Solar-DRAM: Reducing DRAM Access Latency by Exploiting the Variation in Local Bitlines

Jeremie S. Kim[‡§] Minesh Patel[§] Hasan Hassan[§] Onur Mutlu[§‡]
[‡]Carnegie Mellon University [§]ETH Zürich

DRAM latency is a major bottleneck for many applications in modern computing systems. In this work, we rigorously characterize the effects of reducing DRAM access latency on 282 state-of-the-art LPDDR4 DRAM modules. As found in prior work on older DRAM generations (DDR3), we show that regions of LPDDR4 DRAM modules can be accessed with latencies that are significantly lower than manufacturer-specified values without *causing failures. We present novel data that 1) further supports the viability of such latency reduction mechanisms and 2) exposes a variety of new cases in which access latencies can be effectively reduced. Using our observations, we propose a new low-cost mechanism, Solar-DRAM, that 1) identifies failure-prone regions of DRAM at reduced latency and 2) robustly reduces average DRAM access latency while maintaining data correctness, by issuing DRAM requests with reduced access latencies to non-failure-prone DRAM regions. We evaluate Solar-DRAM on a wide variety of multi-core workloads and show that for 4-core homogeneous workloads, Solar-DRAM provides an average (maximum) system performance improvement of 4.31% (10.87%) compared to using the default fixed DRAM access latency.*

keywords— DRAM Latency; DRAM Characterization; Process Variation; LPDDR4; Memory; Memory Controllers

1. Introduction

High DRAM access latency presents a significant bottleneck for memory-intensive applications running on modern systems [46, 49]. The growing disparity between CPU performance and DRAM access latency continues to exacerbate the bottleneck. As technology node sizes continue to decrease in DRAM manufacturing, circuitry variation in DRAM cells, which results from process manufacturing variation, increases. This increase in variation leads to DRAM modules that are comprised of cells with a wide range of properties, and these properties determine a DRAM cell's propensity for failure. We can directly observe a DRAM cell's propensity for failure by accessing it with *reduced* DRAM timing parameters below manufacturer-specified values and observing its rate of failure. We identify cells that fail when accessed with reduced DRAM timing parameters as "weak" cells, and cells that do not fail as "strong" cells. Unfortunately, modern memory controllers do *not* exploit this variation in DRAM cells and simply use, for all cells, a *fixed* set of DRAM timing parameters that account for the most failure-prone (i.e., weakest acceptable) DRAM cell that can be manufactured for a given yield. These fixed timing parameters are set such that the circuit elements in the weakest cell have time to stabilize during a DRAM access, and failures do not occur during regular DRAM operation.

Recent works [6, 37] study the failures that result from reducing DRAM timing parameters related to access latency (i.e., *DRAM access timing parameters*). We refer to these failures as *access failures*. These works observe that access failures exhibit spatial locality in DRAM modules. Based on the assumption that DRAM cells can be *statically* categorized as "weak" or "strong", the authors propose mechanisms to selectively reduce DRAM access timing parameters for accesses to DRAM locations that are comprised of *stronger* bits (i.e., bits that do not fail when accessed with reduced DRAM access timing parameters) using a *static* profile of cells. Unfortunately, these prior works [6,37] 1) analyze access failure patterns only in *older* DDR3 DRAM modules and 2) *fail* to demonstrate the necessary characterization to support their assumption that identifying weak cells via simple *static* profiling is *robust*.

To overcome the shortcomings of prior work, our **goal** in this paper is twofold. We aim to 1) provide a more rigorous characterization of activation failures on *state-of-the-art LPDDR4* DRAM modules to show the viability of mechanisms [6, 37] that employ variable DRAM access latency by relying on a *static* profile, and 2) devise new mechanisms that exploit *more activation failure characteristics* observed on newer *state-of-the-art LPDDR4* DRAM modules.

We characterize 282 state-of-the-art 2y-nm LPDDR4 modules. To do so, we develop an infrastructure with a thermally-controlled chamber and rigorously test our DRAM modules with a sweep of parameters including DRAM temperature, DRAM access latency, testing time interval, and data patterns written to the DRAM array. Using our infrastructure, we study a particular class of access failures, called *activation failures*, that occur when a key parameter for determining the service time of a request (t_{RCD}, i.e., row activation latency) is reduced beyond manufacturer-specified values. We provide a rigorous characterization of activation failures and make *four* key new observations on LPDDR4 modules: 1) activation failures exhibit high spatial locality *within a column* of DRAM cells (i.e., a bitline) at the granularity of a *subarray*, where a subarray is a substructure of DRAM typically containing 512 or 1024 rows of DRAM cells [7, 31]; 2) the probability that a bitline within a subarray (i.e., local bitline) contains activation failures does *not change significantly over time*. This means that we can rely on a one-time profile of weak local bitlines to determine, at any point in time, whether an activation failure might occur in a cache line by an access with a reduced t_{RCD}; 3) a DRAM access to a row that is *closed*, i.e., not currently buffered in the DRAM *row buffer* (an in-DRAM cache that enables quick reads and writes to locations within a DRAM row), requests the 0^{th} cache line of the row, with a high probability. Since t_{RCD} dictates the latency to activate a closed row, reducing the access latency of the 0^{th} cache line alone could provide significant performance benefit; and 4) DRAM *write* requests can be issued with a greatly reduced

t_{RCD} (i.e., by 77%) without compromising DRAM reliability. This is because t_{RCD} dictates the amount of time needed for data in the requested DRAM cells to be amplified to a *readable* voltage level, which does *not* govern *write* operations.

Building on our detailed experimental characterization, we propose *Subarray-optimized Access Latency Reduction DRAM (Solar-DRAM)*, a mechanism that exploits each of these new observations to significantly and robustly reduce DRAM access latency. The key idea of Solar-DRAM is to issue 1) DRAM reads with reduced t_{RCD} (i.e., by 39%) unless the requested DRAM cache line contains weak DRAM cells that are likely to fail under reduced t_{RCD}, and 2) all DRAM writes with reduced t_{RCD} (i.e., by 77%). Solar-DRAM determines whether a DRAM cell is weak using a *static profile* of local bitlines, which we experimentally find to be *reliable across time*. Compared to state-of-the-art LPDDR4 DRAM, Solar-DRAM provides significant system performance improvement while maintaining data correctness.

We make the following six **key contributions**:

1. Using 282 LPDDR4 DRAM modules from three major DRAM manufacturers, we extensively characterize the effects of multiple testing conditions (e.g., DRAM temperature, DRAM access latency parameters, data patterns written in DRAM) on activation failures.
2. We demonstrate the viability of mechanisms that exploit variation in access latency of DRAM cells by showing that cells that operate correctly at reduced latency continue to operate correctly at the same latency over time. That is, a DRAM cell's activation failure probability is *not* vulnerable to significant variation over short time intervals.
3. We present data across our DRAM modules, that activation failures exhibit high spatial locality and are tightly constrained to a small number of *columns* (i.e., on average 3.7%/2.5%/2.2% per bank for DRAM chips of manufacturers A/B/C) at the granularity of a DRAM subarray.
4. We demonstrate that t_{RCD} can be greatly reduced (i.e., by 77%) for DRAM *write* requests while *still* maintaining data integrity. This is because t_{RCD} defines the amount of time required for data within DRAM cells to be amplified to a *readable* voltage level, which does *not* govern DRAM *write* operations.
5. We find that across SPEC CPU2006 benchmarks, DRAM accesses to closed rows typically request the 0^{th} *cache line* in the row, with a maximum (average) probability of 22.2% (6.6%). This is much greater than the expected probability (i.e., 3.1%) assuming that DRAM accesses to closed rows access each cache line with an equal probability. Since t_{RCD} affects only DRAM accesses to closed DRAM rows, we find that simply reducing t_{RCD} for all accesses to the 0^{th} cache lines of all DRAM rows improves overall system performance by up to 6.54%.
6. We propose Solar-DRAM, a mechanism that exploits our three key observations on reliably reducing the t_{RCD} timing parameter. Solar-DRAM selectively reduces t_{RCD} for 1) reads to DRAM cache lines containing "weak" or "strong" cells, and 2) writes to all of DRAM. We evaluate Solar-DRAM on a variety of multi-core workloads and show that compared to *state-of-the-art* LPDDR4 DRAM, Solar-DRAM improves performance by 4.97% (8.79%) on heterogeneous and by 4.31% (10.87%) on homogeneous workloads.

2. Background

We describe the DRAM organization and operation necessary for understanding our observations and mechanism for reducing DRAM access latencies. We refer the reader to prior works [5, 6, 8, 13, 15, 16, 25, 28, 29, 31, 32, 35, 36, 37, 38, 39, 41, 58, 61, 63, 70, 72] for more detail.

2.1. DRAM Organization

Figure 1 illustrates the organization of a DRAM module. The processor interfaces with the DRAM module via a memory controller at the *channel* granularity. A DRAM channel is partitioned into *ranks*. Each rank comprises a set of *chips* that operate in unison to service a single DRAM request at a time.

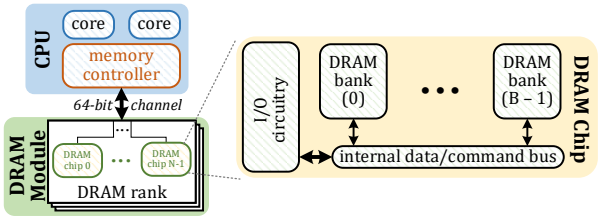
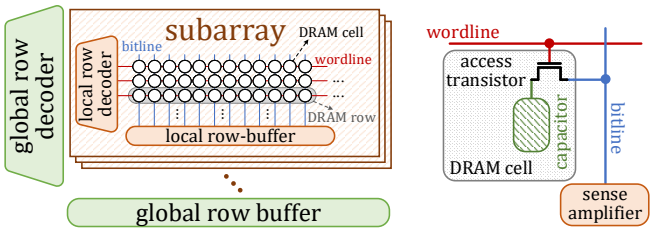


Figure 1: DRAM module organization.

Figure 2a presents the internal organization of a DRAM bank, which consists of a 2D array of *DRAM cells*. A DRAM cell (Figure 2b) consists of 1) a capacitor, which stores data as one of two levels of charge (e.g., *high charge* representing a logical "1" and *low charge* representing a logical "0"), and 2) a transistor, which controls access to the DRAM cell's data. Each DRAM cell in a row is connected to a wire called *wordline* via the gate of the access transistor. Each DRAM cell in a column is connected to another wire called *bitline* via the source of the access transistor. A DRAM cell capacitor is attached to the drain of the access transistor.



(a) DRAM Bank. (b) DRAM Cell.

Figure 2: DRAM bank and cell organization.

The cells in a DRAM bank are organized hierarchically. A DRAM row typically consists of 4096 or 8192 cells, which all share the same wordline. Multiple DRAM rows are grouped into a *subarray*. A subarray typically contains 512 or 1024 rows. Each subarray has its own *local row decoder* and *local row buffer* (i.e., local sense amplifiers). We refer to the shared vertical wire connecting a column of cells to a local sense amplifier as a *local bitline*. All subarrays in a bank share a *global row decoder* and *global row buffer*. We refer to the wire connecting an active local sense amplifier to a global sense amplifier as the *global bitline*. Only a single DRAM row per bank can be activated (i.e., open) at a time in the row buffer. An open row can serve multiple read and write

requests without incurring precharge and activation delays. Thus, the row buffer effectively serves as a single-entry cache for the open row.

2.2. DRAM Operation

The memory controller performs read and write operations on a DRAM module by issuing a set of DRAM commands. The four major commands that the memory controller issues to perform a DRAM access are `ACTIVATE`, `READ`, `WRITE`, and `PRECHARGE`. To correctly perform an access, the memory controller not only issues these commands in a particular order, but also obeys the DRAM *timing parameters* between consecutive commands.

To perform an access, the memory controller first issues an `ACTIVATE` command to *open* (or activate) a row in a bank, as determined based on the requested address. A row activation happens in three steps. First, upon receiving the `ACTIVATE` command, the global and local row decoders enable the wordline of the row that corresponds to the row address provided with the command. Second, the enabled wordline turns on the access transistors of the row's cells. As a result, charge sharing occurs between each cell capacitor in the row and its attached bitline. Charge sharing slightly perturbs the bitline voltage towards the direction of the original charge level of the cell. Third, after *charge-sharing* completes, the sense amplifier detects the perturbation in the bitline voltage and gradually restores the bitline, and thus, the *attached* cell, to full 0 or 1 (i.e., to ground or V_{dd}).

Once the bitline reaches a voltage level called V_{access}, the row is ready to be *reliably* accessed. The timing parameter that dictates when the bitline reaches V_{access} after issuing the `ACTIVATE` command is called t_{RCD}. The memory controller must satisfy t_{RCD} between consecutive `ACTIVATE` and `READ` (or `WRITE`) commands. The memory controller issues a single `READ` command to fetch a *cache line*, which is the granularity at which the DRAM module can be accessed.

To access data from another row, the memory controller must first close, or *precharge*, the currently-open row. A row is ready to precharge when the sense amplifier completes restoring the DRAM cell to a full 0 or 1. The timing parameter that the memory controller has to satisfy between consecutive `ACTIVATE` and `PRECHARGE` commands is called t_{RAS}. Once `PRECHARGE` is issued, a timing parameter called t_{RP} has to be satisfied prior to issuing a new `ACTIVATE` to the *same bank*.

2.3. DRAM Failure Modes

As we describe in Section 2.2, the memory controller must satisfy timing parameters associated with DRAM commands for correct operation. We define *access latency failures* as failures that occur due to accessing a DRAM module with *any* reduced timing parameter. In this paper, we focus on *activation failures*, which is a special case of access latency failures, caused by reducing the t_{RCD} timing parameter.

An *activation failure* occurs due to insufficient time for the sense amplifier to drive the bitline to V_{access}. Depending on the reduction in the t_{RCD} parameter, there are two modes of *activation failure*. First, accessing the DRAM with a reduced t_{RCD} may result in transient failures in the returned data, but no failures in the data stored in the DRAM cells. In this case, the *next access* to the same row that satisfies the timing parameters would return correct data. Such a failure may happen when the bitline does *not* reach V_{access} prior to the read operation but the sense amplifier continues to drive the bitline towards the same direction (i.e., full 0 or 1) as the charge-sharing phase has already started. The second mode of activation failure *destroys* the data stored in a DRAM cell. Such a failure may happen when, at the time the `READ` is issued, the bitline voltage level is even lower compared to the first mode of *activation failure*. In this case, the read operation could significantly disturb the bitline such that the sense amplifier starts driving the bitline towards the opposite of the original direction. We observe both of the *activation failure* modes in our experiments with real LPDDR4 DRAM modules.

3. Motivation and Goal

Many prior works [15, 30, 31, 34, 38, 45, 47, 48, 73] show that various important workloads exhibit *low* access locality and thus are unable to effectively exploit *row-buffer locality*. In other words, these workloads issue a significant number of DRAM accesses that result in bank (i.e., row buffer) conflicts, which *negatively* impact overall system performance. Each access that causes a bank conflict requires activating a closed row, a process whose latency is dictated by the t_{RCD} timing parameter. The memory controller must wait for t_{RCD} before issuing any other command to that bank. To reduce the overhead of bank conflicts, we aim to reduce the t_{RCD} timing parameter while maintaining data correctness.

Prior Observations. In a recent publication, Chang et al. [6] observe that activation failures 1) are *highly* constrained to global bitlines and regions of memory that are closer to the row decoders, 2) can *only* affect cells within the cache line that is first requested in a closed row, and 3) propagate back into DRAM cells and become *permanent* failures in the stored data.

Based on these observations, Chang et al. propose FLY-DRAM, which *statically* profiles DRAM *global bitlines* as *weak* or *strong* using a one-time profiling step. During execution, FLY-DRAM relies on this *static* profile to access *weak* or *strong* global bitlines with *default* or *reduced* t_{RCD}, respectively.

Unfortunately, [6] falls short in three aspects. First, the paper lacks analysis of whether a *strong* bitline will ever become a *weak* bitline or vice versa. This analysis is necessary to demonstrate the viability of relying on a static profile of global bitlines to guarantee data integrity. Second, the authors present a characterization of activation failures on an older generation of DRAM (DDR3). Third, the proposed mechanism, FLY-DRAM, does not *fully* take advantage of all opportunities to reduce t_{RCD} in modern DRAM modules (as we show in Section 5).

Given the shortcomings of prior work [6], **our goal** is to 1) present a more rigorous characterization of activation failures on *state-of-the-art LPDDR4* DRAM modules, 2) demonstrate the viability of mechanisms that rely on a static profile of weak cells to reduce DRAM access latency, and 3) devise new mechanisms that exploit *more activation failure characteristics* on *state-of-the-art LPDDR4* DRAM modules to further reduce DRAM latency.

4. Testing Methodology

To analyze DRAM behavior under reduced t_{RCD} values, we developed an infrastructure to characterize state-of-the-art LPDDR4 DRAM chips [19] in a thermally-controlled chamber. Our testing environment gives us precise control over DRAM commands and t_{RCD}, as verified via a logic analyzer probing the command bus. In addition, we determined the address mapping for internal DRAM row scrambling so that we could study the spatial locality of activation failures in the physical DRAM chip. We test for activation failures across a DRAM module using Algorithm 1. The key idea is to access every cache line across DRAM, and open a closed row on each access. This *guarantees* that we test every DRAM cell's propensity for activation failure.

Algorithm 1: DRAM Activation Failure Testing

```
1 DRAM_ACT_fail_testing(data_pattern, reduced_tRCD):
2   write data_pattern (e.g., solid 1s) into all DRAM cells
3   foreach col in DRAM module:
4     foreach row in DRAM module:
5       refresh(row)                    // replenish cell voltage
6       precharge(row)                  // ensure next access activates row
7       read(col) with reduced_tRCD     // induce activation failures on col
8       find and record activation failures
```

We first write a known data pattern to DRAM (Line 2) for consistent testing conditions. The *for loops* (Lines 3-4) ensure that we test all DRAM cache lines. For each cache line, we 1) refresh the row containing it (Line 5) to induce activation failures in cells with similar levels of charge, 2) precharge the row (Line 6), and 3) activate the row again with a *reduced* t_{RCD} (Line 7) to induce activation failures. We then find and record the activation failures in the row (Line 8), by comparing the read data to the data pattern the row was initialized with. We experimentally determine that Algorithm 1 takes approximately 200ms to test a single bank.

Unless otherwise specified, we perform all tests using 2y-nm LPDDR4 DRAM chips from three major manufacturers in a thermally-controlled chamber held at 55°C. We control the ambient temperature precisely using heaters and fans. A microcontroller-based PID loop controls the heaters and fans to within an accuracy of 0.25°C and a reliable range of 40°C to 55°C. We keep the DRAM temperature at 15°C above ambient temperature using a separate local heating source. This local heating source probes local on-chip temperature sensors to smooth out temperature variations due to self-induced heating.

5. Activation Failure Characterization

We present our extensive characterization of activation failures in modern LPDDR4 DRAM modules from three major DRAM manufacturers. We make a number of key observations that 1) support the viability of a mechanism that uses a *static profile* of weak cells to exploit variation in access latencies of DRAM cells, and 2) enable us to devise new mechanisms that exploit *more activation failure characteristics* to further reduce DRAM latency.

5.1. Spatial Distribution of Activation Failures

We first analyze the spatial distribution of activation failures across DRAM modules by visually inspecting bitmaps of activation failures across many DRAM banks. A *representative* 1024x1024 array of DRAM cells with a significant number of activation failures is shown in Figure 3. Using these bitmaps, we make three key observations. **Observation 1:** Activation failures are highly constrained to *local*

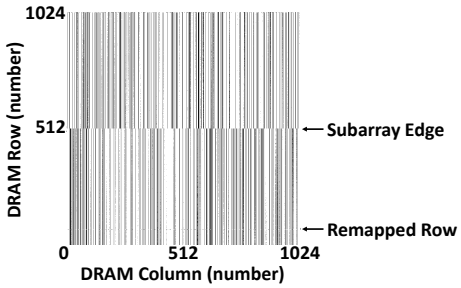


Figure 3: Activation failure bitmap in 1024x1024 cell array.

bitlines. We infer that the granularity at which we see bitline-wide activation failures is a subarray. This is because the number of consecutive rows with activation failures on the same bitline falls within the range of expected modern subarray sizes of 512 to 1024 [31, 37]. We hypothesize that this occurs as a result of process manufacturing variation at the level of the local sense amplifiers. Some sense amplifiers are manufactured "weaker" and *cannot* amplify data on the local bitline as quickly. This results in a higher probability of activation failures in DRAM cells attached to the same "weak" local bitline. While manufacturing process variation dictates the local bitlines that contain errors, the manufacturer design decisions for subarray size dictates the number of cells attached to the same local bitline, and thus, the number of consecutive rows that contain activation failures in the same local bitline. **Observation 2:** Subarrays from Vendor B and C's DRAM modules consist of 512 DRAM rows, while subarrays from Vendor A's DRAM modules consist of 1024 DRAM rows. **Observation 3:** We find that within a set of subarray rows, very few rows (<0.001%) exhibit a significantly different set of cells that experience activation failures compared to the expected set of cells. We hypothesize that the rows with significantly different failures are rows that are *remapped* to redundant rows (see [25, 40]) after the DRAM module was manufactured (indicated in Figure 3).

We next study the granularity at which activation failures can be induced when accessing a row. We make two observations (also seen in prior work [6]). **Observation 4:** When accessing a row with low t_{RCD}, the errors in the row are constrained to the DRAM cache line granularity (typically 32 or 64 bytes), and only occur in the aligned 32 bytes that is first accessed in a closed row (i.e., up to 32 bytes are affected by a single low t_{RCD} access). Prior work [6] also observes that failures are constrained to cache lines on a system with 64 byte cache lines. **Observation 5:** The first cache line accessed in a closed DRAM row is the *only* cache line in the row that we observe to exhibit activation failures. We hypothesize that DRAM cells that are subsequently accessed in the same row have enough time to have their charge amplified and completely restored for correct sensing.

We next study the proportion of weak subarray columns per bank across many DRAM banks from all 282 of our DRAM modules. We collect the proportion of weak subarray columns

per bank across two banks from each of our DRAM modules across all three manufacturers. For a given bank, we aggregate the subarray columns that contain activation failures when accessed with reduced t_{RCD} across our full range of temperatures. **Observation 6:** We observe that banks from manufacturers A, B, and C have an average/maximum (standard deviation) proportion of weak subarray columns of 3.7%/96% (12%), 2.5%/100% (6.5%), and 2.2%/37% (4.3%), respectively. We find that on average, banks have a *very low proportion of weak subarray columns*, which means that the memory controller can issue DRAM accesses to *most* subarray columns with reduced t_{RCD}.

We next study how a real workload might be affected by reducing t_{RCD}. We use Ramulator [1,32] to analyze the spatial distribution of accesses immediately following an `ACTIVATE` (i.e., accesses that can induce activation failures) across 20 workloads from the SPEC CPU2006 benchmark suite [2]. Figure 4 shows the probability that the first access to a newly-activated row is to a particular cache line offset within the row. For a given cache line offset (x-axis value), the probability is presented as a distribution of probabilities, found

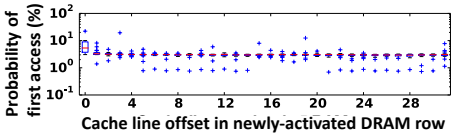


Figure 4: Probability of the first access to a newly-activated row going to a particular cache line offset within the row.

across the SPEC CPU2006 workloads. Each distribution of probabilities is shown as a box-and-whisker plot[1] where the probability (y-axis) is logarithmically scaled. **Observation 7:** A significant proportion of first accesses to a newly-activated DRAM row requests the 0^{th} cache line in the row, with a maximum (average) proportion of 22.2% (6.6%). This indicates that simply reducing t_{RCD} for all accesses to *only* the 0^{th} cache line of each DRAM row can significantly improve overall system performance. We hypothesize that the 0^{th} cache line is accessed with a significantly higher probability due to a significant number of streaming accesses to DRAM rows in our evaluated workloads. Streaming accesses would result in accesses first to the 0^{th} cache line of a newly-activated row followed by accesses to the remaining cache lines in the row in a consecutive manner.

5.2. Data Pattern Dependence

To understand the effects of DRAM data patterns on activation failures in local bitlines, we analyze the number of local bitlines containing activation failures with different data patterns written to the DRAM array. Similar to prior works [40, 52] that extensively describe DRAM data patterns, we study a total of 40 unique data patterns: solid 1s, checkered, row stripe, column stripe, 16 walking 1s, *and* the inverses of all 20 aforementioned data patterns.

Figure 5 shows the *cumulative* number of unique local bitlines containing activation failures over 16 iterations with different data patterns across representative DRAM modules from three DRAM manufacturers. This data was gathered with 100 iterations of Algorithm 1 per data pattern, but we

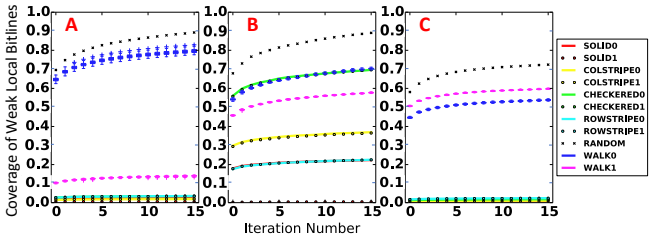


Figure 5: Data pattern dependence of the proportion of local bitlines with activation failures found over 16 iterations.

present only the first 16 iterations to highlight the accumulation rate of local bitlines with failures in earlier iterations. For a given iteration, we calculate the *coverage of each data pattern* as:

$$\frac{\sum_{n=1}^{x} unique\_local\_bitlines(data\_pattern, iteration_n)}{total\_local\_bitlines\_with\_failures} \quad (1)$$

where *unique_local_bitlines*() is the number of local bitlines observed to contain failures in a given iteration but *not* observed to contain failures in any prior iteration when using a specific data pattern, and *total_local_bitlines_with_failures* is the total number of unique local bitlines observed to contain failures at *any* iteration, with *any* data pattern. The *coverage* of a single data pattern indicates the effectiveness of that data pattern to identify the full set of local bitlines containing activation-failure-prone DRAM cells. **Observation 8:** Each walking pattern in a set of WALK1s or WALK0s (i.e., 16 walking 1 patterns and their inverses) finds a similar coverage of local bitlines over many iterations. Given Observation 8, we have already simplified Figure 5 by grouping the set of 16 walking 1 patterns and plotting the distribution of coverages of the patterns as a box-and-whisker-plot (WALK1). We have done the same for the set of 16 walking 0 patterns (WALK0). **Observation 9:** The random data pattern exhibits the highest coverage of activation-failure-prone local bitlines across all three DRAM manufacturers. We hypothesize that the random data results in, on average across DRAM cells, the worst-case coupling noise of a DRAM cell and its neighbors. This is consistent with prior works' experimental observations that the random data pattern causes the highest rate of charge leakage in cells [24, 40, 52].

5.3. Temperature Effects

We next study the effect of DRAM temperature (at the granularity of $5^{\circ}C$) on the number of activation failures across a DRAM module (at reduced t_{RCD}). We make similar observations as prior work [6] and see no clear correlation between the *total number* of activation failures across a DRAM device and DRAM temperature. However, when we analyze the activation failure rates at the granularity of a *local bitline*, we

[1] A box-and-whisker plot emphasizes the important metrics of a dataset's distribution. The box is lower-bounded by the first quartile (i.e., the median of the first half of the ordered set of data points) and upper-bounded by the third quartile (i.e., the median of the second half of the ordered set of data points). The median falls within the box. The *inter-quartile range* (IQR) is defined as the distance between the first and third quartiles, or the size of the box. Whiskers extend an additional $1.5 \times IQR$ on either side of the box. We indicate outliers, or data points outside of the whiskers, with pluses.

observe correlations between DRAM temperature and the number of activation failures in a *local bitline*.

To determine the effect of temperature on a local bitline's probability to contain cells with activation failures, we study activation failures on a local bitline granularity with a range of temperatures. For a set of $5^{\circ}C$ intervals of DRAM temperature between $55^{\circ}C$ and $70^{\circ}C$, we run 100 iterations of Algorithm 1, recording each cell's probability of failure across all our DRAM modules. We indicate a *local bitline's probability of failure (F_{prob})* as:

$$F_{prob} = \sum_{n=1}^{cells\_in\_SA\_bitline} \frac{num\_iters\_failed_{cell_n}}{num\_iters \times cells\_in\_SA\_bitline} \quad (2)$$

where *cells_in_SA_bitline* indicates the number of cells in a local bitline, $num\_iters\_failed_{cell_n}$ indicates the number of iterations out of the 100 tested iterations in which $cell_n$ fails, and *num_iters* is the total number of iterations that the DRAM module is tested for.

Figure 6 aggregates our data across 30 DRAM modules from each DRAM manufacturer. Each point in the figure represents the F_{prob} of a local bitline at temperature T on the x-axis (i.e., the baseline temperature) and the F_{prob} of the same local bitline at temperature $T + 5$ on the y-axis (i.e., $5^{\circ}C$ above the baseline temperature). The F_{prob} values at the

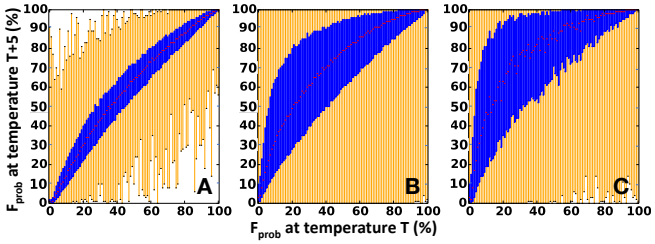


Figure 6: Temperature effects on a local bitline's F_{prob}.

baseline temperature are binned at the granularity of 1% and represent the range of $F_{prob} \pm 0.5\%$. We aggregate the F_{prob} values at temperature $T + 5$ for every local bitline whose F_{prob} at temperature T falls within the same bin on the x-axis. We aggregate each set of F_{prob} values with box-and-whisker plots to show how the F_{prob} is generally affected by increasing the temperature. We draw each box-and-whisker plot with a blue box, orange whiskers, black whisker ends, and red medians. **Observation 10:** We observe that F_{prob} at temperature $T + 5$ tends to be higher than F_{prob} at temperature T (i.e., the blue region of the figure is above the $x = y$ line). Thus, F_{prob} tends to increase with increased temperature. However, there are cases (i.e., <25% of all data points) where the F_{prob} decreases with an increased temperature. We conclude that in order to find a comprehensive set of weak subarray columns, we must profile for activation failures with a range (e.g., 40°C to 55°C) of DRAM temperatures.

5.4. Latency Effects

We next study the effects of changing the value of t_{RCD} on activation failures. We sweep t_{RCD} between 2ns and 18ns (default) at the coarse granularity of 2ns, and we study the correlation of t_{RCD} with the total number of activation failures. We make *two* observations analogous to those made by Chang et al. [6]. **Observation 11:** We observe *no* activation failures when using t_{RCD} values above 14ns regardless of the temperature. The first t_{RCD} at which activation failures occur is 4ns below manufacturer-recommended values. This demonstrates the additional *guardband* that manufacturers place to account for process variation. **Observation 12:** We observe that a small reduction (i.e., by 2ns) in t_{RCD} results in a significant increase (>10x) in the number of activation failures.

In addition to repeating analyses on older generation modules [6], we are the first to study the effects of changing the t_{RCD} value on the failure probability of an individual cell. **Observation 13:** We observe that, if a DRAM cell fails 100% of the time when accessed with a reduced t_{RCD} of n, the same cell will likely fail between 0% and 100% when t_{RCD} is set to $n + 2$, and 0% of the time when t_{RCD} is set to $n + 4$. We hypothesize that the large changes in activation failure probability is due to the coarse granularity with which we can change t_{RCD} (i.e., 2ns; due to experimental infrastructure limitations). For this very reason, we cannot observe gradual changes in the activation failure probability that we expect would occur at smaller intervals of t_{RCD}. We leave the exploration of correlating finer granularity changes of t_{RCD} with the probability of activation failure of a DRAM cell to future work.

5.5. Short-term Variation

Many previous DRAM retention characterization works [3, 9, 16, 22, 23, 24, 25, 26, 27, 29, 37, 40, 44, 52, 53, 54, 56, 68, 69, 71] have shown that there is a well-known phenomenon called variable retention time (VRT), where variation occurs *over time* in DRAM circuit elements that results in significant and sudden changes in the leakage rates of charge from a DRAM cell. This affects the retention time of a DRAM cell over short-term intervals, resulting in varying retention failure probabilities for a given DRAM cell over the span of minutes or hours. To see if a similar time-based variation phenomenon affects the probability of an *activation failure*, we sample the F_{prob} of many local bitlines every six hours over 14 days and study how F_{prob} changes across the samples for a given local bitline. Figure 7 plots the change in F_{prob} of a given local bitline from one time sample to another. For a given local bitline, every pair of sample F_{prob} values (across the 14 day study) are plotted as (x,y) pairs. We collect these data points across all local bitlines in 30 DRAM modules (10 of each DRAM manufacturer) and plot the points. All points sharing the same F_{prob} on the x-axis, are aggregated into box-and-whisker plots. **Ob-**

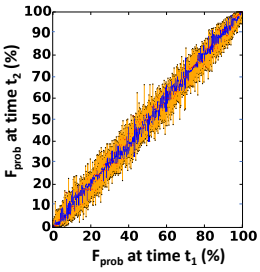


Figure 7: F_{prob} of local bitlines across time.

servation 14: We find that the box-and-whisker plots show a tight distribution around the diagonal axis (where x equals y). This indicates that the F_{prob} of a given local bitline remains highly similar (*correlation* $r = 0.94$) across time. *This means that a weak local bitline is very likely to remain weak and a strong local bitline is very likely to remain strong across time.* Thus, we can identify the set of weak local bitlines *once* and that set would remain *constant* across time. To determine the number of iterations we expect to profile for to find a comprehensive set of weak local bitlines, we run iterations of Algorithm 1 for each bank until we only observe either zero or one failing bit in a local bitline that has never been observed to fail before in the tested bank. At this point, we say that we have found the *entire* set of local bitlines containing activation failures. **Observation 15:** We find that the required number of iterations to find the entire set of local bitlines containing activation failures differs significantly across chips and manufacturers. The average/maximum (standard deviation) number of iterations required to find the entire set of local bitlines for manufacturers A, B, and C is 843/1411 (284.28), 162/441 (174.86), and 1914/1944 (26.28), respectively.

5.6. DRAM Write Operations

We next study the effects of reduced t_{RCD} on *write* operations. We hypothesize that t_{RCD} is *mostly unnecessary* for DRAM write operations, because t_{RCD} dictates the time required for the sense amplifiers to amplify the data in DRAM cells to an *I/O readable value* (V_{access}) such that reads can be correctly serviced. To determine the effects of reducing t_{RCD} on DRAM write operations, we run two experiments with our DRAM modules. First, we sweep the value of t_{RCD} between 2ns and 18ns, and write a known data pattern across DRAM. We then read every value in the DRAM array with the default t_{RCD} and compare each read value with the expected value. We repeat this process 100 times using the *random* data pattern for each of our DRAM modules. We observe activation failures only when t_{RCD} is set below 4ns. We conclude that we can reliably issue DRAM write operations to our LPDDR4 DRAM modules with a *significantly* reduced t_{RCD} (i.e., 4ns; a reduction of 77%) without loss of data integrity.

6. Exploiting Activation Latency Variation

Based on our key observations from our extensive characterization of activation latency failures in DRAM (Section 5), we propose *Subarray-optimized Access Latency Reduction DRAM (Solar-DRAM)*, a mechanism that robustly reduces t_{RCD} for both DRAM read and write requests.

6.1. Solar-DRAM

Solar-DRAM consists of three components that exploit various observations on activation failures and memory access patterns. These three components are pure hardware approaches implemented within the memory controller without any DRAM changes and are invisible to applications.

Component I: Variable-latency cache lines (VLC). The first key observation that we exploit is that activation failures are highly constrained to *some* (or few) *local bitlines* (i.e., only 3.7%/2.5%/2.2% of subarray columns per bank are weak on average for DRAM manufacturers A/B/C respectively. See Section 5.1), and the local bitlines with activation-failure-prone cells are *randomly* distributed across the chip (not shown). Given the known spatial distribution of activation failures, the memory controller can issue memory requests with varying activation latency depending on whether or not the access is to data contained in a "weak" local bitline. To enable such a mechanism, Solar-DRAM requires the use of a *weak subarray column profile* that identifies local bitlines as either *weak* or *strong*. However, since activation failures affect DRAM only at the granularity of a cache line (Section 5.1), Solar-DRAM needs to only store whether or not a column of cache-line-aligned DRAM cells within a subarray, i.e., a *subarray column*, contains a weak local bitline.

The second key observation that we exploit is that the failure probability of a cell, when accessed with a reduced t_{RCD}, is *not* vulnerable to short-term time variation (Section 5.5). This novel observation is necessary to ensure that a profile of weak local bitlines will *not change over time* and thus allows Solar-DRAM to rely on a static profile.[2]

Given a static profile of weak subarray columns, we can safely access the weak subarray columns with the default t_{RCD}, and *all other* subarray columns with a reduced t_{RCD}. We observe that after finding the initial set of failing columns there is still a very low probability (i.e., $< 5 \times 10^{-7}$) that a strong column will result in a single error. Fortunately, we find this probability to be low enough such that employing error correction codes (ECC) [4, 14, 23, 50], which are already present in modern DRAM chips, would transparently mitigate low-probability activation failures in strong columns.

Component II: Reordered subarray columns (RSC). We observe in Section 5.1, that the memory controller accesses the 0^{th} cache line of a newly-activated DRAM row with the highest probability compared to the rest of the cache lines. Thus, we would like to devise a mechanism that reduces access latency (i.e., t_{RCD}) *specifically to the* 0^{th} *cache line in each row* because the first accessed cache line in a newly-activated row is most affected by t_{RCD}. To this end, we propose a mechanism that scrambles column addresses such that the 0^{th} cache line in a row is *unlikely* to get mapped to weak subarray columns. Given a weak subarray column profile, we identify the *global column* (i.e., the column of cache-line-aligned DRAM cells across a full DRAM bank) containing the fewest weak subarray columns, called the *strongest global column.* We then scramble the column address bits such that the 0^{th} cache line for each bank maps to the *strongest global column* in the bank. We perform this scrambling by changing the DRAM address mapping at the granularity of the global column, in order to to reduce the overhead in address scrambling.

Component III: Reduced latency for writes (RLW). The final observation that we exploit in Solar-DRAM is that write operations do *not* require the default t_{RCD} value (Section 5.6). To exploit this observation, we use a reliable, re-

[2] We acknowledge that we do *not* consider long-term variation that may arise from aging or wearout of circuit components. We leave this exploration to future work. Such long-term effects can have implications for a static profile (as discussed in DIVA-DRAM [37]), but one can devise a mechanism that updates the profile at regular long time intervals with low overhead, e.g., as in prior work [52, 53].

duced t_{RCD} (i.e., 4ns, as measured with our experimental infrastructure) for *all* write operations to DRAM.

6.2. Static Profile of Weak Subarray Columns

To obtain the static profile of weak subarray columns, we run multiple iterations of Algorithm 1, recording all subarray columns containing observed activation failures. As we observe in Section 5, there are various factors that affect a local bitline's probability of failure (F_{prob}). We use these factors to determine a method for identifying a comprehensive profile of weak subarray columns for a given DRAM module. First, we use our observation on the accumulation rate of finding weak local bitlines (Section 5.5) to determine the number of iterations we expect to test each DRAM module. However, since there is such high variation across each DRAM module (as seen in the standard deviations of the distributions in Observation 11), we can only provide the expected number of iterations needed to find a comprehensive profile for DRAM modules of a manufacturer, and the time to profile depends on the module. We show in Section 5.2 that no *single* data pattern alone finds a high coverage of weak local bitlines. This indicates that we must test each data pattern (40 data patterns) for the expected number of iterations needed to find a comprehensive profile of a DRAM module for a range of temperatures (Section 5.3). While this could result in many iterations of testing (on the order of a few thousands; see Section 5.5), this is a one-time process on the order of half a day per bank that results in a reliable profile of weak subarray columns. The required one-time profiling can be performed in two ways: 1) the system running Solar-DRAM can profile a DRAM module when the memory controller detects a new DRAM module at bootup, or 2) the DRAM manufacturer can profile each DRAM module and provide the profile within the *Serial Presence Detect* (SPD) circuitry (a Read-Only Memory present in each DIMM) [20].

To minimize the storage overhead of the weak subarray column profile in the memory controller, we encode each subarray column with a bit indicating whether or not to issue accesses to it with a reduced t_{RCD}. After profiling DRAM, the memory controller loads the weak subarray column profile once into a small lookup table in the DRAM channel's memory controller.[3] For any DRAM request, the memory controller references the lookup table with the subarray column that is being accessed. The memory controller determines the t_{RCD} timing parameter according to the value of the bit found in the lookup table.

7. Solar-DRAM Evaluation

We first discuss our evaluation methodology and evaluated system configurations. We then present our multi-core simulation results for our chosen system configurations.

[3]To store the lookup table for a DRAM channel, we require $num\_banks \times num\_subarrays\_per\_bank \times \frac{row\_size}{cacheline\_size}$ bits, where *num_subarrays_per_bank* is the number of subarrays in a bank, *row_size* is the size of a DRAM row in bits, and *cacheline_size* is the size of a cache line in bits. For a 4GB DRAM module with 8 banks, 64 subarrays per bank, 32-byte cache lines, and 2KB per row, the lookup table requires 4KB of storage.

7.1. Evaluation Methodology

System Configurations. We evaluate the performance of Solar-DRAM on a 4-core system using Ramulator [1, 32], an open-source cycle-accurate DRAM simulator, in CPU-trace-driven mode. We analyze various real workloads with traces from the SPEC CPU2006 benchmark [2] that we collect using Pintool [43]. Table 1 shows the configuration of our evaluated system. We use the standard LPDDR4-3200 [18] timing parameters as our baseline. To give a conservative estimate of Solar-DRAM's performance improvement, we simulate with a 64B cache line and a subarray size of 1024 rows.[4]

Processor	4 cores, 4 GHz, 4-wide issue, 8 MSHRs/core, OoO 128-entry window
LLC	8 *MiB* shared, 64B cache line, 8-way associative
Memory Controller	64-entry R/W queue, FR-FCFS [55, 74]
DRAM	LPDDR4-3200 [18], 2 channels, 1 rank/channel, 8 banks/rank, 64K rows/bank, 1024 rows/subarray, 8 *KiB* row-buffer, Baseline: $t_{RCD}/t_{RAS}/t_{WR}$ = 29/67/29 cycles (18.125/41.875/18.125 ns)
Solar-DRAM	reduced t_{RCD} for requests to strong cache lines: 18 cycles (11.25ns) reduced t_{RCD} for write requests: 7 cycles (4.375ns)

Table 1: Evaluated system configuration.

Solar-DRAM Configuration. To evaluate Solar-DRAM and FLY-DRAM [6] on a variety of different DRAM modules with unique properties, we simulate varying 1) the number of weak subarray columns per bank between n = 1 *to* 512, and 2) the chosen weak subarray columns in each bank. For a given n, i.e., weak subarray column count, we generate 10 *unique* profiles with n randomly chosen weak subarray columns per bank. The profile indicates whether a subarray column should be accessed with the default t_{RCD} (29 cycles; 18.13 ns) or the reduced t_{RCD} (18 cycles; 11.25 ns). We use these profiles to evaluate 1) Solar-DRAM's three components (described in Section 6.1) independently, 2) Solar-DRAM with all its three components, 3) FLY-DRAM [6], and 4) our baseline LPDDR4 DRAM.

Variable latency cache lines (VLC), directly uses a weak subarray column profile to determine whether an access should be issued with a reduced or default t_{RCD} value. *Reordered subarray columns* (RSC) takes a profile and maps the 0^{th} cache line to the *strongest global column* in each bank. For a given profile, this maximizes the probability that any access to the 0^{th} cache line of a row will be issued with a reduced t_{RCD}. *Reduced latency for writes* (RLW) reduces t_{RCD} to 7 cycles (4.38 ns) (Section 5.6) for *all* write operations to DRAM. *Solar-DRAM* (Section 6.1) combines all three components (*VLC*, *RSC*, and *RLW*). Since *FLY-DRAM* [6] issues read requests at the granularity of the *global column* depending on whether a global column contains weak bits, we evaluate FLY-DRAM by taking a weak subarray column profile and extending each weak subarray column to the global column containing it. Baseline LPDDR4 uses a fixed t_{RCD} of 29 cycles (18.13 ns) for all accesses. We present performance improvement of the different mechanisms over this LPDDR4 baseline.

[4]Using the typical upper-limit values for these configuration variables reduces the total number of subarray columns that comprise DRAM (to 8,192 subarray columns per bank). A smaller number of subarray columns reduces the granularity at which we can issue DRAM accesses with reduced t_{RCD}, which reduces Solar-DRAM's potential for performance benefit. This is because a single activation failure requires the memory controller to access *larger* regions of DRAM with default t_{RCD}.

7.2. Multi-core Evaluation Results

Figure 8 plots the improvement in weighted speedup [66], which corresponds to system throughput [12], over the baseline on 20 homogeneous mixes of 4-core workloads and 20 heterogeneous mixes of 4-core workloads randomly combined from the set of workloads in the SPEC CPU2006 benchmark suite [2]. For each configuration of <*weak subarray column count, weak subarray column profile, mechanism, workload mix*>, we aggregate all weighted speedup improvement results into a box-and-whisker plot.

We make four key observations. First, Solar-DRAM provides significant weighted speedup improvement. Even when half of the subarray columns are classified as weak (which is very unrealistic and conservative, as our experiments on real DRAM modules show), Solar-DRAM improves performance by 4.03% (7.71%) for heterogeneous and 3.36% (8.80%) for homogeneous workloads. In the ideal case, where there are 0 weak subarray columns per bank and thus, the memory controller issues *all* memory accesses with a reduced t_{RCD}, Solar-DRAM improves performance by 4.97% (8.79%) for heterogeneous and 4.31% (10.87%) for homogeneous workloads. Second, each individual component of Solar-DRAM improves system performance. *RLW* is the best alone: it improves performance by 2.92% (5.90%) for heterogeneous and 2.25% (6.59%) for homogeneous workloads. Because *RLW* is independent of the number of weak subarray columns in a bank, its weighted speedup improvement is constant regardless of the number of weak subarray columns per bank. Third, Solar-DRAM provides higher performance improvement than each of its components, demonstrating that the combination of *VLC*, *RSC*, and *RLW* is *synergistic*. Fourth, Solar-DRAM provides much higher performance improvement than FLY-DRAM. This is because Solar-DRAM 1) exploits the observation that *all write requests* can be issued with a greatly reduced t_{RCD} (i.e., by 77%), and 2) issues read requests with reduced t_{RCD} at the granularity of the *local* bitline rather than the global bitline. This means that for a single weak cache line in a subarray, Solar-DRAM issues read requests with default t_{RCD} *only* to cache lines in the *subarray column* containing the weak cache line, while FLY-DRAM would issue read requests with default t_{RCD} to *all* cache lines in the column across the *full bank*. For this very same reason, we also observe that *VLC* alone outperforms FLY-DRAM. Fourth, Solar-DRAM enables significantly higher performance improvement on DRAM modules with a high rate of activation failures, where FLY-DRAM provides no benefit. Because FLY-DRAM categorizes columns across the *entire bank* as strong or weak, even a low activation failure rate across the DRAM chip results in a high number of cache lines requiring the default t_{RCD} timing parameter in FLY-DRAM. We experimentally observe the average proportion of weak subarray columns per bank to be 3.7%/2.5%/2.2% for DRAM manufacturers A/B/C (Section 5.1). Even at such a low proportion of weak subarray columns (i.e., 38/26/23 subarray columns out of 1024 subarray columns in our evaluated DRAM configuration), we expect the performance benefit of FLY-DRAM to be well below 1.6% (i.e., the median performance benefit when we evaluate FLY-DRAM with 16 weak subarray columns in Figure 8 across all workload mixes) for DRAM manufacturers B and C, and 0% for DRAM manufacturer A. We conclude that Solar-DRAM's three components provide significant performance improvement on modern LPDDR4 DRAM modules over LPDDR4 DRAM and FLY-DRAM.

8. Related Work

Many works seek to improve DRAM access latency. They can be classified according to the mechanisms they take advantage of, as follows.

Static Variation. We have already described these works [6, 37] in detail in Section 3 and compared to FLY-DRAM [6] in Section 7. Solar-DRAM outperforms FLY-DRAM. Das et al. [10] propose a method to reduce *refresh latency*, which is orthogonal to Solar-DRAM.

Operational Factors. Prior works improve DRAM latency by controlling or taking advantage of changes in operational factors such as temperature [35] and voltage [9]. These works are orthogonal to Solar-DRAM since they reduce latency in response to changes in factors that are independent of latency variations inherent to the DRAM module.

Access Locality. Some work exploits locality in DRAM access patterns [15, 65, 70] and reorganizes DRAM accesses to allow for higher locality [33, 48, 59, 64] in order to reduce average DRAM access latency. These can be combined with Solar-DRAM for further latency reduction.

Modifications to DRAM Architecture. Various works [7, 8, 17, 31, 38, 42, 57, 60, 61, 62, 63, 67, 72] propose mechanisms that change the structure of DRAM to reduce latency. Solar-DRAM requires *no* changes to the DRAM chip.

Software Support. Several works [11, 21, 34, 51] propose using compile-time optimizations to improve DRAM access locality and thus, decrease overall DRAM access latency. Solar-DRAM reduces the latency of the average memory access and would provide added benefits to software optimizations. If the profile of weak subarray columns is exposed to the compiler or the system software, the software could potentially use this device-level information to allocate latency-critical data at *stronger* locations in DRAM, while decreasing the hardware overhead of storing weak subarray column profiles in the memory controller.

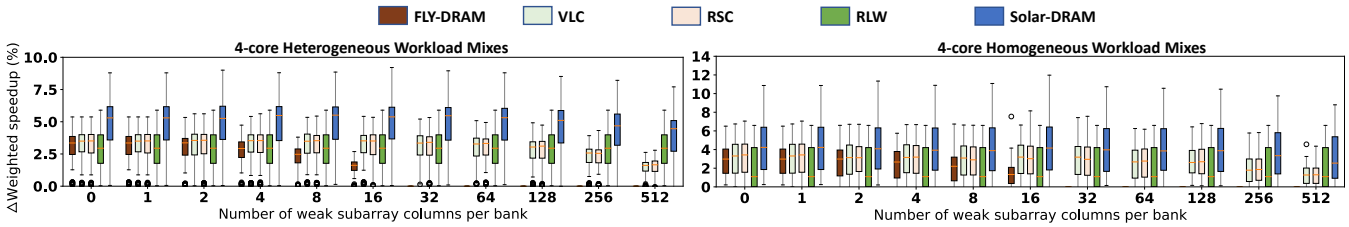


Figure 8: Weighted speedup improvements of Solar-DRAM, its three individual components, and FLY-DRAM over baseline LPDDR4 DRAM, evaluated over various 4-core workload mixes from the SPEC CPU2006 benchmark suite.

9. Conclusion

We introduced 1) a rigorous characterization of activation failures across 282 *real state-of-the-art LPDDR4* DRAM modules, 2) Solar-DRAM, whose key idea is to exploit our observations and issue DRAM accesses with variable latency depending on the target DRAM location's propensity to fail with reduced access latency, and 3) an evaluation of Solar-DRAM and its three individual components, with comparisons to the state-of-the-art [6]. We find that Solar-DRAM provides significant performance improvement over the state-of-the-art DRAM latency reduction mechanism across a wide variety of workloads, *without* requiring any changes to DRAM chips or software.

Acknowledgments

The authors thank the anonymous reviewers for feedback and the SAFARI group members for feedback and the stimulating intellectual environment they provide.

References

[1] "Ramulator Source Code," https://github.com/CMU-SAFARI/ramulator.
[2] "Standard Performance Evaluation Corporation," http://www.spec.org/cpu2006.
[3] S. Baek *et al.*, "Refresh Now and Then," in *TC*, 2014.
[4] S. Cha *et al.*, "Defect Analysis and Cost-effective Resilience Architecture for Future DRAM Devices," in *HPCA*, 2017.
[5] K. K. Chang, "Understanding and Improving Latency of DRAM-Based Memory Systems," Ph.D. dissertation, Carnegie Mellon University, 2017.
[6] K. K. Chang *et al.*, "Understanding Latency Variation in Modern DRAM Chips: Experimental Characterization, Analysis, and Optimization," in *SIGMETRICS*, 2016.
[7] K. K. Chang *et al.*, "Improving DRAM Performance by Parallelizing Refreshes with Accesses," in *HPCA*, 2014.
[8] K. K. Chang *et al.*, "Low-cost Inter-linked Subarrays (LISA): Enabling Fast Inter-subarray Data Movement in DRAM," in *HPCA*, 2016.
[9] K. K. Chang *et al.*, "Understanding Reduced-Voltage Operation in Modern DRAM Devices: Experimental Characterization, Analysis, and Mechanisms," in *SIGMETRICS*, 2017.
[10] A. Das *et al.*, "VRL-DRAM: Improving DRAM Performance via Variable Refresh Latency," *DAC*, 2018.
[11] W. Ding *et al.*, "Compiler Support for Optimizing Memory Bank-level Parallelism," in *MICRO*, 2014.
[12] S. Eyerman and L. Eeckhout, "System-level Performance Metrics for Multiprogram Workloads," in *IEEE Micro*, 2008.
[13] S. Ghose *et al.*, "What Your DRAM Power Models Are Not Telling You: Lessons from a Detailed Experimental Study," *SIGMETRICS*, 2018.
[14] R. W. Hamming, "Error Detecting and Error Correcting Codes," in *Bell Labs Technical Journal*, 1950.
[15] H. Hassan *et al.*, "ChargeCache: Reducing DRAM Latency by Exploiting Row Access Locality," in *HPCA*, 2016.
[16] H. Hassan *et al.*, "SoftMC: A Flexible and Practical Open-source Infrastructure for Enabling Experimental DRAM Studies," in *HPCA*, 2017.
[17] H. Hidaka *et al.*, "The Cache DRAM Architecture: A DRAM with an on-chip Cache Memory," *MICRO*, 1990.
[18] JEDEC, "Low Power Double Data Rate 3 (LPDDR4)," Standard No. JESD209-4, 2014.
[19] JEDEC, "LPDDR4," *JEDEC Standard JESD209–4A*, 2014.
[20] JEDEC, "Annex L: Serial Presence Detect (SPD) for DDR4 SDRAM Modules," 2015.
[21] M. K. Jeong *et al.*, "Balancing DRAM Locality and Parallelism in Shared Memory CMP Systems," in *HPCA*, 2012.
[22] M. Jung *et al.*, "Reverse Engineering of DRAMs: Row Hammer with Crosshair," in *MEMSYS*, 2016.
[23] U. Kang *et al.*, "Co-Architecting Controllers and DRAM to Enhance DRAM Process Scaling," in *The Memory Forum*, 2014.
[24] S. Khan *et al.*, "The Efficacy of Error Mitigation Techniques for DRAM Retention Failures: A Comparative Experimental Study," in *SIGMETRICS*, 2014.
[25] S. Khan *et al.*, "PARBOR: An Efficient System-Level Technique to Detect Data-Dependent Failures in DRAM," in *DSN*, 2016.
[26] S. Khan *et al.*, "A Case for Memory Content-Based Detection and Mitigation of Data-Dependent Failures in DRAM," in *CAL*, 2016.
[27] S. Khan *et al.*, "Detecting and Mitigating Data-Dependent DRAM Failures by Exploiting Current Memory Content," in *MICRO*, 2017.
[28] J. S. Kim *et al.*, "The DRAM Latency PUF: Quickly Evaluating Physical Unclonable Functions by Exploiting the Latency-Reliability Tradeoff in Modern Commodity DRAM Devices," in *HPCA*, 2018.
[29] Y. Kim *et al.*, "Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors," in *ISCA*, 2014.
[30] Y. Kim *et al.*, "Thread Cluster Memory Scheduling: Exploiting Differences in Memory Access Behavior," in *MICRO*, 2010.
[31] Y. Kim *et al.*, "A Case for Exploiting Subarray-level Parallelism (SALP) in DRAM," in *ISCA*, 2012.
[32] Y. Kim *et al.*, "Ramulator: A Fast and Extensible DRAM Simulator," in *CAL*, 2016.
[33] C. J. Lee *et al.*, "DRAM-Aware Last-Level Cache Writeback: Reducing Write-Caused Interference in Memory Systems," *HPS Technical Report*, 2010.
[34] C. J. Lee *et al.*, "Improving Memory Bank-level Parallelism in the Presence of Prefetching," in *MICRO*, 2009.
[35] D. Lee *et al.*, "Adaptive-latency DRAM: Optimizing DRAM Timing for the Common-case," in *HPCA*, 2015.
[36] D. Lee, "Reducing DRAM Latency at Low Cost by Exploiting Heterogeneity," Ph.D. dissertation, Carnegie Mellon University, 2016.
[37] D. Lee *et al.*, "Design-Induced Latency Variation in Modern DRAM Chips: Characterization, Analysis, and Latency Reduction Mechanisms," in *SIGMETRICS*, 2017.
[38] D. Lee *et al.*, "Tiered-Latency DRAM: A Low Latency and Low Cost DRAM Architecture," in *HPCA*, 2013.
[39] D. Lee *et al.*, "Decoupled Direct Memory Access: Isolating CPU and IO Traffic by Leveraging a Dual-Data-Port DRAM," in *PACT*, 2015.
[40] J. Liu *et al.*, "An Experimental Study of Data Retention Behavior in Modern DRAM Devices: Implications for Retention Time Profiling Mechanisms," in *ISCA*, 2013.
[41] J. Liu *et al.*, "RAIDR: Retention-Aware Intelligent DRAM Refresh," in *ISCA*, 2012.
[42] S.-L. Lu *et al.*, "Improving DRAM Latency with Dynamic Asymmetric Subarray," in *MICRO*, 2015.
[43] C.-K. Luk *et al.*, "Pin: Building Customized Program Analysis Tools with Dynamic Instrumentation," in *PLDI*, 2005.
[44] Y. Mori *et al.*, "The Origin of Variable Retention Time in DRAM," in *IEDM*, 2005.
[45] S. P. Muralidhara *et al.*, "Reducing Memory Interference in Multicore Systems via Application-aware Memory Channel Partitioning," in *MICRO*, 2011.
[46] O. Mutlu, "Memory Scaling: A Systems Architecture Perspective," in *IMW*, 2013.
[47] O. Mutlu and T. Moscibroda, "Stall-Time Fair Memory Access Scheduling for Chip Multiprocessors," in *MICRO*, 2007.
[48] O. Mutlu and T. Moscibroda, "Parallelism-aware Batch Scheduling: Enhancing both Performance and Fairness of Shared DRAM Systems," in *ISCA*, 2008.
[49] O. Mutlu and L. Subramanian, "Research Problems and Opportunities in Memory Systems," in *SUPERFRI*, 2014.
[50] P. J. Nair *et al.*, "XED: Exposing On-Die Error Detection Information for Strong Memory Reliability," in *ISCA*, 2016.
[51] V. S. Pai and S. Adve, "Code Transformations to Improve Memory Parallelism," in *MICRO*, 1999.
[52] M. Patel *et al.*, "The Reach Profiler (REAPER): Enabling the Mitigation of DRAM Retention Failures via Profiling at Aggressive Conditions," in *ISCA*, 2017.
[53] M. K. Qureshi *et al.*, "AVATAR: A Variable-Retention-Time (VRT) Aware Refresh for DRAM Systems," in *DSN*, 2015.
[54] P. J. Restle *et al.*, "DRAM Variable Retention Time," in *IEDM*, 1992.
[55] S. Rixner *et al.*, "Memory Access Scheduling," in *ISCA*, 2000.
[56] B. Schroeder *et al.*, "DRAM Errors in the Wild: a Large-scale Field Study," in *SIGMETRICS*, 2009.
[57] O. Seongil *et al.*, "Row-buffer Decoupling: A Case for Low-latency DRAM Microarchitecture," in *ISCA*, 2014.
[58] V. Seshadri, "Simple DRAM and Virtual Memory Abstractions to Enable Highly Efficient Memory Systems," Ph.D. dissertation, Carnegie Mellon University, 2016.
[59] V. Seshadri *et al.*, "The Dirty-block Index," in *ISCA*, 2014.
[60] V. Seshadri *et al.*, "Fast Bulk Bitwise AND and OR in DRAM," *CAL*, 2015.
[61] V. Seshadri *et al.*, "RowClone: Fast and Energy-Efficient In-DRAM Bulk Data Copy and Initialization," in *MICRO*, 2013.
[62] V. Seshadri *et al.*, "Buddy-RAM: Improving the Performance and Efficiency of Bulk Bitwise Operations Using DRAM," in *arXiv*, 2016.
[63] V. Seshadri *et al.*, "Ambit: In-memory Accelerator for Bulk Bitwise Operations Using Commodity DRAM Technology," in *MICRO*, 2017.
[64] V. Seshadri *et al.*, "Gather-scatter DRAM: In-DRAM Address Translation to Improve the Spatial Locality of Non-unit Strided Accesses," in *MICRO*, 2015.
[65] W. Shin *et al.*, "NUAT: A Non-Uniform Access Time Memory Controller," in *HPCA*, 2014.
[66] A. Snavely and D. M. Tullsen, "Symbiotic Jobscheduling for a Simultaneous Mutlithreading Processor," in *ASPLOS*, 2000.
[67] Y. H. Son *et al.*, "Reducing Memory Access Latency with Asymmetric DRAM Bank Organizations," 2013.
[68] V. Sridharan and D. Liberty, "A Study of DRAM Failures in the Field," in *SC*, 2012.
[69] R. K. Venkatesan *et al.*, "Retention-aware Placement in DRAM (RAPID): Software Methods for Quasi-non-volatile DRAM," in *HPCA*, 2006.
[70] Y. Wang *et al.*, "Reducing DRAM Latency via Charge-Level-Aware Look-Ahead Partial Restoration," *MICRO*, 2018.
[71] D. S. Yaney *et al.*, "A Meta-stable Leakage Phenomenon in DRAM Charge Storage-Variable Hold Time," in *IEDM*, 1987.
[72] T. Zhang *et al.*, "Half-DRAM: A High-bandwidth and Low-power DRAM Architecture from the Rethinking of Fine-grained Activation," in *ISCA*, 2014.
[73] Z. Zhang *et al.*, "A Permutation-based Page Interleaving Scheme to Reduce Row-buffer Conflicts and Exploit Data Locality," in *MICRO*, 2000.
[74] W. K. Zuravleff and T. Robinson, "Controller for a Synchronous DRAM that Maximizes Throughput by Allowing Memory Requests and Commands to be Issued Out of Order," US Patent 5,630,096. 1997.