exploit retention-time variations among different cells, i.e., all cells are still treated uniformly with a single temperature-dependent refresh period, determined by the worst cell. Several researchers have recognized the opportunity presented by retention-time variations, proposing custom DRAM support for refreshing different cells or blocks of cells at different refresh rates [21,9,10,15]. However, to our knowledge, this approach has not been implemented in commodity DRAMs.

Instead of custom DRAM support, we propose *Retention-Aware Placement in DRAM* (RAPID), software methods for exploiting retention-time variations among different pages to minimize refresh power in off-the-shelf DRAMs. The key idea is to allocate longer-retention pages to application programs before allocating shorter-retention pages. This allows selecting a single refresh period that depends on the shortest-retention page among pages that are actually populated, rather than the shortest-retention page overall. Note that all pages are refreshed, as usual, but the extended refresh period (lower refresh rate) selected by RAPID is only safe for populated pages, which is sufficient for overall correctness.

0-7803-9368-6/06/$20.00/©2006 IEEE

Three versions of RAPID are explored in this paper, described below in order of increasing complexity.

- *RAPID-1*: Our retention-time measurements of a real DRAM chip reveal a handful of outlier pages with atypically short retention times, consistent with previous retention time characterizations [7]. For example, if the worst 1% of pages are not populated, then a refresh period of 3 seconds (s) is sufficient for the remaining 99% of pages at room temperature (25°C). Likewise, a 1 s refresh period is sufficient at the industry-standard worst-case temperature of 70°C. In contrast, the worst outlier page requires a 500 millisecond (ms) refresh period at room temperature and 150 ms at 70°C. RAPID-1 is a static approach in which the handful of outlier pages are discarded, i.e., never populated, yielding a very reasonable refresh period (e.g., 3 s at room temperature and 1 s at 70°C) with only a negligible reduction in DRAM capacity (e.g., 1%). Since the approach is static, the refresh period is independent of actual DRAM utilization.
- *RAPID-2*: Pages are placed into bins according to their retention times. Longer-retaining pages are allocated before shorter-retaining pages. Thus, the refresh period starts out at the highest setting corresponding to the best bin. The refresh period is decreased when no more free pages are available in the best bin because we begin populating pages in the second-best bin, and so on. The refresh period may only be increased back again if all pages in the lowest populated bin become free. Thus, once the refresh period has been decreased, increasing it again is a matter of "chance" since it depends on which pages happen to be freed by application programs. Note that outlier pages are discarded so that RAPID-2 refresh power is at least as good as RAPID-1 refresh power.
- *RAPID-3*: RAPID-3 builds on RAPID-2. When a free page becomes available in a higher bin, the contents of a populated page in a lower bin can be migrated to the newly freed page in the higher bin. Thus, RAPID-3 continuously reconsolidates data into the longest-retention pages possible. As a result, the refresh period selected by RAPID-3 more closely reflects actual DRAM utilization than RAPID-2. Migration can be naturally implemented as part of our proposed allocation/deallocation routines. A downside is extra power consumption due to migration, which must be weighed against extra refresh power savings. This hints at a deep design space, in terms of designing good criteria and timing for migrating pages.

The contributions of this paper include (1) three versions of RAPID, the first software-only techniques for exploiting retention-time variations in off-the-shelf DRAMs, (2) an interrupt-driven technique for coupling RAPID with arbitrary DRAMs in the context of commonly available refresh options, and (3) a test algorithm for determining page retention times and insights regarding its run-time and other costs.

The remainder of this paper is organized as follows. Section 2 discusses related work. Section 3 presents our test algorithm, discusses on-line/off-line testing trade-offs, and presents measured page retention times for a real DRAM chip. Section 4 presents the RAPID techniques and discusses how RAPID can be coupled with off-the-shelf DRAMs. Section 5 and Section 6 present our evaluation methodology and results, respectively. Section 7 summarizes the paper.

2. Related Work

Yanagisawa [21], Kim and Papaefthymiou [9,10], and Ohsawa et al. [15] propose custom hardware support in the DRAM chip itself, to refresh different cells at different refresh rates and thereby exploit retention-time variations among memory cells. Yanagisawa [21] classifies DRAM cells into two types – cells with long retention times and cells with short retention times. He proposes modifications to conventional counter-based refresh circuitry to refresh cells with long retention times less frequently than cells with short retention times. Kim and Papaefthymiou [9,10] propose a block-based multiperiod refresh approach, where a custom refresh period is selected for each block, i.e., group of cells. They also provide an algorithm to compute the optimal number of blocks. Ohsawa et al. [15] propose a hardware technique to exploit retention-time variations among DRAM pages, where each DRAM page is refreshed at a tailored refresh period that is a multiple of the shortest refresh period among all pages.

Our approach is unique, in that it does not require custom hardware modifications to the DRAM. The underlying reason is that a single refresh period is used at any given time. RAPID software populates longer-retention pages before shorter-retention pages, enabling a single refresh period at any given moment that correlates with the shortest-retention page among only populated pages rather than the shortest-retention page overall. A single refresh period, selected and possibly adjusted by RAPID as DRAM utilization changes, can be used to refresh contemporary off-the-shelf DRAMs for practical exploitation of retention-time variations. We also contribute a software testing method for characterizing page retention times, and discuss the factors that determine the run time, power activity, and precision of the test algorithm.

Hamamoto et al. [7] characterize retention-time variations among individual cells and provide insight into contributing factors, although no techniques to exploit

retention-time variations are proposed. Our retention time measurements corroborate those of Hamamoto et al. and extend their results by showing significant variations even at the page granularity, an important result for practical exploitation of retention time variations at the level of pages instead individual cells, since pages are the natural refresh granularity.

Murotani [13] and Pelley et al. [16] propose modifications to DRAM refresh circuitry, to automatically tune a single refresh period with temperature. Burgan [2] proposes monitoring the leakage of test cells, representative of the entire memory, for automatically detecting a suitable refresh period for the DRAM as a whole. Test cells monitor leakage of both logic "1" and logic "0". These inventions do not appear to exploit retention-time variations across cells, so the shortest-retention cell among all DRAM cells still limits the refresh period.

Hsu et al. [4] and Cho et al. [5] propose monitoring the leakage of four test cells, one for each bank, thus automatically selecting a refresh period for each bank of the DRAM. The shortest-retention cell in a bank still limits the refresh period of the bank as a whole. That is, the granularity is only slightly finer than that of the whole DRAM, and it is not clear if there will be much separation among per-bank refresh periods.

Lebeck et al. [12] exploit low-level support for transitioning DRAM chips among four power modes. Their power-aware page allocation polices attempt to maximize use of lower power modes while minimizing performance overhead of transitioning back to higher power modes. None of the modes appear to disable a chip entirely, i.e., refresh is always present and RAPID techniques can be applied to reduce refresh power.

RAPID can be classified among techniques that exploit the principle of better-than-worst-case design [1,6,18]. Due to future variation-limited technology, better-than-worst-case design has recently received significant attention in the context of microprocessors [1,6,18], for example. Variation-oriented low-power memory techniques such as RAPID complement these variation-oriented low-power microprocessor techniques.

The ITSY prototype is a source of information regarding power consumption in next-generation mobile devices [20]. On-line technology news sources forecast the displacement of SRAM in mobile devices with DRAM [22]. Micron's CellularRAM™ and Samsung's UtRAM™, which are specially tailored DRAMs for mobile devices and marketed as pseudo-SRAMs, seem to corroborate these forecasts [3][19].

3. Testing Methods

This section discusses strategies for testing page retention times and presents test results for a real DRAM chip. We use the DRAM of an embedded systems development board donated by Ubicom [17]. Ubicom develops embedded microprocessors for wireless applications, including some of the first multithreaded embedded microprocessors, such as the IP3023 with 8 hardware threads. Among other components, the board has an IP3023 microprocessor (which has on-chip 64KB instruction and 256KB data scratchpad SRAM, an on-chip SDRAM memory controller, and numerous other on-chip peripheral devices and interfaces), 16MB of synchronous DRAM (IS42S16800A SDRAM manufactured by ISSI), and Flash memory.

The ISSI DRAM supports both auto-refresh and self-refresh. When auto-refresh is enabled, the external memory controller issues auto-refresh commands at regular intervals to the DRAM. The DRAM refreshes the next row in sequence (as kept track of, by the DRAM) when it receives an auto-refresh command. The external memory controller's auto-refresh period is programmable. When self-refresh is enabled, the DRAM refreshes rows autonomously, always at the fixed self-refresh period of 64 ms. Self-refresh can only be used when the DRAM is in standby operation. Both auto-refresh and self-refresh can be disabled, a useful feature for testing page retention times.

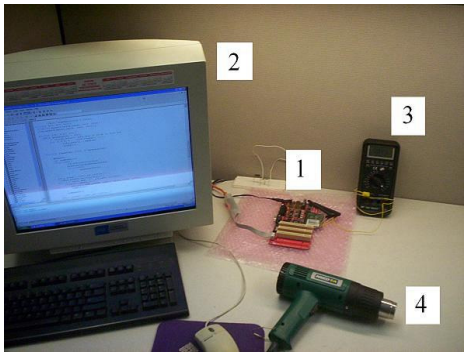

Figure 1. Experimental setup for testing page retention times.

The experimental setup for testing page retention times is pictured in Figure 1. The four parts are labeled 1-4 in the picture: (1) the Ubicom board with DRAM under test, (2) a PC workstation with Ubicom software development environment for compiling and downloading programs to the Flash memory via an ethernet connection to the board, (3) a multimeter with attached thermistor for temperature readout, and (4) a heat gun for varying ambient temperature.

3.1. Testing Algorithm

To characterize retention times of DRAM pages, auto-refresh and self-refresh are disabled in the memory controller and DRAM, respectively, so that pages are not refreshed. Testing a single page consists of writing all 1's or all 0's to the page, waiting a specified amount of time,

and reading the pattern back to see if it remains intact. If the pattern is retained for a wait time of T, the test is repeated for a wait time of T + Δt. If this next attempt fails (one or more bit errors), the retention time for the page is recorded as T (since it worked for a wait time of T but not T + Δt).

Some DRAM technologies may leak to all 0's, all 1's, or a mixture of 0's and 1's. For the ISSI DRAM, we observed that half the cells in a row decay to 1's and half the cells decay to 0's. We have not yet explored the underlying reason. Whatever the reason, we must test a page's retention of all 1's and all 0's. Ten trials of all 1's and ten trials of all 0's are performed. The minimum among the twenty trials is recorded as the retention time of the page.

We measured retention times of some pages approaching a minute at room temperature. Assuming an initial wait time T of 1 second and proceeding in increments Δt of 1 second, it takes 22 minutes (1 + 2 + … + 51 seconds) for just one trial of a page that retains information for 50 seconds. Testing pages one at a time would take far too long. Fortunately, all pages can be tested in parallel. The test program writes the entire DRAM with 1's (or 0's), waits for the next wait interval T + Δt, checks the contents of every page, and records a retention time of T for any page that retained for T but not for T + Δt. One trial is completed for all pages collectively before attempting other trials.

The time to write and check the entire ISSI DRAM is about 74 ms and 250 ms, respectively. A recorded retention time reflects only the wait time after writing and before checking, therefore, the overheads can only mean that the actual retention time for a page is slightly longer than recorded, a safe margin of error as it will ensure that the page is refreshed slightly more often than required.

It is interesting to note that the test time for a trial is insensitive to the number of pages tested in parallel. We stop testing pages whose retention times become known midway through the trial, only as a matter of power efficiency for on-line testing (more on this in the next section). Rather, test time is dominated by the long wait times.

Specifically, the test time for one trial depends directly on the wait time increment Δt, the initial wait time, and the final wait time (either the longest measured retention time or an imposed maximum wait time). The granularity Δt has a dramatic effect on test time, moreover, it is the more flexible parameter of the three. Therefore, we used a non-uniform approach. We used a granularity of 1 second for tests above 3 seconds, 100 ms for tests between 1 and 3 seconds, and 10 ms for tests below 1 second. Moreover, we first test above 3 seconds, so that only pages that do not retain above 3 seconds are considered in the finer tests (the overhead of writing and checking the entire DRAM would skew the finer tests). Likewise, we test between 1 and 3 seconds before testing below 1 second so that only residual outlier pages are tested at the finest granularity. This testing strategy allows for precise measurements of the shortest-retention pages while keeping the test time reasonable. However, these parameters can be adjusted in the interest of test time or precision, as desired. At room temperature, we found it necessary to set the maximum wait time to 50 seconds.

Table 1 summarizes the test time and the number of page writes/checks for one complete trial of the DRAM. Retention times from the first trial can be exploited to reduce the number of page writes/checks in subsequent trials. Subsequent trials can begin at the median of the distribution and spiral outward from the median, eliminating the greatest number of pages earliest in the trial. Thus, the number of page writes/checks in Table 1 is a worst-case number for the first trial. The test time is not reduced for subsequent trials because we still cycle through all the same wait times.

Table 1. One complete trial of the DRAM.

test time	22.3 minutes
# page writes/reads (worst-case – first trial)	274,000

3.2. Off-line vs. On-line Testing

Page retention times can be tested off-line by the DRAM manufacturer, off-line by the system designer, or on-line during actual use of the system.

Currently, the DRAM manufacturer only tests chips at the worst-case refresh period (e.g., 64 ms) to confirm correct operation of the chip. This basic test is much faster than our generalized test algorithm due to its single short wait time. Applying our test algorithm would increase DRAM tester time, causing an unwanted increase in price. Furthermore, conveying chip-specific page retention times to DRAM customers poses unusual logistics problems – a small Flash table coupled with the DRAM in a multi-chip package or a web-accessible database are improbable.

Alternatively, system designers (e.g., cell phone designers, sensor network researchers) could run the tests off-line for their particular applications. Since extreme energy efficiency is needed for these applications, this form of off-line testing is potentially justifiable from a business standpoint.

In the case of on-line testing, trials are performed gradually, opportunistically, and overall as non-intrusively as possible, e.g., while recharging the batteries of a mobile device or during extensive idle periods. (Or, for mobile devices, a Windows-style installation wizard could give users the option of testing the device upon first use or at some later time.) Multiple trials are collected over days, weeks, or even months. Testing time and power overheads recede over time as the DRAM is mapped. RAPID is only employed when the trials are

complete. Correct operation is assured because the system safely reverts to the default refresh policy when trials are incomplete.

3.3. Testing and Temperature

Conventional DRAMs use a single worst-case refresh period (typically 64 ms), safe for the highest rated temperature and all lower temperatures. CellularRAM™ temperature-compensated refresh (TCR) provides four refresh settings, corresponding to four temperature ranges. A given refresh setting is safe for the highest temperature in its range and all lower temperatures. Exploiting TCR requires an external temperature sensor or *a priori* knowledge of the peak operating temperature, to guide selection of the lowest safe refresh setting.

Temperature can be accommodated in RAPID similarly, using either a single or multiple temperature ranges. The only difference is the use of page-dependent information instead of page-independent information.

However, the testing method – off-line vs. on-line – affects temperature options. A key advantage of off-line testing is that temperature can be explicitly controlled. Thus, testing can be done for the highest rated temperature and possibly several lower temperatures. If a temperature sensor is available, we can select page retention times corresponding to the current temperature. If a temperature sensor is not available, we can at least select page retention times corresponding to the highest rated temperature.

On-line testing is less flexible in two ways. First, a temperature sensor is mandatory, to tag each trial with the maximum temperature for which the trial is safe. The sensor is likewise needed to select safe page retention times corresponding to the current temperature. Second, temperature cannot be controlled during testing. It is possible for the current temperature to exceed the maximum recorded temperature among trials, in which case a RAPID refresh period cannot be determined and we must downgrade to either of the two conventional refresh policies (worst-case refresh period or lowest safe refresh setting of TCR).

3.4. Page Retention Times

This section presents page retention times using the testing algorithm described in Section 3.1. Note that the DRAM under test implements a "close page" policy, meaning the page is not held open in the row buffer. Therefore, the row buffer is not used as a cache, guaranteeing reads and writes always truly access the memory cells.

The graph in Figure 2 shows retention times of the 16,384 pages in the chip. These measurements were taken at the particular room temperature (24°C). The y-axis shows retention time in milliseconds (ms) and the x-axis shows all 16,384 pages, labeled by "page #".

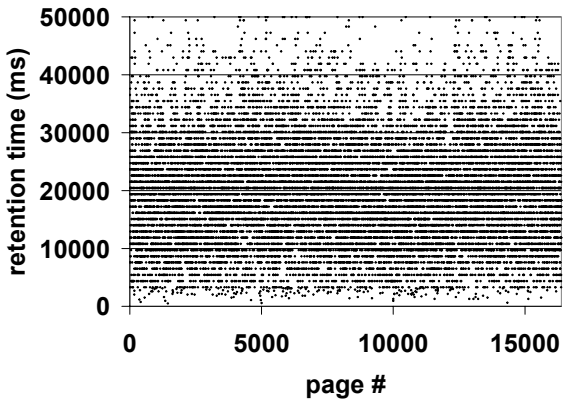

Figure 2. Measured retention times of different pages (rows) of DRAM at 24°C.

The first observation is that retention time varies noticeably among different pages, from 500 ms to slightly over 50 seconds.

The second observation is that most pages have longer retention times than the default 64 ms refresh period or even the longest temperature-compensated refresh period (TCR), which is at most 500 ms according to our results in Figure 2. For example, a refresh period of 1,400 ms covers 99.9% of all pages and a refresh period of 3,100 ms covers 99% of all pages. For a 99% DRAM bank utilization, RAPID-1 can consolidate data into the longest-retention 99% of the DRAM bank, permitting a refresh period as long as 3,100 ms without losing information.

As mentioned earlier, we observed some cells in the ISSI DRAM leaked to 0 and others to 1. However, leaking to 1 seems to be a much slower process, as the average page retention time for 1's is only 17 seconds compared to an average page retention time of 50 seconds for 0's (at room temperature for pages in the first bank).

The distribution of page retention times follows a bell curve, as shown on the left-hand side of Figure 3. The corresponding cumulative distribution on the right-hand side of Figure 3 shows the percent usable DRAM for a given target refresh period. For example, a 10 second refresh period covers about 85% of the DRAM pages.

Temperature is known to have a significant effect on leakage and, therefore, retention time in DRAM. We measured the page retention times of only the first bank at two higher temperatures, 45°C and 70°C (the latter is the maximum operating temperature for the chip). The temperature was raised using the heat gun pictured in Figure 1 (labeled 4). The temperature was monitored during testing, using a multi-meter with temperature-sensing thermistor (labeled 3) placed in contact with the DRAM package, and was kept within two degrees of the desired temperature. The results are shown in Figure 4 and Figure 5. Retention times decrease with increasing temperature as expected.

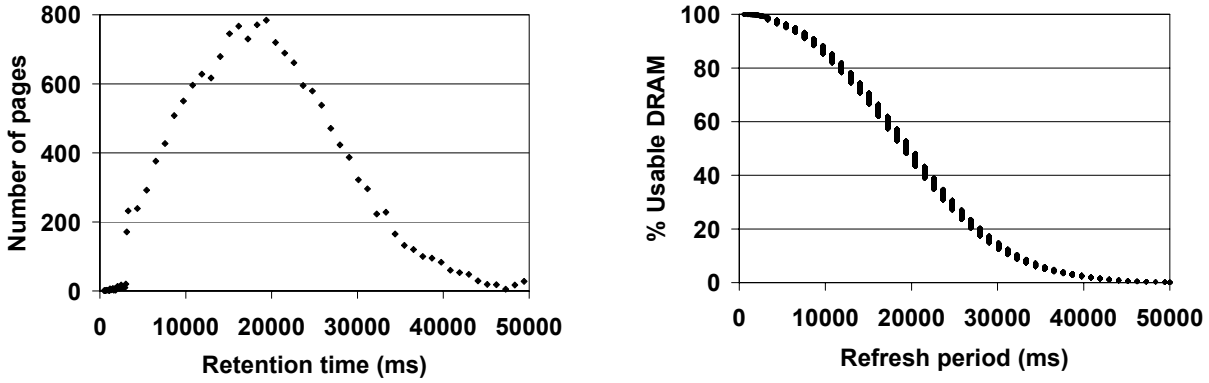

Figure 3. Distribution (left) and cumulative distribution (right) of page retention times for entire DRAM at room temperature (24°C).

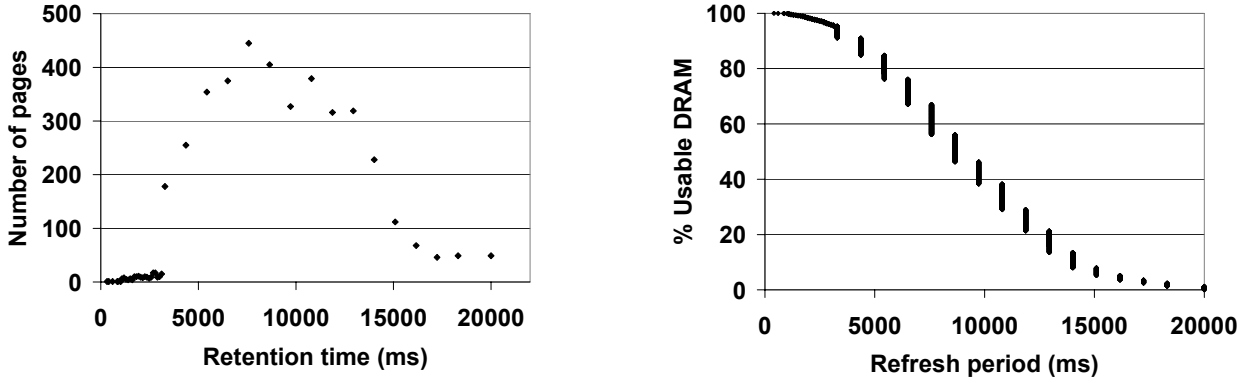

Figure 4. Distribution (left) and cumulative distribution (right) of page retention times for first DRAM bank at 45°C.

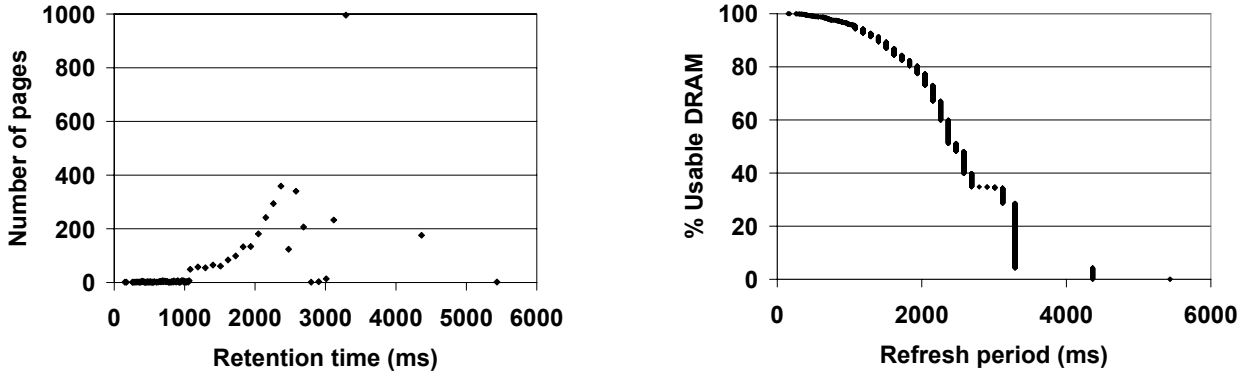

Figure 5. Distribution (left) and cumulative distribution (right) of page retention times for first DRAM bank at maximum operating temperature (70°C).

4. Retention-Aware Placement in DRAM (RAPID)

RAPID is a software-only solution for exploiting retention-time variations among different pages to minimize the refresh power in contemporary off-the-shelf DRAMs, a major component of power consumption in standby operation. The key idea is that longer-retention pages are preferred for allocation over shorter-retention pages. This enables selecting a global refresh period that is the longest possible, based on the shortest retention time among occupied pages instead of the shortest retention time among all DRAM pages.

The primary software support is modifications to routines which allocate and deallocate physical pages in memory. For example, the Linux kernel's virtual memory manager maintains a free-list of inactive physical pages – pages not touched for extended periods of time and deemed inactive – called the Inactive List. The Inactive List contains pages marked as "Inactive Dirty" or "Inactive Clean" (inactive dirty pages have yet to be flushed to their corresponding pages in the backing non-volatile storage). The RAPID-1 allocation/deallocation routines merely exclude outlier pages from the initial Inactive List. The RAPID-2 and RAPID-3 allocation/deallocation routines transform the single Inactive List into multiple Inactive Lists, one for each retention time bin. For systems with a virtual memory manager, existing Inactive List management routines would be adapted for RAPID as just described. Many embedded systems do not implement virtual memory, in which case equivalent user routines would be adapted or substituted with the RAPID allocation/deallocation routines.

The three RAPID versions are described in Sections 4.1 through 4.3. Some of the software implementation costs, in terms of software tables, are summarized in Section 4.4. Section 4.5 gives background on common refresh options available in off-the-shelf DRAMs, and explains how the single refresh period selected by RAPID can be applied to these DRAMs without custom hardware support.

4.1. RAPID-1

RAPID-1 is a static approach in which shortest-retention pages, constituting a small fraction of the overall DRAM, are made unavailable for use by application programs. They are made unavailable by excluding these pages from the Inactive List when it is initialized.

In this paper, for room temperature, we excluded all pages with retention times below 3,118 ms, a total of 168 pages (out of 16,384 pages). This yields a static refresh period of 3.2 s with 99% DRAM availability.

4.2. RAPID-2

Pages are placed into bins according to their retention times. Initial studies show 10 bins provide most of the benefit so 10 bins are used in this paper. The 10 bins are equally spaced between 3.2 s and 49 s. The shortest retention time is 3.2 s because the 168 pages below 3.2 s are excluded, like in RAPID-1. In the future, we plan to explore bins that are balanced in terms of page count. Also, note that using only 10 bins can be exploited *a priori* to reduce the run time of the testing algorithm described in Section 3.1.

The original Inactive List is split into 10 Inactive Lists, one per bin. Pages within each Inactive List are unordered, as usual, so allocating and deallocating pages from a given list remains an O(1) operation.

The bins are ordered from longest to shortest retention time. When none of the DRAM pages are populated, i.e., all Inactive Lists are full, the RAPID refresh period corresponds to the retention time of the highest bin.

A page allocation request is satisfied from the highest bin that has a non-empty Inactive List. If no more free pages are available in any of the bins that have populated pages, a page is allocated from the next lower bin. However, before the page is given over to the application program, the RAPID refresh period is decreased to accommodate the shorter retention time of the fresh bin.

When a page is deallocated, the page is returned to the Inactive List corresponding to its retention time. The page's entry in the page bin table (see Section 4.4) indicates the RAPID-2 bin to which the page belongs, to facilitate returning it to the correct Inactive List. Each bin knows how many pages (active plus inactive) belong to it. Thus, if an Inactive List becomes full when a page is returned to it, and the corresponding bin was previously the lowest bin with populated pages, then the RAPID refresh period is increased to the next higher bin that still has populated pages.

4.3. RAPID-3

RAPID-3 builds on RAPID-2. It uses the same allocation/deallocation routines described in Section 4.2, enhanced with page migration to reconsolidate data from lower to higher bins when higher bins become fragmented, i.e., underutilized. This causes the RAPID refresh period to respond faster to decreases in DRAM utilization, compared to RAPID-2.

Many migration policies are possible, from migrating immediately when a page in a higher bin becomes free, to periodically reconsolidating pages *en masse*. For our experiments, we migrate immediately when an opportunity presents itself.

4.4. RAPID Software Storage Costs

There are three major software data structures. The first is unique to RAPID. The last two have counterparts in existing page-managed systems.

- *Page bin table*. This table indicates which RAPID-2 bin each page belongs too. For 16K DRAM pages, there are 16K 4-bit entries (to encode up to 16 bins) for a storage cost of 8KB.
- *Logical-to-physical address translation table*. This is the page table in traditional virtual memory systems. The RAPID allocation routine does not change the structure or management of the traditional page table, it only affects which DRAM pages are otherwise sequentially or randomly allocated. For 16K DRAM pages, a one-to-one mapping of logical to physical

pages requires 16K 14-bit entries for a storage cost of 28KB.

- *Inactive list*. Although the single Inactive List is divided into multiple Inactive Lists for RAPID-2 and RAPID-3, the total storage cost remains the same as in traditional page-managed systems. For 16K DRAM pages, 16K list nodes are needed when no pages are populated. Assuming 8-byte nodes, the storage cost is 128KB.

4.5. Using RAPID Refresh Period with Off-the-shelf DRAMs

Off-the-shelf DRAMs typically provide one or both of the following conventional refresh options.

- *Self-refresh*. Self-refresh is implemented solely within the DRAM chip itself.
- *Auto-refresh*. The external memory controller issues regularly timed auto-refresh commands and the DRAM chip refreshes the next row in sequence. The memory controller does not send an address since the DRAM keeps track of the next row to be refreshed in sequence.

Self-refresh is significantly more power-efficient than auto-refresh, both in terms of DRAM power and external memory controller power. However, our search of DRAM datasheets reveals no DRAMs with a programmable self-refresh period. It is either 64 ms or temperature compensated, neither of which approaches the quasi-non-volatile RAPID refresh periods. If there exists a DRAM with a programmable self-refresh period, RAPID can exploit self-refresh directly by setting the self-refresh period to the RAPID refresh period.

An external memory controller typically supports a single programmable auto-refresh period. While directly compatible with the RAPID refresh period, this is not a good general solution because auto-refresh is too inefficient during long standby periods, i.e., it is better to power down the memory controller and DRAM leaving only self-refresh on. Auto-refresh may make sense during active periods. In fact, some DRAMs (like our ISSI DRAM) only support auto-refresh during active periods and self-refresh during standby periods, to explicitly avoid conflicts between internal self-refresh and external requests.

The CellularRAM™ [3] for ultra-low-power systems only supports the efficient self-refresh option. Self-refresh is used during both active and standby periods. However, the self-refresh period is not programmable except for limited temperature compensation support. As is typical, self-refresh can be enabled/disabled via a configuration register. Thus, we propose the following approach for coupling RAPID with CellularRAM™:

- When RAPID is in use, self-refresh is disabled by default.
- RAPID sets up a periodic timer interrupt, its period equal to the RAPID refresh period.
- When the interrupt occurs, a lightweight handler enables the DRAM's self-refresh and sets up a near-term interrupt for 64 ms in the future. During the 64 ms interval, self-refresh transparently refreshes the entire DRAM. The near-term interrupt invokes a second lightweight handler that disables the DRAM's self-refresh.
- In this way, the entire DRAM is refreshed only once – in a burst fashion – every RAPID refresh period.

While we discussed the above approach in the context of DRAMs with only self-refresh, the approach can be adapted to exploit programmable auto-refresh during active periods and self-refresh during standby periods, for the ISSI DRAM and others with a similar refresh dichotomy.

5. Evaluation Methodology

Mobile devices often exhibit short bursts of activity (active mode) followed by long idle periods (standby mode) [23][9]. During active mode, DRAM refresh may be a small component of overall system power, whereas, during standby mode, DRAM refresh may be the largest power component.

Obtaining activity traces of a real mobile device or sensor node is beyond the scope of this paper. We use the above bimodal active/standby characterization to guide a simplified evaluation technique for comparing refresh optimizations. This evaluation technique focuses on DRAM utilization over long tracts of time. We divide the long timeline into consecutive 100-second intervals and randomly inject 100-second active periods in place of otherwise standby periods, with a random probability that yields a typical active utilization, e.g., 5%. During active periods, we inject a random number of page requests. A request is equally probable to be an allocation vs. a deallocation so that DRAM utilization remains on average what the initial utilization was, making it possible to target a particular average DRAM utilization while still having significant utilization fluctuations during the timeline. Different refresh techniques may or may not exploit fluctuations in DRAM utilization, yielding a means for comparing techniques in a generic way. We compare the following refresh techniques:

- *TCR* (TCR): Optimal temperature-compensated refresh (TCR), i.e., the self-refresh period is based on the shortest retention time among all pages for the current temperature, as measured in Section 3.4. We use this as the baseline since the default 64 ms self-refresh is overly pessimistic and TCR is available in some current DRAMs.
- *RAPID-1* (R-1), *RAPID-2* (R-2), *RAPID-3* (R-3): These are the new RAPID methods.

- *HW-Multiperiod* (HW-M): A custom hardware solution, in which each page is refreshed at a tailored refresh period that is a multiple of the shortest refresh period among all pages in the DRAM.
- *HW-Multiperiod-Occupied* (HW-M-O): Same as HW-Multiperiod, but only pages that are currently occupied are refreshed.
- *HW-Ideal* (HW-I): An ideal custom hardware solution, in which each page is refreshed at its own tailored refresh period.
- *HW-Ideal-Occupied* (HW-I-O): Same as HW-Ideal, but only pages that are currently occupied are refreshed.

The simulated timeline is 24 hours. All simulations are configured for 5% activity and 95% standby operation. Average DRAM utilizations of 25%, 50%, and 75% are targeted. Three operating temperatures are simulated, 25°C (room temperature), 45°C, and 70°C.

The Micron CellularRAM™ [3] is the basis for refresh power. Given the CellularRAM™ refresh power at its default self-refresh period, refresh power can be calculated for an arbitrary refresh period via simple scaling.

6. Results

Figure 6 shows the refresh energy consumption (in mW·hours) for an average DRAM utilization of 75% at three different temperatures. TCR uses the worst-case refresh period at a given temperature (500 ms at 25°C, 323 ms at 45°C, and 151 ms at 70°C). By simply discarding 1% of outlier pages, RAPID-1 yields 83% (25°C), 80% (45°C), and 70% (70°C) energy savings with respect to TCR. RAPID-2 yields 93% (25°C and 45°C) and 92% (70°C) energy savings with respect to TCR. RAPID-3 yields 95% (25°C and 45°C) and 93% (70°C) energy savings with respect to TCR. Another positive result is that RAPID-2 and RAPID-3 are nearly as effective as the custom hardware approaches.

Energy savings of RAPID-1 decreases only moderately as temperature increases, from 83% to 70% over the full temperature range. One factor contributing to this decline is that, some pages that lie above the RAPID-1 retention-time threshold at 25°C, lie below this threshold at 70°C. These pages become "outliers" with respect to the 25°C RAPID-1 standard, yet they are not excluded.

In contrast, energy savings of RAPID-2 and RAPID-3 hardly decline with increasing temperature. At 75% average DRAM utilization, typical RAPID-2 and RAPID-3 refresh periods are substantially long even at 70°C.

Energy consumption of RAPID-1 at 70°C (worst-case RAPID-1) is comparable to energy consumption of TCR at 25°C (best-case TCR). This implies that a worst-case, non-temperature-adjusted RAPID-1 implementation yields the same or better energy than TCR, suggesting a potentially simpler alternative to temperature-aware DRAM design. The same case can be made in even stronger terms, for non-temperature-adjusted RAPID-2 and RAPID-3 implementations.

Figure 7 shows refresh energy at 25°C for average DRAM utilizations of 75%, 50%, and 25%. A key observation is that, as the DRAM utilization decreases, both RAPID-2 and RAPID-3 yield more energy savings than the two hardware techniques HW-M and HW-I. At 75% average utilization, the energy consumption of RAPID-2 and RAPID-3 are slightly higher than HW-M and HW-I. Yet, at 50% and 25% average utilizations, both RAPID-2 and RAPID-3 consume less energy than HW-M and even HW-I. At lower DRAM utilizations, many lines are unoccupied and RAPID's refresh period exceeds the "average" period of the hardware techniques.

The ideal hardware technique, HW-I-O, provides a more reliable lower bound on energy because it does not refresh unoccupied pages. For 25% average utilization, RAPID-3 (and even RAPID-2) approaches this lower bound.

However, even HW-I-O does not necessarily provide a lower bound on refresh energy. For non-RAPID implementations, pages are allocated without regard to page retention times. Thus, HW-I-O energy can differ for the same DRAM utilization, depending on which pages are allocated. For example, for 25% utilization, the energy of HW-I-O can vary from 0.01 mW·hours to 0.023 mW·hours, depending on whether the best or worst 25% of the DRAM pages are occupied. Energy of RAPID-3 at 25% utilization is 0.018 mW·hours. Therefore, it is possible for RAPID-3 to outperform HW-I-O.

7. Summary and Future Work

DRAM is predicted to displace SRAM in future embedded systems as functionality evolves. This future can be better met by dealing with the DRAM refresh problem and thereby reap the capacity benefits of DRAM without impacting battery life.

The key lies with exploiting dramatic variations in retention times among different DRAM pages. We proposed Retention-Aware Placement in DRAM (RAPID), novel software approaches that can exploit off-the-shelf DRAMs to reduce refresh power to vanishingly small levels approaching non-volatile memory. The key idea is to favor longer-retention pages over shorter-retention pages when allocating DRAM pages. This allows selecting a single refresh period that depends on the shortest-retention page among populated pages, instead of the shortest-retention page overall. We explore three versions of RAPID and observe refresh energy savings of 83%, 93%, and 95%, relative to conventional temperature-compensated refresh. RAPID with off-the-shelf DRAM also approaches the energy levels of

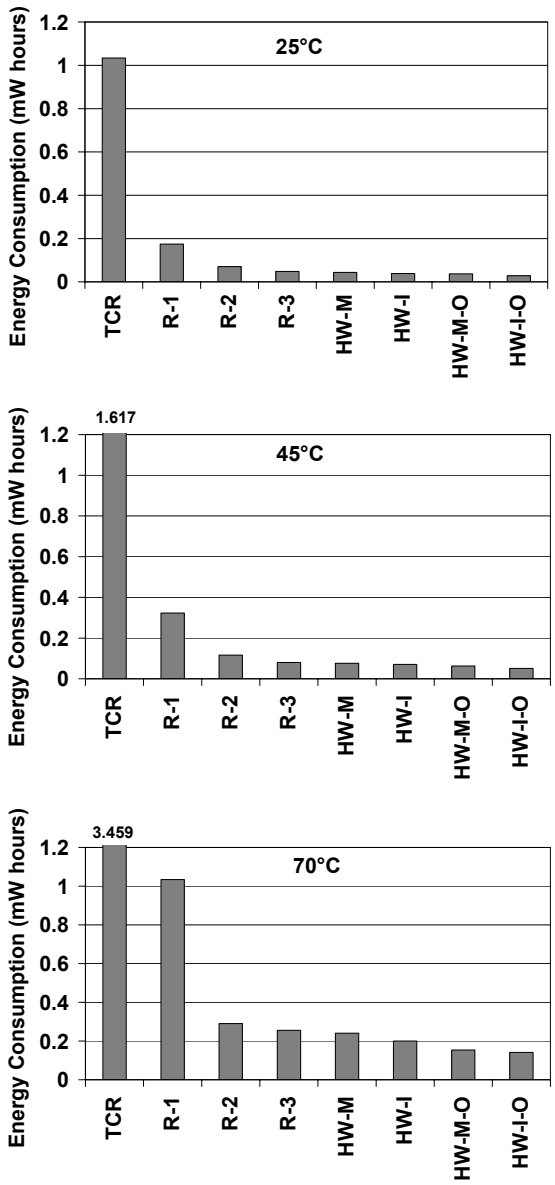

Figure 6. Refresh energy for 75% average DRAM utilization, at 25°C, 45°C, and 70°C.

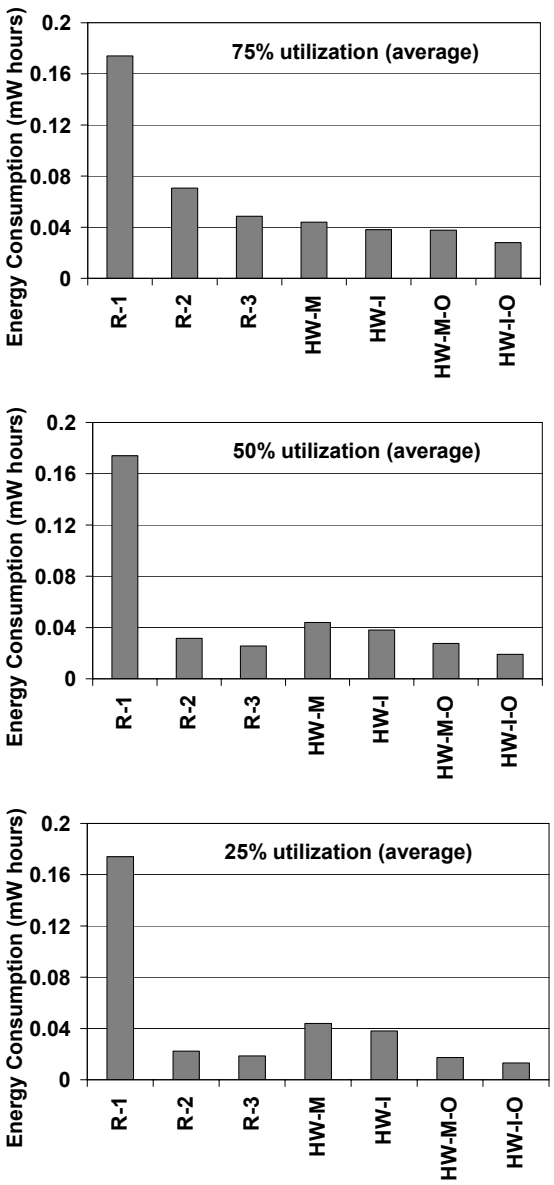

Figure 7. Refresh energy at 25°C for average DRAM utilizations of 75%, 50%, and 25%.

idealized techniques that require custom DRAM support. This ultimately yields a software implementation of quasi-non-volatile DRAM.

Much future work remains. In the area of testing, we plan to extend retention-time measurements to multiple chips from different vendors, develop rigorous testing methods with a statistical basis, apply arbitrary test patterns to identify potential cross-cell retention-time degradation or other interactions, and explore formulas for automatically scaling retention times with temperature. In the area of RAPID software, we plan to explore modifications for handling differing O/S and DRAM page sizes, sub-page retention-aware placement (e.g., padding data structures to skip over sub-par cells), and binning strategies in RAPID-2/RAPID-3. We may also explore novel interactions between RAPID and ECC, for example, defining a page's retention time according to its second-worst bit and leveraging ECC to repair the worst bit. Finally, we plan to develop a prototype embedded system with RAPID, to demonstrate extended battery life, conveniences of instant-on/instant-off computing, enhanced resilience to power outages, and other benefits traditionally afforded by non-volatile memory.

8. Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments. This research was supported in part by NSF grants CCR-0207785, CCR-0208581, and CCR-0310860, NSF CAREER grant CCR-0092832, and funding and equipment donations from Intel and Ubicom.

9. References

[1] T. Austin, D. Blaauw, T. Mudge, K. Flautner. Making Typical Silicon Matter with Razor. IEEE Computer, Volume 37 Number 3, March 2004.

[2] J. Burgan. Variable Refresh Control for a Memory. US Patent #6,778,457, Aug. 2004.

[3] CellularRAM™ – Micron website: http://download.micron.com/pdf/flyers/cellularram_flyer.pdf.

[4] L.L. Chen Hsu, G. Frankowsky, O. Weinfurtner. Dynamic DRAM Refresh Rate Adjustment Based on Cell Leakage Monitoring. US Patent #6,483,764, Nov. 2002.

[5] H.Y. Cho, J.K. Oh. Self-Refresh Apparatus for a Semiconductor Memory Device. US Patent #6,229,747, May 2001.

[6] D. Ernst et al. Razor: A Low Power Pipeline Based on Circuit-Level Timing Speculation. MICRO-36, IEEE CS Press, 2003, pp.7-18.

[7] T. Hamamoto, S. Sugiura, S. Sawada. On the Retention Time Distribution of Dynamic Random Access Memory (DRAM). IEEE Transactions on Electron Devices, Volume 45 Issue 6, June 1998, pp.1300-1309.

[8] M. Hempstead, N. Tripathi, et al. An Ultra Low Power System Architecture for Sensor Network Applications. 32nd Int'l Symp. on Computer Architecture, June 2005.

[9] J. Kim, M. Papaefthymiou. Block-Based Multiperiod Dynamic Memory Design for Low Data-Retention Power. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 11, No. 6, Dec. 2003, pp.1006-1018.

[10] J. Kim, M. Papaefthymiou. Block-Based Multi-Period Refresh for Energy Efficient Dynamic Memory. 14th ASIC/SOC Conference, Sept. 2001, pp 193-197.

[11] T. Komatsu. Self-Refreshing of Dynamic Random Access Memory Device and Operating Method Therefor. US Patent #4,943,960, July 1990.

[12] A. Lebeck, X. Fan, H. Zeng, C. Ellis. Power Aware Page Allocation. ASPLOS IX, Nov. 2000.

[13] T. Murotani. Temperature Responsive Refresh Control Circuit. US Patent #4,393,477, June 1983.

[14] L. Nazahandali, B. Zhai, et al. Energy Optimization of Subthreshold-Voltage Sensor Network Processors. 32nd Int'l Symp. on Computer Architecture, June 2005.

[15] T. Ohsawa, K. Kai, K. Murakami. Optimizing the DRAM Refresh Count for Merged DRAM/Logic LSIs. ISPLED98, pp. 82-87, August 1998.

[16] P. Pelley, J. Burgan. Memory Having Variable Refresh Control and Method Therefor. US Patent #6,781,908, Aug. 2004.

[17] Ubicom. http://www.ubicom.com

[18] A. Uht. Going Beyond Worst-Case Specs with TEAtime. IEEE Computer, Volume 37 Number 3, March 2004.

[19] UtRAM™ - Samsung website http://www.samsung.com/Products/Semiconductor/SRAM.

[20] M. Viredaz, D. Wallach. Power Evaluation of a Handheld Computer: A Case Study. Compaq-WRL Research Report 2001/1, 2001. http://www.hpl.hp.com/research/papers/2003/handheld.pdf

[21] K. Yanagisawa. Semiconductor Memory. US Patent #4,736,344, April 1988.

[22] S. Fyffe. DRAMatic Cell Phone Showdown. Electronic News, 2/12/2001.
M. Kanellos, B. Charny. Phone makers copy PCs. CNET News.Com, 4/2/2002.
Dram Makers Prep Multichip Cell Phone Memories. ElectroSpec, 1/3/2003.

[23] Infineon – Mobile-RAM, Specialty DRAMs. Application Note, V 1.1, Feb. 2002.