What Your DRAM Power Models Are Not Telling You: Lessons from a Detailed Experimental Study

SAUGATA GHOSE, Carnegie Mellon University
ABDULLAH GIRAY YAĞLIKÇI, ETH Zürich & Carnegie Mellon University
RAGHAV GUPTA, Carnegie Mellon University
DONGHYUK LEE, NVIDIA
KAIS KUDROLLI, Carnegie Mellon University
WILLIAM X. LIU, Carnegie Mellon University
HASAN HASSAN, ETH Zürich
KEVIN K. CHANG, Carnegie Mellon University
NILADRISH CHATTERJEE, NVIDIA
ADITYA AGRAWAL, NVIDIA
MIKE O'CONNOR, NVIDIA & The University of Texas at Austin
ONUR MUTLU, ETH Zürich & Carnegie Mellon University

Main memory (DRAM) consumes as much as half of the total system power in a computer today, due to the increasing demand for memory capacity and bandwidth. There is a growing need to understand and analyze DRAM power consumption, which can be used to research new DRAM architectures and systems that consume less power. A major obstacle against such research is the lack of detailed and accurate information on the power consumption behavior of modern DRAM devices. Researchers have long relied on DRAM power models that are predominantly based off of a set of standardized current measurements provided by DRAM vendors, called IDD values. Unfortunately, we find that state-of-the-art DRAM power models are often highly inaccurate, as these models do *not* reflect the actual power consumed by *real* DRAM devices.

To build an *accurate* model and provide insights into DRAM power consumption, we perform the first comprehensive experimental characterization of the power consumed by modern real-world DRAM modules. Our extensive characterization of 50 DDR3L DRAM modules from three major vendors (A, B, and C) yields four key new observations about DRAM power consumption that prior models cannot capture: (1) across all IDD values that we measure, the current consumed by real DRAM modules *varies significantly* from the current specified by the vendors; (2) DRAM power consumption strongly *depends on the data value* that is read or written; (3) there is significant *structural variation*, where the same banks and rows across multiple DRAM modules from the same model consume more power than other banks or rows; and (4) over successive process technology generations, DRAM power consumption has not decreased by as much as vendor specifications have indicated. Because state-of-the-art DRAM power models do *not* account for any of these four key characteristics, they are highly inaccurate compared to the actual, measured power consumption of 50 real DDR3L modules.

Based on our detailed analysis and characterization data, we develop the *Variation-Aware model of Memory Power Informed by Real Experiments* (VAMPIRE). VAMPIRE is a new, accurate power consumption model for DRAM that takes into account (1) module-to-module and intra-module variations, and (2) power consumption variation due to data value dependency. We show that VAMPIRE has a mean absolute percentage error of only 6.8% compared to actual measured DRAM power. VAMPIRE enables a wide range of studies that were *not* possible using prior DRAM power models. As an example, we use VAMPIRE to evaluate the energy efficiency of three different encodings that can be used to store data in DRAM. We find that a new power-aware data encoding mechanism can reduce total DRAM energy consumption by an average of 12.2%, across a wide range of applications. We plan to open-source both VAMPIRE and our extensive raw data collected during our experimental characterization.

CCS Concepts: • **Hardware** → **Dynamic memory**; **Power estimation and optimization**; **Board- and system-level test**; • **Computing methodologies** → *Model development and analysis*; • **Computer systems organization** → *Architectures*;

Additional Key Words and Phrases: DRAM; memory systems; energy; power consumption; power modeling; experimental characterization; data encoding; low-power design

ACM Reference Format:
S. Ghose et al. 2018. What Your DRAM Power Models Are Not Telling You: Lessons from a Detailed Experimental Study. *Proc. ACM Meas. Anal. Comput. Syst.* (2018), 41 pages.

1 INTRODUCTION

As processor power consumption has been reduced via many techniques designed over multiple decades, main memory, which is built using the dynamic random access memory (DRAM) technology, has now become a significant source of power consumption in modern computer systems. This is because the amount of DRAM in a computer has been increasing over the years, to keep up with the growing demand for memory capacity and bandwidth in modern applications [50, 104, 117, 155]. In a contemporary system, DRAM power consumption accounts for as much as 46% of the *total system power* [38, 48, 56, 93, 104, 122, 157, 163]. In response, vendors have developed several low-power and low-voltage variants of DRAM (e.g., DDR3L [61], LPDDR3 [63] and LPDDR4 [64]), and there has been some research on reducing the power consumed by modern DRAM architectures (e.g., [8, 9, 28, 31, 37, 40–42, 44, 49, 74, 75, 77, 88, 98, 99, 105, 119, 121, 139]). However, as DRAM consumes a growing fraction of the total system power, a much greater effort is necessary to invent new low-power solutions.

One major hindrance towards further research is the relative lack of information available on the low-level power consumption behavior of modern DRAM devices. It has historically been difficult to collect *accurate* power consumption data from real DRAM devices, as computer systems (1) do not offer fine-grained control over commands being issued to DRAM, instead exposing only high-level operations such as loads and stores; and (2) often lack dedicated monitors that track the power consumed by DRAM. As a result, for many years, researchers have instead relied on current specifications that vendors provide for each DRAM part, which are known as *IDD values* [111]. A vendor determines IDD values by using a standardized set of benchmarks that are meant to represent common high-level DRAM operations, such as reading one cache line of data from DRAM, or *refreshing* data (i.e., restoring the charge lost from DRAM cells due to charge leakage). State-of-the-art DRAM power models (e.g., [25, 27, 65, 111]), which researchers currently use to perform DRAM power studies, and which are used by many popular simulators (e.g., [6, 11, 83, 120, 131]), are predominantly based on these IDD values.

We find that state-of-the-art DRAM power models are often highly inaccurate when compared with the power consumed by real DRAM chips. This is because existing DRAM power models (1) are based off of the *worst-case* power consumption of devices, as vendor specifications list the current consumed by the most power-hungry device sold; (2) do *not* capture variations in DRAM power consumption due to different *data value* patterns; and (3) do *not* account for any *variation* across different devices or within a device. Because existing DRAM power models do *not* capture these characteristics, it is often difficult for researchers to accurately (1) identify sources of inefficiency within DRAM; and (2) evaluate the effectiveness of memory energy saving techniques, including new hardware designs and new software mechanisms. *Our goal* in this work is to *rigorously* measure and analyze the power consumption of *real* DRAM devices, and to use our analysis to develop an accurate and detailed DRAM power model, which can be useful for a wide variety of purposes.

To this end, we perform the first extensive characterization of the power consumed by real DRAM devices. To overcome prior obstacles to collecting real power measurements from DRAM, we significantly extend the SoftMC FPGA-based DRAM testing infrastructure [53, 134] to work with high-precision current measurement equipment, which we describe in detail in Section 3. Our testing infrastructure allows us to execute precise test procedures to characterize the effects of (1) intra-chip and inter-chip variation and (2) data dependency on the power consumed by DRAM. We collect detailed power measurement data from 50 DDR3L DRAM modules, comprising of 200 chips, which were manufactured by three major vendors (A, B, and C). Our testing infrastructure allows us to make four key new observations about DRAM power consumption that prior models *cannot* capture:

(1) Across all IDD values that we measure, the current consumed by real DRAM modules varies significantly from the current specified by the vendors (Section 4). For example, to read one cache line of data from DRAM, the measured current of modules from Vendor A is lower than the current specified in the datasheet by an average of 54.1% (up to 61.6%).
(2) DRAM power consumption strongly depends on the data value that is read or written (Section 5). Reading a cache line where all bits are *ones* uses an average of 39.2% (up to 91.6%) more power than reading a cache line where all bits are *zeroes*.
(3) There is significant *structural variation*, where the current varies based on which bank or row is selected in a DRAM module (Section 6). For example, in modules from Vendor C, the idle current consumed when one of the eight banks is active (i.e., open) can vary by an average of 15.4% (up to 23.6%) depending on the bank.
(4) Across successive process technology generations, the actual power reduction of DRAM is *much lower* than the savings indicated by the vendor-specified IDD values in the datasheets (Section 7). Across five key IDD values, the measured savings of modules from Vendor A are lower than indicated by an average of 48.0% (up to 66.7%).

Because state-of-the-art DRAM power models [25, 27, 65, 111] do *not* adequately capture these four key characteristics, their predicted DRAM power is highly inaccurate compared to the actual measured DRAM power of 50 real DDR3L modules. We perform a validation of two state-of-the-art models, DRAMPower [25, 27] and the Micron power model [111], using our FPGA-based current measurement platform, and find that the mean absolute percent error compared to real DRAM power measurements is 32.4% for DRAMPower and 160.6% for the Micron power model.

Building upon our new insights and characterization data, we develop the *Variation-Aware model of Memory Power Informed by Real Experiments* (VAMPIRE). VAMPIRE is a new power model for DRAM, which captures important characteristics such as module-to-module and intra-module variations, and power consumption variation due to data value dependency (Section 9). We show that VAMPIRE is highly accurate: it has a mean absolute percentage error of only 6.8% compared to actual measured DRAM power.

VAMPIRE enables a wide range of studies that were *not* possible using prior DRAM power models. For example, we use VAMPIRE to evaluate the impact of different data encoding mechanisms on DRAM power consumption (Section 10). We find that a new power-aware data encoding technique can reduce DRAM energy by an average of 12.2% (up to 28.6%) across a wide range of applications.

We plan to open-source both VAMPIRE and our extensive raw data collected during our experimental characterization [135]. We hope that our findings and our new power model model will inspire new research directions, new ideas, and rigorous evaluations in power- and energy-aware DRAM design.

We make the following contributions in this work:

- We conduct a detailed and accurate experimental characterization of the power consumed by 50 real, state-of-the-art DDR3L DRAM modules from three major DRAM vendors, comprising 200 DRAM chips. To our knowledge, our characterization is the first to (1) report real power consumption across a wide variety of tests on a large number of DRAM modules from three major

DRAM vendors, (2) comprehensively demonstrate the inter-vendor and intra-vendor module-to-module variation of DRAM power consumption, (3) study the impact of data dependency and structural variation on DRAM power consumption, and (4) examine power consumption trends over multiple product generations.

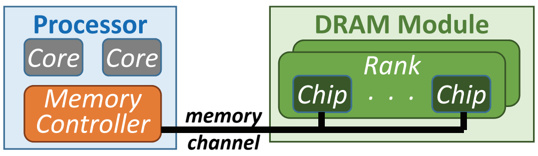
- We make four major new observations based on our measurements, showing that DRAM power consumption (1) varies significantly from the values provided by DRAM vendors, (2) depends on the data value that is read or written, (3) varies based on which bank and which row are used, (4) has not decreased as much as vendor specifications indicate over successive generations.
- We build VAMPIRE, a new DRAM power consumption model based on our new insights and characterization data. VAMPIRE provides significantly greater accuracy than existing power models, and enables studies that were previously not easily possible. We plan to release our power model and our characterization data online [135].

2 BACKGROUND

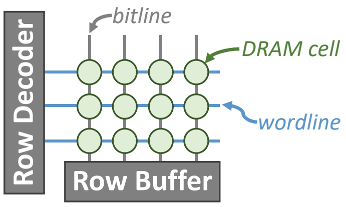
In this section, we first provide necessary DRAM background. We discuss the hierarchical organization of a modern memory system in Section 2.1. We discuss the fundamental operations performed on DRAM in Section 2.2. For a detailed overview of DRAM operation, we refer the reader to our prior works [28–32, 52, 53, 80, 82–84, 87–92, 97, 137–140].

2.1 DRAM Organization

Figure 1a shows the basic overview of a DRAM-based memory system. The memory system is organized in a hierarchical manner. The highest level in the hierarchy is a *memory channel*. Each channel consists of its own bus to the host device, and has a dedicated *memory controller* that interfaces between the DRAM and the host. A channel can connect to one or more *dual inline memory modules* (DIMMs). Each DIMM contains multiple DRAM *chips*. A DRAM row typically spans across several chips, which requires these chips to perform all operations in lockstep with each other. Each group of chips operating in lockstep is known as a *rank*.



(a) Memory system hierarchy



(b) DRAM array structure

Fig. 1. High-level overview of DRAM organization.

Inside each rank, there are several *banks*, where each bank can independently process DRAM commands sent by the memory controller. While each of the banks within a memory channel can operate concurrently, banks share a single memory bus, and, thus, the controller must coordinate the operations across multiple banks in order to avoid interference on the bus. The ability to operate multiple banks concurrently is known as *bank-level parallelism* [81, 86, 116]. DDR3 DRAM typically contains eight banks in each rank [60].

Each bank contains a two-dimensional *array* of DRAM *cells*, as shown in Figure 1b, where each cell stores a single bit of data in a capacitor. Within the array, cells can be selected one row at a time, and the access transistors of the cells in one row are connected together using a *wordline*. Each bank contains a *row buffer*, which consists of a row of sense amplifiers that are used to temporarily buffer the data from a single row in the array during read and write operations. Cells within the array are connected using vertical wires, known as *bitlines*, to the sense amplifiers.

A typical row in a DRAM module, which spans across all of the DRAM chips within a rank, is 8 kB wide, and holds 128 64-byte cache lines of data. For example, in a DDR3 DRAM module with four x16 chips per rank, each chip contains a 2 kB portion of the 8 kB row. Each chip holds a piece of each cache line within the row.

2.2 DRAM Operations

In order to access and update data stored within DRAM, the memory controller issues a series of commands across the memory channel to the DRAM chips. Figure 2 shows the four fundamental DRAM commands: *activate*, *read*, *write*, and *precharge*. We describe each of these commands below.

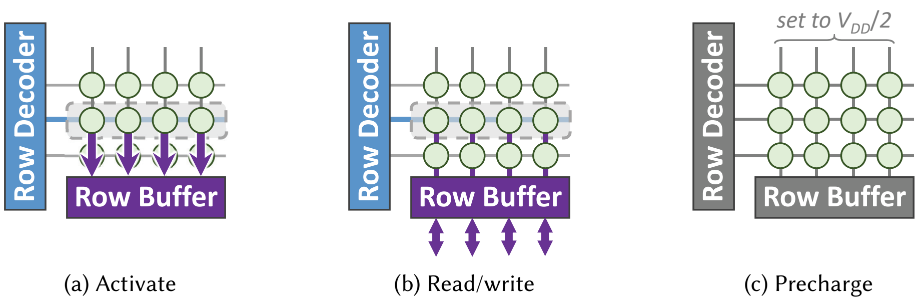


Fig. 2. Overview of fundamental DRAM commands.

Activate. To start processing a request, the controller issues a command to *activate* the row (i.e., open the row to perform reads and writes) within each DRAM chip that contains a part of the desired cache line, as shown in Figure 2a. Initially, each bitline is set to half of V_{DD}, the full supply voltage of the DRAM module. When an activate command is issued, a *row decoder* turns on one of the wordlines on the array, based on the row address of the request. This activates the row of DRAM cells that are connected to the wordline that is turned on. Each DRAM cell in the activated row starts *sharing its charge* with the bitline that the cell is attached to, perturbing the bitline voltage by a small amount. Once the bitline voltage changes by more than a set threshold, the sense amplifier connected to the bitline detects this charge, and amplifies the bitline voltage to either V_{DD} (if the DRAM cell connected to the bitline holds a data value '1') or to 0 V (if the DRAM cell connected to the bitline holds a data value '0'). A latch in the row buffer is enabled to hold the full voltage value.

When a row is activated, the charge sharing process between the cell and the bitline drains charge from the cells in the selected row (i.e., destroys the contents of the cells). This change in cell charge can lead to data corruption if the cell charge is not restored to correspond to the cell's original data value. To avoid such data corruption, the DRAM chip automatically *restores* the charge within the cell back to its starting voltage once the sense amplifier detects the change in the bitline voltage.

Read/Write. Once an activated row is latched into the row buffer, the controller sends *read and write commands* to the row buffer, as shown in Figure 2b. Each read/write command operates on one *column* of data at a time in each chip of a single rank. Across the entire rank, the width of data operated on by a read/write command (i.e., *column width* × *# chips*) is the same width as a processor cache line (64 B). Figure 3 shows the *peripheral circuitry* in a DRAM module that is used by the read and write commands. We walk through the four steps of an example read command as its requested data moves through the peripheral circuitry of one x16 DRAM chip. First, the read command uses the *column select logic* (❶ in Figure 3) to select the 128-bit column (which is one part of the cache line) that the request wants to read. Second, the column is sent over the *global bitline* to the *bank select logic* (❷), which is set by the read command to select the bank that contains the requested cache line. Third, the column is then sent over the *peripheral bus* to the *I/O drivers* (❸). The 128-bit column is split up into eight 16-bit *data bursts*. Across all four x16 chips in our example module, 64 bits of data are sent per data burst. The I/O drivers send the data bursts one at a time across the memory channel, where each wire of the memory channel has its own dedicated I/O driver. In *double data rate* (DDR) DRAM, a burst can be sent on every positive *or* negative DRAM clock edge, allowing the entire cache line to be transmitted in four DRAM clock cycles. Fourth, the bursts are received by the I/O drivers that sit inside the memory controller at the processor (❹). The memory controller reassembles the bursts into a 64-byte cache line, and sends the data to the processor caches. For a write operation, the process is similar, but in the reverse direction: the I/O drivers on the memory controller side send the data across the memory channel.

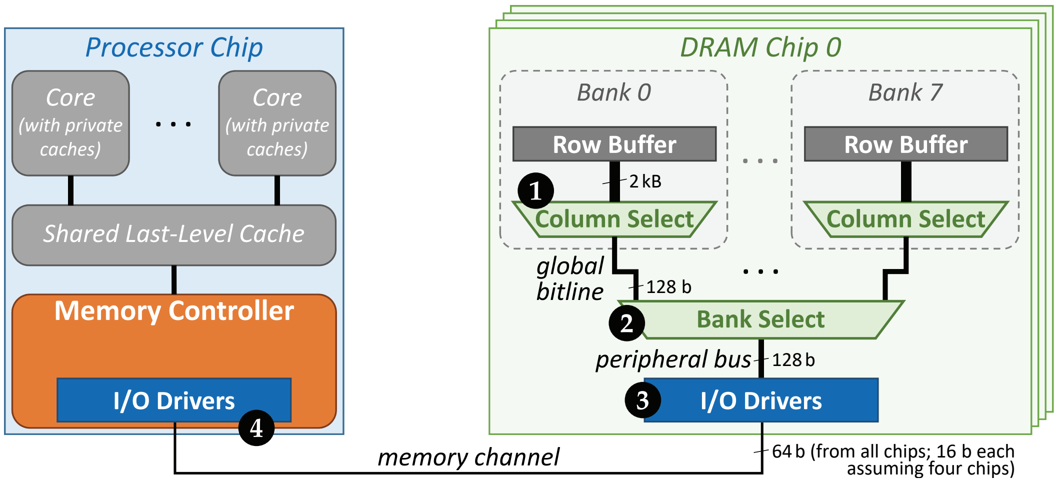


Fig. 3. Overview of peripheral circuitry and I/O in a four-chip DRAM module.

Precharge. Once the read and write operations to the row are complete, the controller issues a *precharge* command, to prepare the array for commands to a different row, as shown in Figure 2c. During precharge, the latches in the row buffer are disabled, disconnecting the DRAM cells from the peripheral circuitry, and the voltage of the bitlines is set to half of V_{DD}. Note that a precharge command can be issued by the controller *only after* the DRAM cells in the activated row are fully restored.

DRAM Refresh. A key issue in DRAM is that charge leaks from a DRAM cell, as this charge is stored in a capacitor. When a row has *not* been accessed for a long time, the amount of charge that leaks out of the cell can be large enough to lead to data corruption. To avoid data loss due to excessive charge leakage, the memory controller periodically issues *refresh* commands, which activate the row, restore the charge, and precharge the bank. In DDR3 DRAM, refresh is typically performed on each row every 64 ms [60, 98]. More detail about DRAM refresh can be found in our recent works [29, 74–77, 79, 97, 98, 121, 128].

3 EXPERIMENTAL METHODOLOGY

To develop a thorough understanding of the factors that affect DRAM power consumption, we perform an extensive experimental characterization and analysis of the power consumption of real modern DRAM chips. Each operation described in Section 2.2 consumes a different amount of current. We can directly correlate current to power and energy in DRAM, as (1) DRAM operates at a constant voltage in modern systems; and (2) DRAM operations take a fixed amount of time to complete, which is dictated by a series of *timing parameters* provided by DRAM vendors for each model. Therefore, we provide current measurements in our characterization.

In this section, we describe our methodology for measuring the current consumed by real DRAM modules. In Section 4, we show how real-world current measurements differ significantly from the vendor-specified values that form the basis of existing DRAM power models. In Sections 5 and 6, we study several factors that existing power models neglect to account for, which significantly affect DRAM current consumption. In Section 7, we show current consumption trends over several generations of DRAM. We use our measurements to develop VAMPIRE, a new DRAM power model, in Section 9. We plan to open-source our power model, along with all of our raw measurement data [135].

3.1 Current Measurement Infrastructure

Collecting real power measurements from DRAM has historically been a challenging problem, because in a real system, we do not have the ability to determine or control the sequence of commands that are sent to DRAM, making it difficult to correlate commands with measured power. To work around these obstacles, we construct a custom FPGA-based infrastructure that allows us to (1) precisely control the commands that are issued to the DRAM chips, and (2) accurately measure the current drawn only by the module under test.

Our infrastructure makes use of a significantly-modified version of SoftMC [53, 134], an open-source programmable memory control infrastructure, and allows us to transparently send customized sequences of commands to DRAM chips in order to *reliably* measure current. One of our major modifications adds support to loop continuously over a fixed set of DRAM commands, which the base SoftMC code does not currently support. We do this because even high-end current measuring equipment can read the average current only on the order of every hundreds of microseconds [72], whereas DRAM commands take on the order of tens of nanoseconds. With our command loop support, we repeatedly perform the same microbenchmark of DRAM commands back-to-back, providing us with enough time to accurately measure the current. Our looping functionality ensures that required periodic maintenance operations such as ZQ synchronization [161] are issued correctly to the DRAM chips. As these maintenance operations can alter the state of the DRAM row buffer, we issue them only between loop iterations. We guarantee that maintenance operations do not take more than 0.3% of the total microbenchmark execution time, and thus have a negligible impact on our current measurements. Another of our major modifications adds support for power-down modes, which are an important technique employed in modern DRAM chips to reduce idle power, but are not supported by the base SoftMC code. This requires us to develop new API calls and DRAM commands to start and stop the power-down modes. We plan to incorporate these modifications into the open-source release of SoftMC [134].

Figure 4 shows a photo of the current measurement hardware used for one test setup in our infrastructure, which extends upon the base infrastructure used for SoftMC [53]. We program

SoftMC on a Xilinx ML605 [160], a Virtex-6 [159] FPGA board, which is connected to a host PC and contains an SO-DIMM (small outline dual in-line memory module) [62] socket. To measure the current consumed by each DRAM module that we test, we attach a module to a JET-5467A current-sensing extender board [109]. We remove the shunt resistor provided on the extender, and add in a 5-coil wire. We then insert the coil into a Keysight 34134A high-precision DC current probe [72], which is coupled to a Keysight 34461A high-precision multimeter [73]. The current-sensing extender is then inserted into the SO-DIMM socket on the FPGA board. To validate the accuracy of our infrastructure, we (1) use independent power supplies to confirm the accuracy of the current measurements that are read from each DC current probe, (2) perform electrical connectivity tests to verify against the DDR3 SO-DIMM standard [62] that all power pins on our tested DRAM modules are connected through the extender board's coiled wire, and (3) read back data from the DRAM modules to verify that each FPGA sends the correct DRAM commands to the module that is attached to the FPGA board.

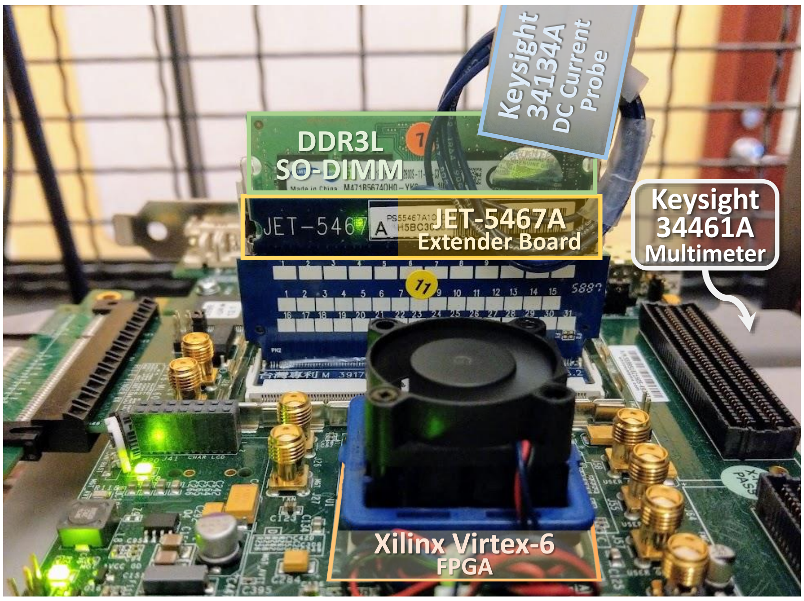


Fig. 4. Our experimental infrastructure connected to an FPGA to measure DRAM current.

We write custom DRAM command microbenchmarks to perform each of our tests (see Sections 4 through 7), controlling three factors: (1) the command sequence issued to DRAM, (2) the data that is read or written, and (3) the latency of each command. We execute one microbenchmark at a time by launching the microbenchmark on the host PC, which sends the DRAM command loop to the SoftMC controller on the FPGA, and we connect the multimeter to the host to sample current measurements while the microbenchmark iterates over the loop. For each test that we execute, we perform ten runs of the test per DRAM module. During each run, we sample the current while the microbenchmark performs the command loop, ensuring that we capture at least ten current samples for each run, and determine the current reading for the overall run by averaging each sample together. We then average the current measured over the ten runs to represent the current consumed by the DRAM module under test. In other words, we collect at least 100 samples per test for each module.

Unless otherwise stated, all DRAM modules are tested at an ambient temperature of 20 ± 1 °C. We examine the effects of high ambient temperature (70 ± 1 °C) using a custom-build heat chamber, where we use a controller connected to a heater to regulate the temperature [31, 32, 79, 92, 97, 121]. We discuss high ambient temperature results in Section 6.2.

3.2 DRAM Modules Tested

We characterize power consumption on 50 DDR3L [61] DRAM modules, which (1) are comprised of 200 DRAM chips, and (2) use the SO-DIMM form factor [62]. Table 1 shows a summary of the modules that we test. These modules are sourced from three major DRAM vendors. Each of our modules contains a single rank, and has a 2 GB capacity. The modules support a channel frequency of up to 1600 MT/s,[1] but all of our tests are conducted at 800 MT/s, due to limitations on the maximum frequency at which our FPGA can operate. In order to anonymize the vendors, we simply refer to them as Vendors A, B, and C in this paper. Many of these modules are the same ones used in our prior work [31, 132], where we characterize the latency and supply voltage behavior, but *not* the measured power consumption, of each module. We supply the modules with their nominal operating voltage of 1.35 V [61].

Table 1. Selected properties of the tested DDR3L DRAM modules.

Vendor	**Total Number of Chips**	**Timing** (ns) ($t_{RCD}/t_{RP}/t_{RAS}$)	**Assembly Year**	**Supply Voltage**	**Max. Channel Frequency** (MT/s)
A (14 SO-DIMMs)	56	13.75/13.75/35	2015-16	1.35 V	1600
B (13 SO-DIMMs)	52	13.75/13.75/35	2014-15	1.35 V	1600
C (23 SO-DIMMs)	112	13.75/13.75/35	2015	1.35 V	1600

We note that while DDR4 modules are available on the market, there is poor experimental infrastructure support available for such modules today; hence our use of DDR3L modules in our characterization. In particular, at the time of writing, no tool equivalent to SoftMC has support for issuing test routines to DDR4 DRAM at a command-level granularity, and it is very difficult and time-consuming to develop a new current measurement infrastructure for DDR4 modules (based on both our prior experience [14–24, 31, 32, 53, 75–77, 79, 84, 89, 92, 97, 101, 102, 121, 128] and on other prior work on building DRAM current measurement infrastructures [66, 106]). However, due to the large number of similarities between the design of DDR3 memory [110] and DDR4 memory [112], we believe that the general trends observed in our characterization should apply to DDR4 DRAM as well. We leave the exact adaptation of the power models that we develop to DDR4 modules and an investigation of the differences between DDR3 power consumption and the power consumption of other DDRx DRAM architectures to future work.

4 MEASURING REAL IDD CURRENT

Most existing DRAM power models are based on *IDD values*, which are a series of current measurement tests [60] that are standardized by the JEDEC Solid State Technology Association (commonly referred to as JEDEC). DRAM vendors conduct these current measurement tests for each DRAM part that they manufacture, and publish the measured values in part-specific datasheets. In order to perform these measurements, a specific series of commands is executed continuously in a loop, and

[1] In double data rate (DDR) DRAM, the channel frequency is typically expressed as *megatransfers per second* (MT/s), where one transfer sends a single 64-bit burst of data across the channel. DDR DRAM sends *two* transfers per clock cycle (one on the positive clock edge, and another on the negative clock edge). This means that for a DRAM with a channel frequency of 1600 MT/s, the channel uses an 800 MHz clock [60].

average current measurements are taken while the loop executes. We start our characterization by measuring the *actual* current consumed by the modules listed in Table 1, and present a summary of the *actual* measurements in this section.

Recall from Section 3 that due to limitations in the maximum frequency attainable on an FPGA, our infrastructure can operate the DRAM modules using a channel frequency of only 800 MT/s. While each vendor provides IDD values for multiple channel frequencies in their datasheets, they do *not* provide IDD values for 800 MT/s, the channel frequency employed by our FPGA infrastructure. However, we can take advantage of the following relationship to extrapolate the expected IDD values at 800 MT/s:

$$P = IV \propto V^2 f \tag{1}$$

where P is power, I is the current, V is the voltage, and f is the frequency. Since the operating voltage is constant at 1.35 V, a linear relationship exists between I and f. As a result, we perform regression using linear least squares [51, 94] to fit the datasheet values to a quadratic model, and use this model to extrapolate the estimated IDD values at 800 MT/s. We find that the datasheet values fit well to the linear model determined through regression. For Vendor C, which has the worst fit out of our three vendors, the lowest R^2 value (which represents the goodness of fit) across all IDD values is 0.9783. Therefore, we conclude that our estimated IDD values at 800 MT/s are accurate.

There are five types of IDD current values that we measure: (1) idle: IDD2N, IDD3N; (2) activate and precharge: IDD0, IDD1; (3) read and write: IDD4R, IDD4W, IDD7; (4) refresh: IDD5B; and (5) power-down mode: IDD2P1.

4.1 Idle (IDD2N/IDD3N)

We start by measuring the idle (i.e., standby) current. JEDEC defines two idle current measurement loops: (1) IDD2N, which measures the current consumed by the module when *no* banks have a row activated; and (2) IDD3N, which measures the current consumed by the module when *all* banks have a row activated.

Figure 5 shows the average current measured during the IDD2N loop. We use *box plots* to show the distribution across all modules from each vendor. Each box illustrates the quartiles of the distribution, and the whiskers illustrate the minimum and maximum values. We make two key observations from this data. First, *there is non-trivial variation in the amount of current consumed from module to module for the same DRAM vendor.* The amount of variation is different for each vendor, and the range normalized to the datasheet current varies from 14.7% for Vendor A to 37.5% for Vendor B. As the architecture of the module remains the same for modules from the same vendor (because we study a single part per vendor), we conclude that these differences are a result of manufacturing process variation. Second, *the measured currents are significantly lower than the datasheet values.* As we can see in the figure, DRAM vendors leave a *guardband* (i.e., margin) in the reported IDD values. The capacitors used for DRAM cells are very tall and narrow trenches [114], which improves the chip density. Unfortunately, the very high *aspect ratio* (i.e., height over width) of the cells (e.g., > 70 for modern DRAM chips [57]) increases the difficulty of DRAM lithography, and can result in significant process variation. Vendors use the guardband to account for the expected worst-case process variation in IDD values. Our average IDD2N measurement is 38.3%, 76.6%, and 54.9% of the specified IDD2N current for Vendors A, B, and C, respectively.

We see the same trends for the current measured during the IDD3N loop, as shown in Figure 6. The average measured IDD3N current is 23.4%, 53.2%, and 33.4% of the specified IDD3N current for Vendors A, B, and C, respectively. We observe that the full *normalized* range of the measured current (i.e., the difference in current between the highest-current DRAM module and the lowest-current

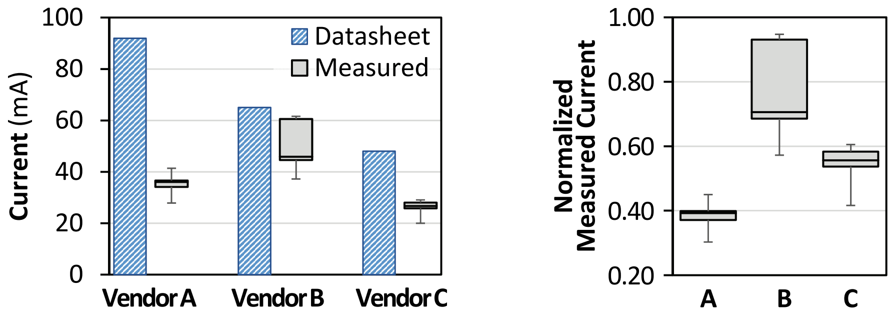


Fig. 5. IDD2N current measurements (left), and current normalized to datasheet value (right).

module) is 8.8%, 19.3%, and 12.4% of the specified current, respectively for the three vendors. The normalized range represents how much variation in current exists across the modules that we tested for each vendor.

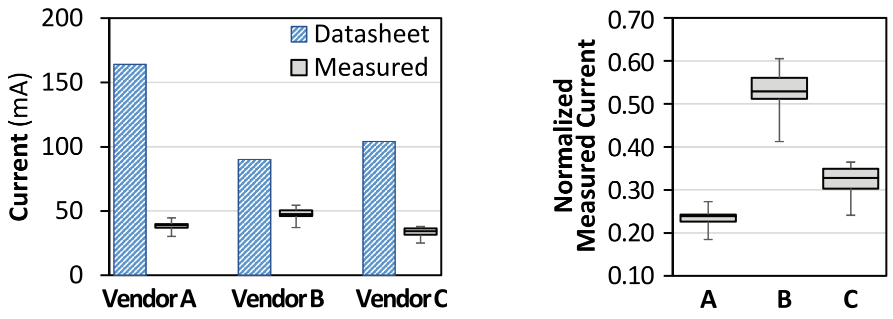


Fig. 6. IDD3N current measurements (left), and current normalized to datasheet value (right).

We conclude that (1) the actual power consumed by real DRAM modules in the idle state is much lower than the IDD2N/IDD3N values provided by the vendors, and (2) there is significant variation of these current values across parts manufactured by a given vendor.

4.2 Activate and Precharge (IDD0/IDD1)

Next, we study the amount of current consumed during activate and precharge operations. Unfortunately, it is not possible to measure activation and precharge current *independently* in real DRAM modules, as a second activation *cannot* take place before an already-activated row is precharged. JEDEC defines two measurement loops for activation and precharge: (1) IDD0, which performs successive activate and precharge operations as quickly as possible without violating DRAM timing parameters; and (2) IDD1, which performs successive {*activate*, *read*, *precharge*} operations in a similar manner.

Figure 7 shows the average current measured during the IDD0 loop. We make two key observations. First, we again observe a large margin between the datasheet values and our measurements, and find that the activation and precharge current consumption is much lower than expected. Our average IDD0 measurement is 40.2%, 42.6%, and 45.4% of the specified IDD0 current for Vendors A, B, and C, respectively. Second, we find that the absolute amount of current consumed across all three models is somewhat similar despite the large difference in the datasheet specification, with average current measurements of 72.2 mA, 70.4 mA, and 58.1 mA for the three respective vendors.

We note very similar trends for IDD1, as shown in Figure 8, with average current measurements of 107.4 mA, 114.9 mA, and 87.9 mA for the three respective vendors

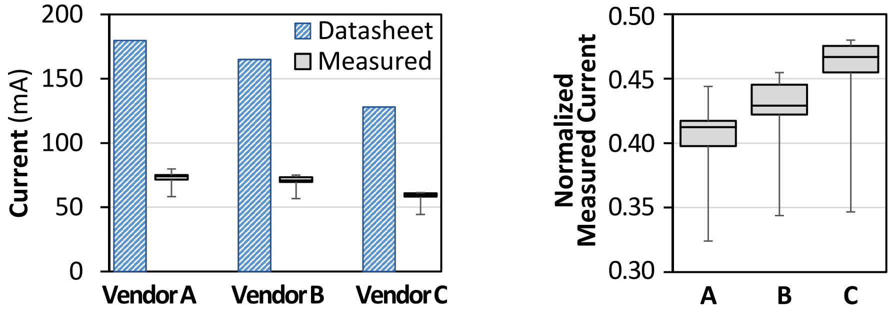


Fig. 7. IDD0 current measurements (left), and current normalized to datasheet value (right).

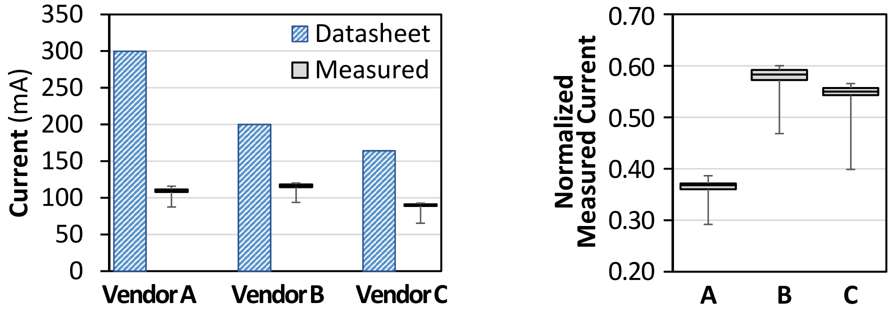


Fig. 8. IDD1 current measurements (left), and current normalized to datasheet value (right).

We conclude that the actual power consumed by real DRAM modules during the activate and precharge operations is much lower than the IDD0/IDD1 values provided by the vendors.

4.3 Read and Write (IDD4R/IDD4W/IDD7)

We study the amount of current consumed during read and write operations. JEDEC defines three measurement loops for these operations: (1) IDD4R, which performs back-to-back read operations to open rows across *all* eight banks; (2) IDD4W, which performs back-to-back write operations to open rows across *all* eight banks; and (3) IDD7, which interleaves {*activate*, *read*, *auto-precharge*} operations across *all* eight banks.

Figure 9 shows the average current measured during the IDD4R loop. As we observe from the figure, several of our current measurements actually *exceed* the value specified by vendors in the datasheets. In fact, for the modules from Vendor C, the average current measured from the modules exceeds the datasheet value by 11.4%, with a current of 343.5 mA. These measurements should not be interpreted as a lack of a margin for IDD4R or an underestimation by the DRAM vendor. Instead, these measurements represent a limitation of our FPGA measurement infrastructure. As part of the read operation, the DRAM module must drive the data values across the memory channel. To do so, a read operation selects a column from an open row of each DRAM chip in the target rank, and uses the *peripheral circuitry* inside a DRAM chip, which is responsible for performing the external I/O (see Section 2.2). While vendor specifications *ignore* the portion of the current used by the I/O driver in the IDD4R value, our measurement infrastructure *captures* the I/O driver current, which can account for a sizable portion of the total measured current. As a result, our measured current includes a portion that is not included by the DRAM vendors, causing some of our measured values to be larger than those reported by vendor datasheets.

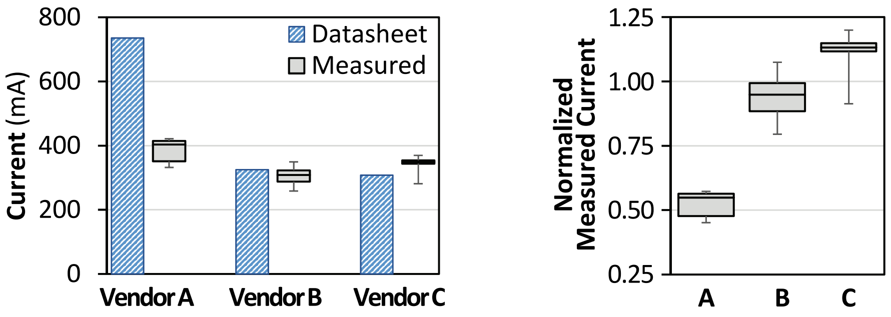


Fig. 9. IDD4R current measurements (left), and current normalized to datasheet value (right).

We estimate the amount of the current consumed by the I/O driver (see Section 5.1), and subtract this amount from our original IDD4R measurement, as shown in Figure 10 (the *Corrected* bars). After this correction, the average IDD4R value drops from 52.6%, 94.7%, and 111.4% of the specified IDD4R current to 45.9%, 79.5%, and 95.4% for Vendors A, B, and C, respectively. We observe that even with the corrections, the margins provided by Vendors B and C for IDD4R are much smaller than the margins for the other IDD values that we measure. This may be because the read operation does *not* interact directly with DRAM cells, which are susceptible to significant manufacturing process variation, and predominantly makes use of the sense amplifiers, peripheral logic, and I/O drivers.

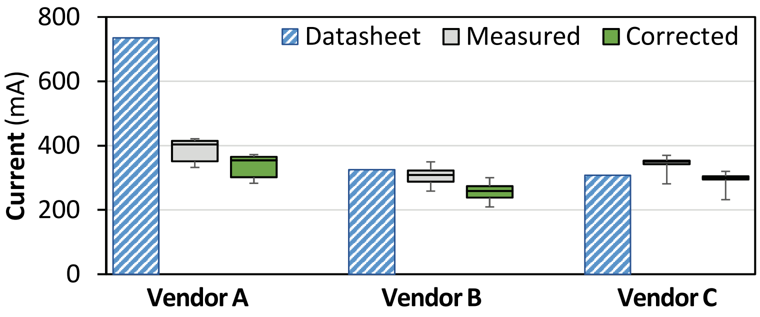


Fig. 10. IDD4R current measurements before and after correction for I/O driver current.

Figure 11 shows the average current measured during the IDD4W loop. We observe that unlike the IDD4R results, our measurements for IDD4W are much smaller than the datasheet values. This is due to two reasons. First, during a write operation, the peripheral circuitry within the DRAM chip does *not* need to drive current across the memory channel, instead acting as a current sink. Second, unlike read operations, a write operation affects the charge stored within DRAM cells. Because the DRAM cells are susceptible to significant manufacturing process variation, the IDD4W numbers that are reported by the vendors include a large guardband to account for worst-case DRAM cells that can consume much higher current than the typical cell. On average, the measured

IDD4W current is 49.1%, 54.5%, and 59.0% of the specified IDD4W current for Vendors A, B, and C, respectively.

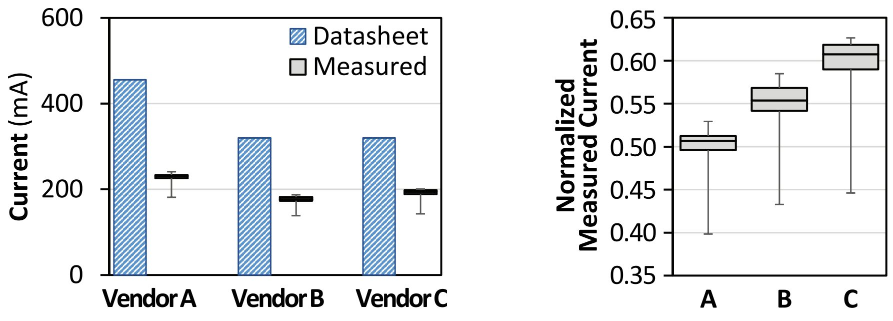


Fig. 11. IDD4W current measurements (left), and current normalized to datasheet value (right).

Our IDD7 measurements behave very similarly to IDD0 and IDD1, but have a larger range, as shown in Figure 12. As the read operations are interleaved with activate and precharge operations to each bank, the IDD7 measurement loop accesses the DRAM cells. As a result, unlike what we observed for IDD4R, the measured IDD7 values have large margins compared to the datasheet. The average measured IDD7 current is 58.4%, 43.5%, and 52.7% of the specified IDD7 current for Vendors A, B, and C, respectively, and the full normalized range (i.e., the difference between the highest-current module and the lowest-current module) is 10.1%, 17.9%, and 18.1%, respectively for the three vendors.

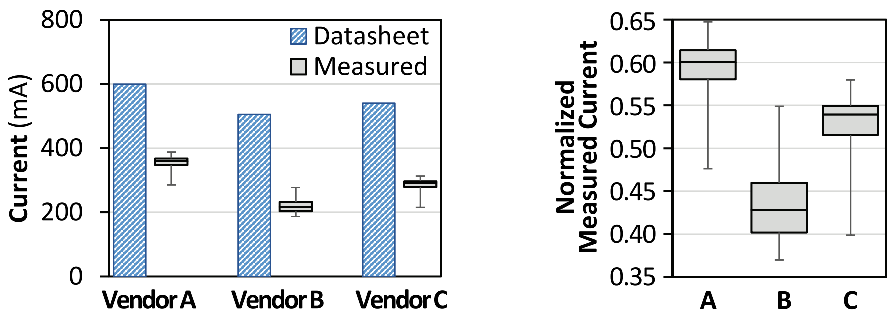


Fig. 12. IDD7 current measurements (left), and current normalized to datasheet value (right).

We conclude that (1) the measured read current is *not* much lower than the datasheet value, even after we subtract the effect of the I/O driver current; and (2) operations that access the cell array in addition to the peripheral circuitry are likely to consume less current than the specified datasheet values.

4.4 Refresh (IDD5B)

Next, we study the amount of current consumed during refresh operations. We study the IDD5B current measurement loop defined by JEDEC, which performs a continuous burst of refresh commands. Figure 13 shows the current measured during the IDD5B loop. We note that the refresh current consumes the *highest* current of any of the operations that we have observed, and that the margin for refresh is *not* as large as the idle current margin (see Section 4.1). For Vendors A, B, and

C, we observe average current consumption across all modules to be 88.6%, 72.0%, and 88.0% of the specified IDD5B current. However, while the margin is small, the measured refresh current never exceeds the specified value.

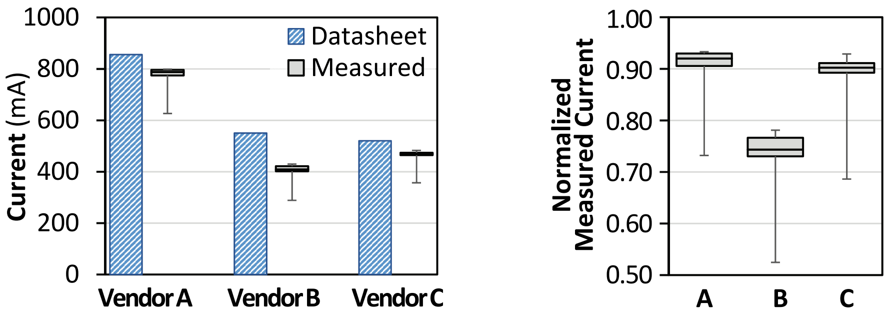


Fig. 13. IDD5B current measurements (left), and current normalized to datasheet value (right).

We conclude that the measured refresh current is *not* significantly lower than the corresponding IDD5B value in the datasheet.

4.5 Power-Down Mode (IDD2P1)

Last, we study the impact of low-power modes in DRAM. Modern DDR DRAM architectures provide several modes to minimize power consumption during periods of low memory activity. We focus on the *fast power-down* mode available in DDR3L DRAM [60, 61], which turns off the internal clock, decode logic, and I/O buffers, but keeps the delay-locked loop (DLL) circuit active. We study the IDD2P1 measurement loop defined by JEDEC for the fast power-down mode, which measures current when *no* bank is active.[2]

Figure 14 shows the current measured during the IDD2P1 loop. We observe that the power-down mode is quite effective when no bank is active, reducing the current significantly compared to when no bank is active in normal power mode (which we characterize above using the IDD2N measurement loop, as shown in Figure 5). For Vendors A, B, and C, power-down mode reduces the current by 65.8%, 30.6%, and 48.7%, respectively (as observed by comparing the measured values in Figure 14 to those in Figure 5.). For Vendors A and C, for whom power-down mode is highly effective, the variation across modules in power-down current is small as well, with a normalized range of 4.8% and 17.3% of the specified IDD2P1 current, respectively. In contrast, the power-down mode for Vendor B consumes significantly greater power, and its current ranges by as much as 47.9% of the specified IDD2P1 current, indicating a less efficient and more variation-prone power-down implementation than the implementations of Vendors A and C.

We conclude that the power-down mode is effective at reducing power, but it has significant power variation across vendors.

4.6 General Observations

Collectively examining all of the IDD results that we present in Sections 4.1–4.5, we make two key observations.

[2]A second mode, known as *slow power-down*, turns off the DLL circuit *in addition to* the internal clock, decode logic, and I/O buffers. We are unable to test the slow power-down mode, because our test infrastructure does *not* allow us to disable the DLL circuit. As a result, we do not include results for the IDD2P0 measurement loop, which is designed to test the slow power-down mode.

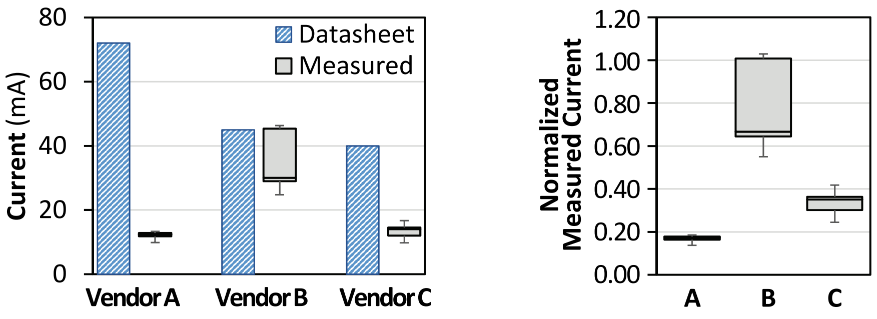


Fig. 14. IDD2P1 current measurements (left), and current normalized to datasheet value (right).

First, we find that the majority of IDD values specified by the DRAM vendors are *drastically* different from our measured results. We believe that our measurements provide a more realistic vision of the amount of power consumed by real-world modules than the vendor-specified IDD values, and demonstrate inter-vendor and intra-vendor module-to-module variation in DRAM power consumption. In fact, we find that power models based on the IDD values are oblivious to many significant factors that can affect the power consumed by DRAM. These power models *assume* accesses to specific banks, rows, and columns using specific data patterns, but as we show in Sections 5 and 6, varying these factors can have a *non-trivial* impact on the energy consumed by a DRAM module.

Second, we find that while there is a large difference in the datasheet-reported IDD values across our three vendors, the difference in *measured* current is much smaller than the IDD values suggest for activate, read, write, and precharge operations. For example, for the IDD4R value, the datasheets state that Vendor A's DRAM modules consume 139% more current than Vendor C's modules (a total difference of 427 mA). In reality, we find from our measurements that Vendor A's modules consume only 26% more current than Vendor C's modules on average (a total difference of 79 mA). We believe that this observation is a result of all three vendors' modules being manufactured using similar process technology nodes. This is quite likely given the fact that the modules were all manufactured around the same time (see Table 1). Despite the use of similar process technology nodes by all three vendors, Vendor A's IDD values appear to include much larger margins than the IDD values from Vendors B and C. This could reflect either different levels of conservatism among the vendors, or could imply that modules from vendors that employ larger margins have greater variation (though we do *not* observe this in our experiments).

5 DATA DEPENDENCY

One major aspect that existing DRAM power models ignore is the effect that a *data value* has on DRAM power consumption. The IDD measurement loops, as defined by JEDEC, use only the data patterns 0x00 and 0x33 (where the byte value is repeated for each byte within the cache line). While this allows for the tests to be standardized, it does *not* offer any insight into whether or not current consumption depends on the data value that is being read or written.

In this section, we design a set of measurements that determine whether or not a relation exists between the data value and power consumption, and analyze the results of these measurements. We break these studies down into two parts: (1) whether the *number of ones* within a cache line impacts the power consumed by DRAM (Section 5.1), and (2) for a *fixed number of ones*, whether the fraction of *bits toggled* within a cache line impacts the power consumed by DRAM (Section 5.2). Based on our studies, we develop models for our modules that quantify how power consumption changes for each operation when we account for (1) the number of ones in a cache line and (2) the number of wires that experience toggling during a read or write operation (Section 5.3).

5.1 Effect of Number of Ones in Data

We start by exploring the relationship between the number of ones in a 64-byte cache line (which consists of a number of columns read from DRAM, as we discuss in Section 2.2) and the power consumed. In order to test this behavior, we select a set of rows that we would like to test, and populate each column of the row with the same data pattern. We then repeatedly read data out of a *single column in a single row*. Figure 15 shows how the current drawn by the DRAM module (y-axis) changes as we increase the number of ones in the cache line (x-axis), for both reads (Figure 15a) and writes (Figure 15b). We make two key observations from the figure.

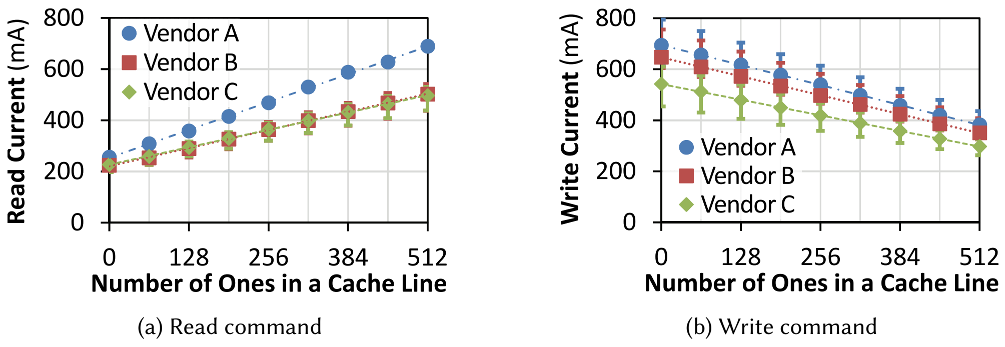


Fig. 15. Effect of the number of ones on the read (left) and write (right) current drawn by DRAM. Error bars indicate the 25th and 75th percentile measurements.

First, *as the number of ones in a cache line increases, the current required for a read operation increases, while the current required for a write operation decreases.* The variation in power consumption with the number of ones is as much as 434 mA for reads and 311 mA for writes (Vendor A). There are two causes for this: (1) the I/O driver design, and (2) data-dependent power consumption within the DRAM module. As we discuss in Section 2.2, when data is transferred over the memory channel, each wire of the channel is attached to two I/O drivers: one inside the DRAM module, and another inside the memory controller. Only one of the I/O drivers actively *drives* current on the wire at a time, depending on (1) the operation being performed, and (2) the bit value that is transferred on the wire. When one of the I/O drivers is driving current on the wire, the other I/O driver *sinks* current [71]. For example, when the wire is transferring a bit value *one* during a *read* operation from DRAM, the I/O driver in the DRAM module drives the current on the wire, while the I/O driver in the memory controller sinks current. When the wire is transferring a bit value *zero* during a *read* operation from DRAM, the I/O driver in the DRAM module sinks current, while the I/O driver in the memory controller drives current. The opposite is true for *write* operations to DRAM: the I/O driver in the memory controller drives current when the wire is transferring a bit value *one* to DRAM, and the I/O driver in the DRAM module drives current when the wire is transferring a bit value *zero* to DRAM. Even if we eliminate the estimated current used by the I/O drivers (see Section 5.3), as shown in Figure 16, we still observe significantly data-dependent power consumption, which can change the current by as much as 230 mA for reads and 111 mA for writes (Vendor A). While we cannot definitively identify the sources of data-dependent power consumption within the DRAM, we suspect that other peripheral circuitry within the DRAM, such as the bank select and column select logic (see Section 5.2), may be responsible for the data-dependent current behavior.

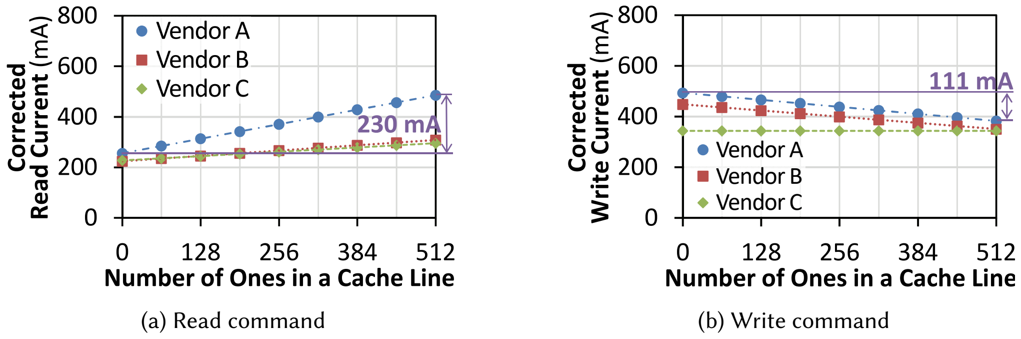


Fig. 16. Effect of the number of ones on the read (left) and write (right) current drawn by DRAM, after subtracting a conservative estimate of the I/O driver current.

Second, we observe that *the relationship between the current consumption and the number of ones is linear*. Note that this linear relationship is true *even after* we remove the effect of the I/O driver current on the current measurements shown in Figure 15, as shown in Figure 16. We use this linear relationship to build models of the current consumption in Section 5.3.

We also perform tests to determine whether the data value stored within a row affects the activate and precharge current (not shown). We find that there is *no* notable variation for activate and precharge current consumption based on the stored data value. This is due to the way in which bitlines access a row during activation. For each bitline, there is a corresponding reference bitline. For a DRAM where a bit value 0 is represented as 0 V and a bit value 1 is represented as V_{DD}, the bitline and reference bitline are precharged to voltage $V_{DD}/2$, allowing them to swing in either direction based on the data held in the row. During activation, a cell swings (i.e., *perturbs*) the voltage of its bitline in the direction of the charge stored in the cell, and the reference bitline is perturbed in the *opposite* direction. Thus, for every activation, there is always one line swinging up to V_{DD}, and another line swinging down to 0V [71]. As a result, there is little difference between the power required for a cell that stores a zero and a cell that stores a one during activate and precharge operations.

5.2 Effect of Bit Toggling

Next, we explore how interleaving memory requests across multiple columns and banks affects the current drawn by DRAM. Figure 17 shows a high-level overview of the logic used in a DRAM chip to select the bank and column desired by a request. Each bank contains *column select* logic (e.g., ❶ for Bank 0 in Figure 17). When a row is active in a bank, the entire contents of the row are latched in a row buffer (see Section 2.1). The column select logic chooses one column of data from the row buffer to output on the global bitline of the bank. The DRAM chip then uses the *bank select* logic (❷) to choose which global bitline contains the column that the chip should send across the memory channel. The bank select logic sends this data across the peripheral bus to the I/O drivers.

Bit toggling can occur when two back-to-back requests go to different banks and/or columns. We can see where bit toggling occurs by using the example 4-bit select logic shown in Figure 17. If we have two back-to-back read requests, where Request W reads Bank 0, Column 0, and Request X reads

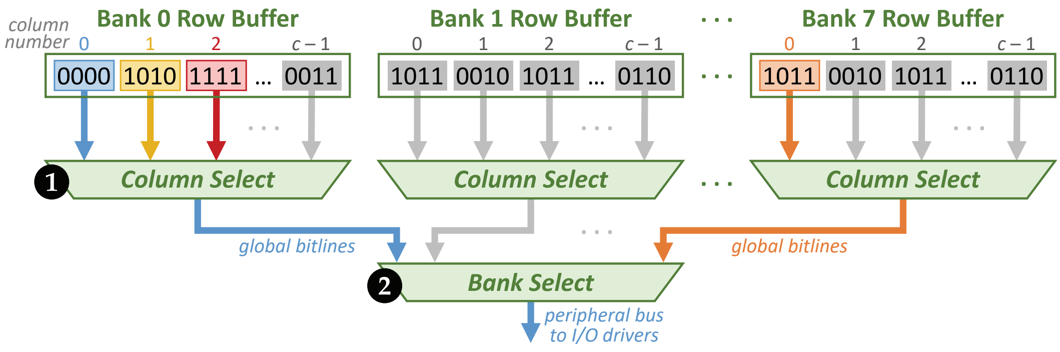


Fig. 17. Column and bank select logic organization.

Bank 0, Column 1, the global bitline wires for Bank 0 first send binary value 0000 (for Request W), and then send binary value 1010 (for Request X). This causes two of the wires in the global bitline, and two of the wires in the peripheral bus, to toggle from a bit value 0 to a bit value 1. The number of wires that toggle is data dependent: if Request X reads Bank 0, Column 2 instead, all four wires of the global bitline and all four wires of the peripheral bus toggle. Requests W and X are an example of *column interleaving*, because the requests go to two separate columns in the same bank. Bit toggling can also take place during *bank interleaving*, where back-to-back requests go to different banks. For example, if we have two back-to-back read requests, where Request Y reads Bank 0, Column 0, and Request Z reads Bank 7, Column 0, three of the peripheral bus wires experience bit toggling, as the wires first send binary value 0000 (for Request Y) and then send binary value 1011 (for Request Z). Note that toggling does not always occur when requests are column- and/or bank-interleaved. For example, if one request reads Bank 7, Column 0, and another request reads Bank 7, Column 2, the data on both the global bitlines for Bank 7 and the peripheral bus does not change.

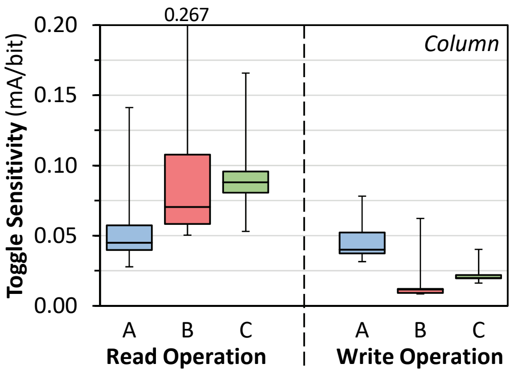
We design a series of tests that can capture the current consumed by the column and bank select logic, and how bit toggling at the column/bank select logic affects current consumption. We perform three types of tests:

(1) *No Interleaving*: All requests access the *same* bank and the *same* column.
(2) *Column Interleaving*: Back-to-back requests access *different* columns in the *same* bank.
(3) *Bank+Column Interleaving*: Back-to-back requests access *different* banks, and the column that is being accessed in a particular bank is *different* from the column that was accessed by the last access to that bank.

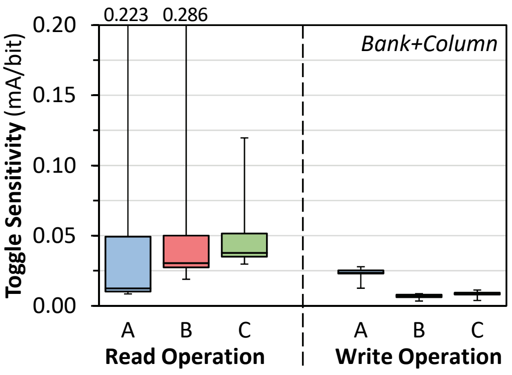
For each of the three types above, we perform multiple tests, varying (1) the data pattern stored in or written to each column, and (2) whether the test consists of all read requests or all write requests.

Note that when we change the data pattern being used, the change in current is affected by two factors: (1) the increase in current due to bit toggling, and (2) the increase in current due to the number of ones in the data (see Section 5.1). As an example, consider what happens during the *column interleaving* test for two different pairs of data values. If we constantly alternate between reading data value 0x00 and data value 0xAA from two columns, 50% of the bitlines experience toggling. If we instead alternate between data value 0x00 and data value 0x0A, the toggle rate is only 25%. However, the column with data value 0xAA also has two more bits set to '1' than the the column with data value 0x0A, which requires more current, as we discuss in Section 5.1. In order to isolate the effect of *only* bit toggling, we calculate the total number of ones in the two reads, and subtract the toggle-free current consumed when reading the same number of ones with column interleaving. For our example test where we alternate between a column with data value 0x00 and another column with data value 0xAA (i.e., across both columns, there are an average of two '1's per column), we eliminate the impact of the number of ones by subtracting the current consumed when we alternate between two columns that both contain data value 0x88 (where each column has two '1's).

Figure 18 shows how the measured current increases as we increase the number of bits that are toggling, summarizing the increase per bit across a wide range of data values for our *column interleaving* (Figure 18a) and *bank+column interleaving* (Figure 18b) tests. As we found for the number of ones, the current increases linearly as we increase the number of bits that are toggling. In Figure 18, we plot *toggle sensitivity* on the y-axis, which shows the *increase* in current for each additional bit that is toggling, in terms of mA/bit.



(a) *Column interleaving*



(b) *Bank+column interleaving*

Fig. 18. Effect of bit toggling on read and write current consumption.

We make two key observations from the figure. First, *the impact of bit toggling on DRAM current consumption* (*up to* a total of 26 mA for Vendor A with column interleaving when *all* bits are toggling) *is much smaller than the impact of the number of ones* (230 mA for Vendor A when *all* bits are set to ones; see Section 5.1). Second, *toggling for the* bank+column interleaving *test requires less current than toggling for the* column interleaving *test.* We believe that both of these observations are due to the design of the select logic, which we show in Figure 17. When the DRAM selects another column in the same bank, both the wires between the column select logic and the bank select logic, and the wires between the bank select logic and the I/O drivers, experience toggling. In contrast, when DRAM selects a different bank, only the wires of the peripheral bus experience toggling, which reduces the bit toggling energy in the *bank+column interleaving* test compared to the bit toggling energy in the *column interleaving* test.

We conclude that there is a linear relationship between the current consumed and the number of bits that toggle, but that the amount of current consumed as a result of bit toggling is small, especially when compared to the current consumption effect of the number of ones in the data.

5.3 Data Dependency Models

From our experiments in Sections 5.1 and 5.2, we observe a linear relationship between the current consumed and the number of ones in the cache line, as well as a linear relationship between the current consumed and the number of bits that toggle due to back-to-back read/write requests. As a result, we use linear least-squares regression on our characterization data to develop quantitative models for this relationship, in the following form:

$$I_{total} = I_{zero} + \Delta I_{one} N_{ones} + \Delta I_{toggle} N_{toggles} \quad (2)$$

where I_{total} is the total current consumed (in mA), I_{zero} is the current consumed when the cache line contains all zeroes, ΔI_{one} represents the extra current for each additional one in the cache line, N_{ones} is the number of ones in the cache line, ΔI_{toggle} represents the extra current for each additional bit that is toggling, and $N_{toggles}$ is the number of bits that were toggled, We confirm the linear relationships of current with (1) the number of ones in the cache line and (2) the number of bits toggled, by using the square of the Pearson correlation coefficient [123], commonly known as the R^2 value. We find that across all of the modules that we measure, the R^2 value of these two correlations is never lower than 0.990.

Because different types of interleaving make use of different switching circuitry, we require a separate set of model parameters to use in Equation 2 for each type of operation interleaving. Table 2 shows the average values of I_{zero}, ΔI_{one}, and ΔI_{toggle} in mA for read and write operations, with *column interleaving*, for each module vendor. We compare the output of our model shown in Table 2 to the average measured current of each data point shown in Figure 15 (see Section 5.1), and find that the percent error of our model never exceeds 1.40% (and the average percent error across all data points is 0.34%). Table 5 in the appendix shows the parameters used to model other combinations of bank and/or column interleaving. These data dependency models form a core component of our new measurement-based DRAM power model (see Section 9).

Table 2. Parameters (I_{zero}, ΔI_{one}, ΔI_{toggle}; in mA) used in Equation 2 to model current consumption (I_{total}) due to data dependency when read/write operations are column interleaved.

Vendor	**Read**			**Write**		
	I_{zero} (mA)	ΔI_{one} (mA)	ΔI_{toggle}(mA)	I_{zero} (mA)	ΔI_{one} (mA)	ΔI_{toggle} (mA)
A	246.44	0.433	0.0515	531.18	-0.246	0.0461
B	217.42	0.157	0.0947	466.84	-0.215	0.0166
C	234.42	0.154	0.0856	368.29	-0.116	0.0229

We conclude that our models accurately capture the relationship between data dependency and DRAM current consumption.

6 CHARACTERIZING VARIATION OF CURRENT

A significant limitation of existing DRAM power models [25, 27, 65, 111] is that they are based on IDD tests that are performed by DRAM vendors on only a fixed set of banks and rows at room temperature [60]. While these fixed conditions ensure repeatability, the resulting existing models do *not* capture power consumption variation across banks, rows, or temperature. In this section, we perform a series of experiments to characterize (1) *structural variation*, where current may vary based on the bank, row, or column selected due to the circuit-level design of the DRAM chip (Section 6.1), and (2) whether the *operating temperature* of DRAM affects its current consumption (Section 6.2). The results we have presented throughout the paper so far already capture a third type of variation: *process variation*. As we have shown using box plots (Figures 5–14), different modules of the same part, from the same vendor, exhibit a non-trivial amount of current variation for tests that target the same bank(s), row(s), and column(s).

6.1 Structural Variation of Current

Each module consists of a number of hierarchical structures (i.e., banks, rows, and columns) that are connected together to provide density and parallelism. Due to the need to maximize density and optimize the physical chip layout, there may be low-level differences among some components. For example, in the Open Bitline architecture [59], a DRAM array is broken up into *subarrays* [29, 30, 82, 139], and pairs of subarrays share common row buffer structures. Subarrays in the middle of the bank share structures with *both* neighbors, but subarrays placed at the edge share structures with only *one* neighbor. Likewise, due to variation in the distance between different rows in a subarray and the logic required to access the row (e.g., wordline select logic, row buffers), there can be significant variation in the latency of different rows [32, 89]. We now study if such structural variation factors impact the current consumed by DRAM. We consider variation to be structural in nature only when we observe the same trend repeated in each of the modules we study from the same vendor. We consider all other variation to be due to manufacturing process variation, and do not report it in this section.

6.1.1 Structural Variation Across Banks. We first characterize current variation *across banks within the same module.* Figure 19 shows the **idle current** consumed when we keep Row 0, which contains all zeros, open (i.e., activated) in each bank. In the figure, the average measured current of each bank is normalized to the average measured current consumed by Bank 0, where Row 0 is activated and contains all zeroes, for each vendor. We observe that modules from Vendors A and B show little to no inter-bank variation in their idle current consumption, but modules from Vendor C have significant variation. Depending on which bank is activated, the current can vary by as much as 23.6%. We find that these results hold for other rows that we test. As we discuss in detail below, we hypothesize that Vendor C's DRAM chip organization contains structural differences across each bank, resulting in the current variation that we observe.

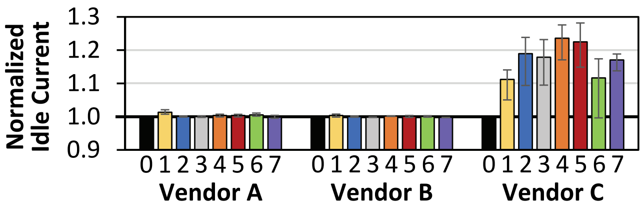


Fig. 19. Idle current variation across banks when one bank is active, normalized to current for Bank 0. Error bars indicate the 25th and 75th percentile measurements.

Next, we characterize the variation in **read current** for each bank within the same module. Figure 20 shows the average current consumed when we repeatedly read Column 0 from Row 0 for each bank, normalized to the average measured current consumed for Bank 0 for each vendor. For these experiments, Row 0 contains all zeroes. We observe that, in this case, *all* of the modules exhibit variation. The variation for Vendor C does not match the variation trend observed in Figure 19. We perform the same experiments for writes (Figure 21), but find no notable variation for any of our modules. This indicates that the structural variation is a result of components that are used during read operations but not during write operations. We observe similar variation trends when we repeat the experiments with different data values in Row 0.

Based on our results, we hypothesize that for modules from Vendors A and B, the variation is a result of structural variation in the I/O driver circuits used to read data, as the I/O drivers in the DRAM module drive current on the DRAM channel only during a read operation. As Vendor C's

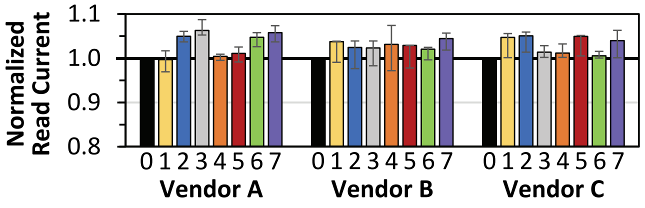


Fig. 20. Read current variation across banks, normalized to current for Bank 0. Error bars indicate the 25th and 75th percentile measurements.

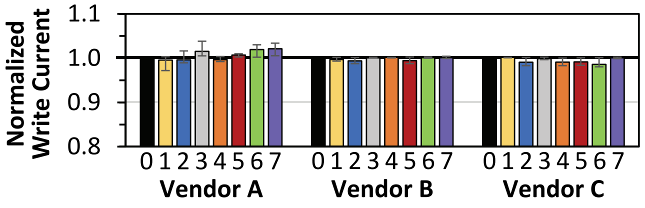


Fig. 21. Write current variation across banks, normalized to current for Bank 0. Error bars indicate the 25th and 75th percentile measurements.

modules show variation in the idle state and during read operations, but the variation trends do not match, we conclude that there are multiple sources for the variation that we observe, which include I/O driver variation. Overall, we find that structural variation exists across banks, but that the pattern of variation is highly dependent on the vendor, due to differences in the DRAM architecture from vendor to vendor. Unfortunately, without access to detailed information about the underlying DRAM architecture of each part (which is information proprietary to DRAM vendors [75, 89]), we are currently unable to pinpoint the exact sources of this structural variation.

6.1.2 Structural Variation Across Rows. Next, we characterize current variation *across rows within the same bank*. For each module, we measure and compare the current consumed when we repeatedly activate and precharge 512 different rows.[3] We find that there is systematic structural variation in each of our modules. We observe that the *current consumed by each row increases with the number of ones in the row address*. Figure 22 shows this trend, where we average together the current consumed by rows that contain the same number of ones in their row address, and plot the average current sorted by the number of ones in the address (on the x-axis). As the figure shows, modules from Vendors A and B show a correlation between the number of ones in the row address and the current. For modules from Vendor B, a row with 15 ones in its address consumes 14.6% more current than a row with all zeroes in its address. Modules from Vendor C show a similar trend, but exhibit a much smaller slope, and thus less variation, than modules from Vendors A and B.

We hypothesize that there are two potential sources of this variation. First, due to the way that rows are organized within the DRAM cell array, rows with more ones in their row addresses are *more* likely to be physically *further* away from the sense amplifier and column select logic [89]. As a result, longer segments of wires must be driven for operations to rows with more ones in their row addresses. Second, the row decoding logic uses some or all of the row address bits to enable

[3]We do not study the power consumed by read and write operations across different rows, as these operations are *not* performed on the cells themselves. Instead, reads and writes operate on data that is already in the row buffer, and the same row buffer is used by all of the rows in a bank. Thus, the read and write operations use the same hardware structures regardless of the row being accessed.

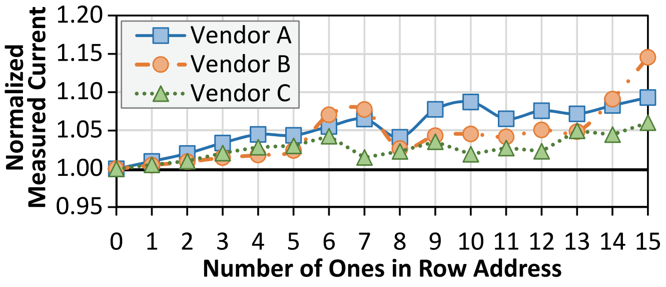


Fig. 22. Relationship between activation current and number of ones in row address, normalized to activation current for Row 0.

the wordline of the row that is activated. The row decoding logic may consume more energy when a greater number of row address bits are set to one. We cannot, however, confirm these hypotheses without knowing the circuit-level implementation of the internal logic of each DRAM chip, which is information proprietary to DRAM vendors.

6.1.3 Structural Variation Across Columns. Last, we characterize the current variation *across columns within the same row*. To this end, for each column in an activated row, we measure the current consumed when we repeatedly read from or write to that column, and compare this with the current consumed when we repeatedly read from or write to Column 0. (We do not show these comparisons for brevity.) We find that for both read and write operations, there is no notable variation in current consumption from one column to another. We hypothesize that this lack of variation is because read and write operations to different columns in a row make use of the *same* global bitlines, bank select logic, peripheral bus, and I/O drivers (see Figure 3 in Section 2.2). Thus, we conclude that there is *no* significant source of structural current variation between columns.

6.2 Variation of Current Due to Temperature

Prior work has shown that DRAM latency and refresh rates can be affected by the temperature at which DRAM operates [31, 32, 79, 92, 97, 121]. To investigate if a relationship exists between operating temperature and the current consumed by DRAM, we repeat all of our experiments at 70 ± 1 °C.

From our experiments, we do *not* observe any measurable current variation due to temperature (results not shown for brevity). We believe that this is a limitation of our DRAM testing infrastructure. In DRAM, the main source of temperature-related effects is the change in charge leakage. At higher temperatures, the charge stored within a DRAM cell leaks more rapidly [29, 79, 97, 98, 113, 121]. As a result, refresh operations must either (1) restore a greater amount of charge into the cell or (2) be performed more frequently, to make up for the additional charge that leaked over the same amount of time. As our test infrastructure continually iterates over a loop of DRAM commands, the DRAM cells are continually accessed, and do *not* have enough time to leak charge [97]. Thus, our measurements *cannot* capture the impact of charge leakage without extensive modifications to the SoftMC design. We leave such modifications to SoftMC, and the resulting characterization of how charge leakage due to temperature affects DRAM power consumption, to future work.

7 GENERATIONAL TRENDS

The results we have presented so far examine the power consumed by modules manufactured in recent years, using the latest process technologies developed for DRAM. As is the case with microprocessors, end users and system designers have grown accustomed to reduced power consumption when new process technologies are used. For DRAM, users and designers currently rely on datasheet current specifications to estimate the amount of power savings from one generation to another. In this section, we compare the power reduction trends indicated by the datasheet values with the *actual* power savings, as measured using our infrastructure.

We study changes in power consumption across DRAM generations for Vendor C. In addition to the modules listed in Table 1, we have access to a number of older modules manufactured by Vendor C. Table 3 summarizes select properties of these modules. We test modules of two older parts, with one of the parts manufactured in 2011, and the second part manufactured in 2012. In comparison, the Vendor C modules studied thus far in this paper were manufactured in 2015.

Table 3. Properties of older DDR3L modules from Vendor C.

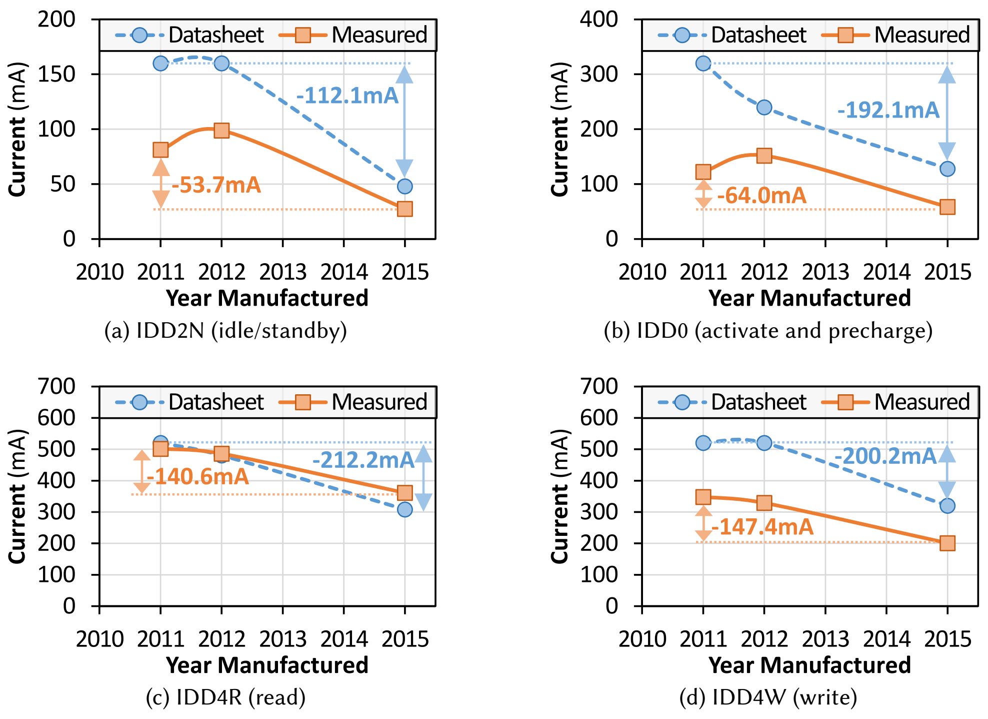
Number of Modules	**Total Number of Chips**	**Timing** (ns) ($t_{RCD}/t_{RP}/t_{RAS}$)	**Assembly Year**	**Supply Voltage**	**Max. Channel Frequency** (MT/s)
3 SO-DIMMs	24	13.75/13.75/35	2011	1.35 V	1333
4 SO-DIMMs	32	13.75/13.75/35	2012	1.35 V	1600

To compare the change in power consumption across generations, we measure the IDD values for each module. Figure 23 shows four of these IDD values, representing idle/standby (IDD2N), activate and precharge (IDD0), read (IDD4R), and write (IDD4W) currents. If we study the expected savings from the datasheet values (dotted blue lines), we see a general downward trend as modules move to newer process technologies (we plot the year of manufacture along the x-axis), indicating that the power consumption should have decreased. Based on our measurements, we make two key observations.

First, we observe that for each IDD value, *the actual power saved by switching to a newer-generation module, as measured by using our infrastructure, is significantly lower than the savings predicted by the datasheet.* For example, based on the datasheet, the *expected* decrease in IDD0 current (Figure 23b) for moving from a module manufactured in 2011 to a module manufactured in 2015 is 192.1 mA. In comparison, we measure an *actual* decrease of 64.0 mA, representing only 33.3% of the expected savings. For read and write operations, the difference is less drastic, but still statistically significant. Using IDD4W (Figure 23d) as an example, we see that the expected decrease from the datasheet is 200.2 mA, but the decrease measured from the actual DRAM modules is 147.4 mA, or 73.6% of the expected savings.

Second, we observe that in the case of IDD4R (Figure 23c), while the read power consumed by older-generation modules was within the IDD4R value in the datasheet, the lower-than-expected savings have caused the measured current to *exceed* the expected current based on the datasheet. This is in part due to the fact that the I/O driver current is included in our read current measurement due to the design of our infrastructure, but the I/O driver current is *not* included as part of the vendor-specified current. As we discuss in Section 4.3, the inclusion of the I/O driver current is a limitation of our measurement infrastructure, but we can eliminate the I/O driver current by applying correction mechanisms. Since the I/O driver current is a constant value for all of our measurements, the amount by which the measured read current *decreases* across generations is *not* affected by the I/O driver current. We find that the actual measured power savings for IDD4R is only 66.3% of the expected savings reported in the datasheets (a measured decrease of 140.6 mA vs. an expected decrease of 212.2 mA).

From our observations, we conclude that the power reduction from DRAM scaling is *not* as significant as expected from the datasheet values provided by DRAM vendors. With almost half of



(a) IDD2N (idle/standby)

(b) IDD0 (activate and precharge)

(c) IDD4R (read)

(d) IDD4W (write)

Fig. 23. Generational trends in IDD measurements.

the total system power now consumed by DRAM [38, 48, 56, 93, 104, 122, 157, 163], system designers may not be able to obtain the total system power savings they had expected by transitioning to newer DRAM models, which could adversely affect the amount of power and/or battery that is provisioned for a system.

8 SUMMARY OF KEY FINDINGS

We have presented extensive experimental characterization results and analyses of DRAM power consumption that capture a wide range of properties affecting the power consumption of real modern DRAM devices. We summarize our findings in four key conclusions:

(1) The current consumed by real DRAM modules *varies significantly* from the current specified in datasheets by the vendors, across all IDD values that we measure (Section 4). We comprehensively show that there is significant inter-vendor and intra-vendor module-to-module variation in DRAM power consumption.

(2) DRAM power consumption strongly *depends on the data value* that is read from or written to the DRAM chip, but does not strongly depend on the amount of bit toggling (Section 5).

(3) There is significant *structural variation* of power consumption within a DRAM chip, where the current varies based on which bank or row is accessed in the DRAM chip (Section 6).

(4) Across successive process technology generations, the actual power reduction of DRAM is *much lower* than the savings indicated by the vendor-specified IDD values in the datasheets (Section 7).

9 VAMPIRE: MODELING DRAM POWER

In order to overcome the shortcomings of existing power models, we use the new observations from our rigorous experimental characterization of real DRAM chips (Sections 4–6) to build the *Variation-Aware model of Memory Power Informed by Real Experiments* (VAMPIRE). To our knowledge, VAMPIRE is the first real-measurement-based power model for DRAM. By using our actual measurements from our characterization, VAMPIRE predicts a realistic value for DRAM power consumption, with little error. We validate VAMPIRE against microbenchmarks executed on our power measurement infrastructure to test and ensure its accuracy (see Section 9.1).

VAMPIRE takes in DRAM command traces, similar to the command traces used by DRAMPower [25, 27]. Each line of the command trace contains (1) the command name; (2) the target rank, bank, row, and column for the command, if applicable; and (3) for read and write commands, the 64-byte data that is read from or written to DRAM. By annotating the data alongside the write command, VAMPIRE can determine data-dependent power consumption. VAMPIRE also supports traces that do *not* include the written data values: users can instead manually input a certain distribution for the fraction of ones and the amount of bit toggling, which VAMPIRE uses to approximate the effect of data dependency on power consumption.

VAMPIRE consists of three core components: (1) read and write power modeling, which incorporates data-dependent behavior; (2) idle/activation/precharge power modeling; and (3) structural variation modeling. The first model component, read and write power modeling, uses the data-dependency-aware current models that we develop in Section 5.3. These current models incorporate the change in power consumption due to (1) the number of bits set to one in the data, (2) bit toggling, and (3) switching between different banks and columns. The second model component, idle/activation/precharge power modeling, uses the measurements from Section 4 to capture the actual power consumed for all DRAM commands aside from read and write. This includes the power consumed by activate operations, precharge operations, refresh operations and power-down modes, and when a DIMM is idle. The third model component, structural variation modeling, uses our characterization results from Section 6 to adjust the estimated current based on which bank and row are accessed.

VAMPIRE outputs a separate power value for each vendor, and can account for process variation by outputting a range of power values based on the impact of variation, as estimated from the variation that we capture in our experimental characterization. We plan to integrate VAMPIRE into several memory system simulators (e.g., gem5 [11], DRAMSim2 [131], Ramulator [83, 133], NVSim [45]), and will open-source the model [135].

9.1 Model Validation

We use a new series of experimental measurements, which were *not* used to construct VAMPIRE, to *validate* the accuracy of our model and compare it with the accuracy of two popular state-of-the-art DRAM power models: the Micron power calculator [111] and DRAMPower [25, 27]. Both models are based off of worst-case IDD values reported in vendor datasheets, and *neither* of them models process variation, data-dependent power consumption, or structural variation. We use the extrapolated IDD values that we calculate in Section 4 as parameter inputs into both the Micron power calculator and DRAMPower.

In the validation experiments, we execute the following sequence of commands: {*activate*, $n\times$*read*, *precharge*}, where we sweep various values of n between 0 and 764.[4] Each read operation reads a cache line where all bytes of the cache line contain the data value 0xAA. All reads are performed to Bank 0, Row 128, and back-to-back reads are interleaved across different columns (see Section 5.2. For

[4]We ensure that all DRAM timing constraints are met for each experiment.

the validation experiments, we measure the power consumption of 22 DRAM modules (8 modules for Vendor A, and 7 modules each for Vendors B and C), where the modules of each vendor are selected randomly from the 50 modules listed in Table 1. We generate traces that capture the behavior of each experiment, and then feed them into each of the DRAM power models that we evaluate.

Figure 24 shows the *mean absolute percentage error* (MAPE) across each of our validation experiments, for each DRAM power model compared to the measured current of each vendor's DRAM modules. We make three observations from the figure. First, the Micron power model has a very high error across all three vendors, with a MAPE of 160.6%, averaged across all three vendors. The Micron model significantly overestimates the power consumption of DRAM, as prior work has shown [25, 65], since it does *not* accurately model a number of important phenomena, such as the fact that when only one bank is active, the DRAM module consumes much less power than when all banks are active [65]. Second, DRAMPower has a MAPE of 32.4%, averaged across all three vendors. While this is lower than the error of the Micron power model, DRAMPower still has high error rates for Vendors A and B. This is because DRAMPower does *not* capture the impact of our four observations (see Section 8). The largest source of DRAMPower's high error rates is its inability to accurately model (1) the impact of data dependency on the power consumed during each read operation; and (2) the much lower power consumed by activate and precharge operations and during idle time, as compared to the IDD0 and IDD2N values. Third, VAMPIRE has a MAPE of only 6.8%, averaged across all three vendors. Unlike the Micron power model and DRAMPower, VAMPIRE has a low MAPE for all three vendors (with the highest per-vendor MAPE being 7.1%).

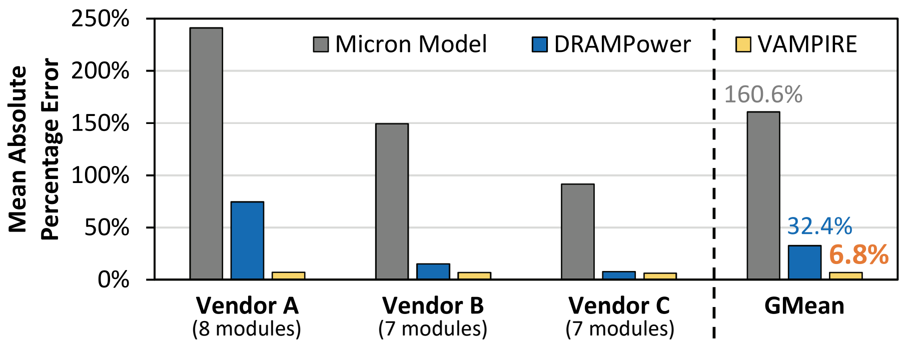


Fig. 24. Mean absolute percentage error of state-of-the-art DRAM power models and of VAMPIRE, compared to real measured DRAM power.

We conclude that VAMPIRE is significantly more accurate than state-of-the-art DRAM power models, because it incorporates our new observations on (1) the large differences between the real measured DRAM power and the vendor-provided IDD values, (2) data-dependent DRAM power consumption, and (3) the impact of structural variation on DRAM power consumption.

9.2 Evaluating DRAM Power Consumption with Large Applications

In addition to the measurement-based validation, we compare the power consumption reported by VAMPIRE to the power consumption reported by DRAMPower [25, 27] (the best state-of-the-art DRAM power model) when we simulate the memory access behavior of real applications. Unfortunately, we cannot compare the reported power consumption numbers to real DRAM power measurements, due to the inability of SoftMC to interactively execute command traces from full applications [53]. Instead, we measure the *relative error* of DRAMPower, which does not capture several aspects of DRAM power consumption, with respect to VAMPIRE, which captures all of the key observations that we make based on our experimental characterization.

For each DRAM power model, we determine the power consumed by a single channel of DDR3L memory while executing applications on a single CPU core. Table 4 shows the system configuration that we simulate. We evaluate 23 applications from the SPEC CPU2006 suite [146]. We use Pin [100] to record the last level cache misses generated by each application. We fast-forward each application past its initialization phase, and collect a memory trace for a representative 100 million instruction portion. We generate the input for each model by executing the memory trace on Ramulator [83, 133], an open-source DRAM system simulator. We modify Ramulator to output the correct format of DRAM commands for VAMPIRE, which includes data values for write commands.

Table 4. Evaluated system configuration.

Processor	x86-64 ISA, one core, 3.2 GHz, 128-entry instruction window
Cache	L1: 64 kB, 4-way associative; L2: 2 MB, 16-way associative
Memory Controller	64/64-entry read/write request queues, FR-FCFS [130, 167]
DRAM	DDR3L-800 [61], 1 channel, 1 rank/8 banks per channel

Figure 25 shows the relative error for the Micron power model compared to VAMPIRE. We show the relative error as a box plot, where the box represents the quartiles of the output for each application. We observe that there is significant error in DRAMPower compared to VAMPIRE. The average relative errors of DRAMPower for Vendors A, B, and C are 58.3%, 45.0%, and 33.5%, respectively. From the figure, we observe that the error actually varies significantly from application to application. This is because relative error of each application is highly dependent on the application's memory access behavior. In general, DRAMPower reports a much higher power consumption value than VAMPIRE for applications that are memory intensive, and reports a lower power consumption value than VAMPIRE for applications that are not memory intensive. We conclude that the properties of DRAM power consumption that are missing from DRAMPower greatly affect its reported power consumption. VAMPIRE, by accurately modeling key properties that affect DRAM power consumption, provides much more accurate estimates of DRAM power consumption by large applications.

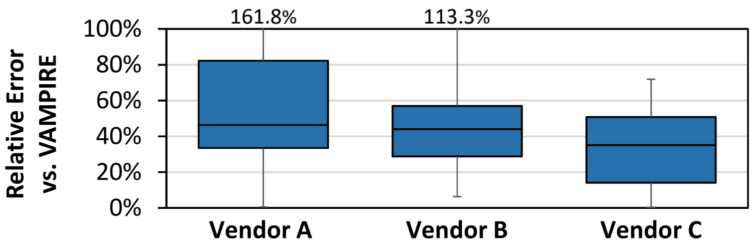


Fig. 25. Box plots showing the distribution of the relative error for the DRAMPower model compared to VAMPIRE, across the applications that we simulate. Each box illustrates the quartiles of the distribution of the relative error for each application, and the whiskers illustrate the minimum and maximum error values, across all applications evaluated.

9.3 Example Applications of VAMPIRE

A large fraction of the error that we observe in existing power models is likely the result of missing data dependency and variation characteristics in these models. By capturing these characteristics in detail, VAMPIRE allows researchers, architects and system designers to accurately evaluate and take into account the various sources of power consumption in modern DRAM modules. This has several implications for DRAM architectures and system designs, of which we present three examples. First, taking advantage of the systematic structural variation that VAMPIRE captures across different banks and rows in a DRAM module, a system designer can rewrite the virtual memory manager in the operating system to optimize physical page allocation for low energy consumption. Instead of treating all physical page locations within DRAM as equal, the virtual memory manager could allocate data that is accessed more frequently to those physical pages that reside in banks and rows that consume less power. Second, VAMPIRE's insights can be used to determine when to schedule power-down modes for DRAM. VAMPIRE provides accurate information on the actual power saved during power-down mode by a DRAM module, and on the actual power required to wake the module back up to full-power mode, allowing designers and architects to accurately predict whether there is enough time spent in power-down mode to amortize the performance and power overheads of powering down and waking up the module. Third, VAMPIRE's model of data-dependent power consumption can be used to design alternate data encodings that reduce power consumption within a DRAM chip. We discuss one such data encoding in Section 10, and show how it takes advantage of the data-dependent behavior captured by VAMPIRE to reduce DRAM energy. We believe there are many other use cases of VAMPIRE, and leave it to future work to uncover such other use cases.

10 CASE STUDY: DATA ENCODING

VAMPIRE enables a wide range of studies that were *not* possible using existing models, because it captures characteristics such as data dependency and structural variation that the existing models do *not* take into account. An example of such a study is exploring how the DRAM power consumption changes for different *data encodings*, i.e., the mechanisms with which the memory controller and/or DRAM module transform the cache line data values that are stored in the DRAM module. In this section, we examine the potential of cache line encodings that exploit the data dependency of DRAM power consumption.

Prior studies on specialized data encodings for DRAM [47, 55, 124, 144, 145, 158] have largely focused on minimizing the amount of bit toggling that takes place on the off-chip memory channel (see Section 11.4). When a 64-byte cache line is transmitted along the 64-bit memory channel, the data is split up into eight *bursts*, and is sent one burst at a time. This can increase DRAM power consumption due to bit toggling across different bursts *from the same cache line*. A number of studies [7, 12, 55, 124, 144, 145, 150] have shown the increased power consumption due to this inter-burst bit toggling. In contrast to all these prior studies, we study the data dependency and bit toggling that takes place *within* the DRAM chip, where the bit toggling occurs across *different cache lines* (see Section 5.2). As we discuss in Section 5, the amount of power consumed during read and write operations depends on the number of ones in the cache line, and on the number of bits that are toggled within DRAM.

10.1 Encodings Studied

We examine how four different cache-line-level data encodings can affect DRAM power consumption when they are applied to the data before the data is written to DRAM:

- *Baseline*: The data is not encoded before being transferred to DRAM.

- *Base-Delta Immediate* (BDI) [127]: We apply BDI compression to the data. Prior work [124] has shown that many compression algorithms, including BDI, can consume more power than *Baseline* as a result of the bit toggling that takes place on the memory channel.
- *Optimized*: This per-byte encoding scheme encodes the most- frequently-used byte values using the least number of ones in the encoded byte value. For each application, we sort all possible byte values (i.e., 0–255) based on their frequency of occurrence.[5] The byte values are then assigned to encoded values such that the most frequent byte value is assigned to the encoded value with the least number of ones (i.e., an encoded value of zero), and the least frequent byte value is assigned to the encoded value with the most ones. This encoding has two advantages. First, it minimizes the number of ones used for the application, which is likely to reduce the read power, as the power consumed by a read operation increases when the data that is read contains a greater number of ones. Second, it reduces the probability of toggling many bits at the same time between transmissions of different cache lines. On the other hand, one drawback of this encoding scheme is that this increases the power consumed by write operations, as the data-dependent power consumption of writes has an inverse relationship with the number of ones as that of reads.
- *Optimized with Write Inversion* (OWI): We develop a variation of our *Optimized* encoding, to minimize DRAM power consumption for *both* reads and writes. Our measurements in Section 5 show that read power increases with the number of ones in the cache line, and write power *decreases* with the number of ones, due to I/O driver power and data-dependent power consumption within DRAM. In order to maximize the power savings, we assume that cache lines that are to be written to DRAM are first transformed using our Optimized encoding, and then *inverted* (i.e., bitwise complemented) by the memory controller. Once the data that is to be written passes through the I/O drivers and peripheral circuitry within the module, the DRAM chip inverts (i.e., bitwise complements) the data before it is written to the DRAM cells.

The overhead to implement the *Optimized* and *OWI* encodings is small. We assume that both encoding mechanisms use 256 bit × 8 bit lookup tables in each DRAM chip. We use CACTI 7.0 [54] to estimate the area and latency of the lookup table using a 22 nm manufacturing process technology, and find that each table requires only 0.0024 mm^2 of area, and can perform a lookup in 0.134 ns. The lookup tables are hard-coded, and we conservatively assume that the lookup adds one DRAM cycle of latency. This latency is similar to that incurred by channel encodings such as data bus inversion [55, 144, 145], which is an optional feature in LPDDR4 memory [64].

We use the same simulation methodology described in Section 9, and develop a tool to encode the data recorded by Pin [100] and Ramulator [83, 133] for each encoding. To account for the performance overhead of the *Optimized* and *OWI* encodings, we add one DRAM cycle to each read and write operation in Ramulator. We account for the additional energy consumed for encoding data using *Optimized* and *OWI*.

10.2 Evaluation

Figure 26 shows the average energy consumed inside DRAM for each encoding, normalized to the energy consumption of *Baseline*, for each vendor. We make three observations from the figure. First, the *OWI* encoding, which optimizes the number of ones for both reads and writes, achieves significant savings. *OWI* reduces the energy consumption over *Baseline* by 12.2% on average, and up to 28.6%. Second, our *Optimized* encoding provides no tangible reduction in energy, as the

[5]Note that our goal is to perform limit studies to gauge the potential of data encodings that exploit DRAM power variation. As such, we do not assess the practicality of sorting all byte values that are read and written by an application. We leave such considerations for future work, which we hope will develop a more practical encoding mechanism.

increase in write current energy due to data dependency cancels out any savings from optimizing the read current. Third, data compressed with BDI consumes no more energy than uncompressed data. While compression may increase the amount of energy consumed on the memory channel, we see no notable effect *within* DRAM, due to the relatively minor impact bit toggling has on the current observed in our real chip measurements (see Section 5.2).

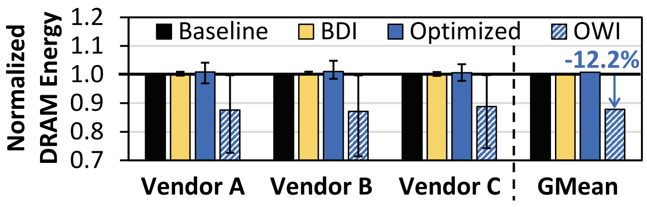


Fig. 26. Average energy consumption for various cache-line-level data encodings, normalized to energy consumed with *Baseline* encoding. Error bars show maximum and minimum normalized energy values across all applications.

We conclude that (1) DRAM energy can be reduced by using a power-aware and DRAM-data-dependence-aware data encoding mechanism; and (2) VAMPIRE, our new power model, enables the exploration of the energy consumption of the encodings we have examined, as well as the new encodings that we hope future research will develop.

11 RELATED WORK

Several DRAM power and energy models exist. Most of these models are derived from the IDD(current) values provided by DRAM manufacturers in their datasheets for worst-case power consumption. To our knowledge, our work is the first to (1) characterize the power consumption of a large number of real DRAM modules from three major DRAM vendors, across a wide variety of tests; (2) demonstrate how IDD values often deviate significantly from actual DRAM power consumption and contain large guardbands; (3) comprehensively demonstrate the inter-vendor and intra-vendor module-to-module variation of DRAM power consumption; (4) show that DRAM power consumption depends on data values and structural variation; (5) show that DRAM power consumption has *not* decreased as much as vendor specifications indicate over successive generations; and (6) develop a DRAM power model that accounts for the impact of IDD value guardbands, data dependency, and structural variation on the power that is consumed.

11.1 Architectural Power Models

The Micron DRAM power model [111] uses IDD values, command count, execution time, and timing parameters from datasheets to calculate power consumption. However, prior works [26, 65] have shown that the Micron power model does *not* (1) account for any additional time that may elapse between two DRAM commands [26], (2) model an open-row policy [26] or more sophisticated row buffer management policies, or (3) properly account for the power consumed when the number of active banks changes [65]. As we discuss in Section 9.1, we find that the Micron power model also does *not* take into account typical-case DRAM power consumption (which, as we show in Section 4, is much lower than the IDD values specified by vendors in datasheets), data-dependent power consumption, or the impact of structural variation on power consumption in DRAM.

DRAMPower [26, 27] is an open-source tool that can be used at the command level and transaction level to estimate power consumption. It allows DRAM command traces to be logged and also has an optional command scheduler, which emulates a memory controller. While DRAMPower accounts for three of the major characteristics that are missing from the Micron power model (the additional time between commands, alternative row buffer management policies, and the number of active banks), it is still predominantly based upon the worst-case IDD estimates extracted from datasheets. With the exception of (1) the change in active background power based on the number of banks open [65, 107] and (2) IDD values and module-to-module variation for a small number (10) of tested DRAM modules [27], the power models and input parameters employed by DRAMPower are *not* based off of measured data, and they do *not* take into account data dependency, a comprehensive notion of module-to-module variation, or structural variation in DRAM power. To our knowledge, the limited measurements from real devices used in DRAMPower are performed on a small number of DRAM modules, do not capture any variation trends, and do not span across multiple vendors. As we show in Section 9.1, because it is still based off of worst-case IDD values, DRAMPower has a mean absolute percentage error of 32.4% compared to real measured power in the DRAM chips that we test. In contrast, VAMPIRE, our power model, is based fully off of measured data, and captures the impact of data dependency and structural variation on power consumption, providing high accuracy (only 6.8% error, as shown in Section 9.1).

CACTI-D [34] and Vogelsang [154] use circuit-level models to characterize the power consumed by DRAM. The peripheral circuitry used in CACTI-D [34] is largely based on SRAM caches, and does not accurately reflect the design of DRAM peripheral logic. Vogelsang [154] uses transistor-level models to predict future trends in DRAM power consumption, but the models are calibrated to datasheet IDD values, and do not capture data dependency, module-to-module variation, or structural variation. Similarly, the PADRAM model [85] models a subset of RDRAM [129] components and vendor specifications to develop a DRAM power model. Aside from not capturing real DRAM power behavior, the PADRAM model is designed for RDRAM modules, and is not fully compatible with the modern DDR SDRAMs that we characterize.

Wattch [13], McPAT [95], and the model by Fan et al. [49] are models designed to capture the power dissipation and energy consumption of modern processors. While these models can include a DRAM power component, this component again relies on the IDD datasheet values.

11.2 Low-Power DRAM

Prior works propose models, chip designs, and architectures to reduce DRAM power consumption. A number of works [31, 36, 38] study how to reduce the voltage and/or frequency of DRAM to lower power consumption. Several prior works exploit low-power modes to increase the time spent in a low-power state [1, 4, 5, 10, 39, 42–44, 49, 67–69, 85, 103, 147, 148]. Row-buffer-aware DRAM designs [70, 82, 149] optimize data placement to increase row buffer hits and, thus, reduce the energy spent on row activation and precharge. A number of DRAM architectures reduce DRAM power by activating only a fraction of the row [33, 37, 151, 165], a fraction of the bitlines [90], or a fraction of the DRAM chips on a module [156, 164, 166], by reducing the access latency [30, 52, 89, 92, 139], or by reducing the operating frequencies of some layers in a 3D-stacked DRAM chip [88].

Many works study eliminating margins designed for worst-case DRAM modules to improve energy efficiency. Various works [2, 9, 74–76, 96–98, 118, 121, 128, 152] reduce DRAM refresh power by characterizing cell retention times and reducing unnecessary refresh operations. Multiple works [25, 30, 32, 52, 82, 89, 90, 92, 113, 141] make DRAM more energy efficient by reducing the latencies of DRAM operations.

Low-power DDR (LPDDR) [63, 64] is a family of DRAM architectures designed by JEDEC for use in low-power systems (e.g., mobile devices). LPDDR architectures employ major design changes over conventional DDR memory architectures. Two such changes include the use of a low-voltage-swing I/O interface (which, in LPDDR4 DRAM, consumes 40% less I/O power than DDR4 DRAM [36]), and the inclusion of additional low-power modes that make use of lower supply voltage levels that are not available in DDR memory.

None of these works characterize DRAM power consumption, and their ideas are orthogonal to ours.

11.3 Experimental DRAM Characterization

Various experimental studies [25, 31, 32, 53, 58, 65, 66, 74–77, 79, 84, 89, 92, 97, 108, 115, 121, 128, 136, 142, 143] characterize DRAM reliability, data retention, and latency by measuring characteristics of real DRAM chips, but they do not measure power consumption. Kim et al. [78] study DRAM energy consumption under different processor cache configurations, but do not study how different DRAM operations contribute to energy consumption in modern DRAM devices. Jung et al. [65] experimentally characterize a limited subset of IDD values to study the effect of the number of open banks on active background power, but do not characterize any other aspect of DRAM power consumption. A subsequent work by Jung et al. [66] demonstrates a power measurement platform for DDR3 DRAM, but does not (1) comprehensively report power consumption across all DRAM operations; or (2) study the effects of process variation, data dependency, or structural variation on power consumption. In Section 4, we measure a comprehensive set of IDD values, and we show how the power consumption differs and varies significantly from the values provided in the datasheets.

11.4 Compression and Encoding Schemes

Prior works propose using compression and encoding schemes for caches [3, 35, 46, 125, 127, 153, 162] and memory [47, 126, 158]. Most of these are pattern-based schemes and perform compression at a word granularity. Base-Delta-Immediate compression [126, 127] performs compression at a cache line granularity, by identifying cache lines where the data value of each word in the line is within a small range of values. Data bus inversion (DBI) [55, 144, 145] is an encoding that reduces the power consumed by the memory channel, by inverting the data transmitted during each data burst when the data contains more zeroes than ones. DBI is an optional feature that can be enabled in LPDDR4 memories [64].

These prior works either do not study the impact of compression encodings on DRAM power consumption, or do not study the impact of implementing different encodings within the DRAM chip. In Section 10, we study how a new power-aware cache line encoding mechanism can reduce energy consumption in a DRAM chip by exploiting our observations on data-dependent DRAM power consumption.

12 CONCLUSION

DRAM power consumption is a critical issue in contemporary computer systems, as DRAM now accounts for as much as half of the total system power consumption. While there is a pressing need to invent new low-power DRAM architectures, existing DRAM power models do *not* accurately model the power consumed by DRAM modules, limiting researchers' understanding of the sources of power consumption. The existing DRAM power models are inaccurate because they rely only on vendor-specified current measurements, and do *not* capture several important characteristics of power consumption that are present in real DRAM devices.

To address the shortcomings of existing DRAM power models, we first perform an extensive *experimental characterization* of the power consumed by real state-of-the-art DDR3L DRAM devices. We measure the current consumed by 50 DRAM modules from three major vendors, and make four key new observations that previous models did not capture: (1) the actual current consumed deviates significantly from the vendor specifications in the datasheets; (2) the data value that is read from or written to DRAM significantly impacts power consumption; (3) power consumption varies significantly based on which bank or row of a DRAM module is being accessed; and (4) across successive process technology generations, the actual power [reduction is often much lower than the savings indicated by vendor specifications.

Based on our real device measurements and analysis, we build VAMPIRE, a new, accurate DRAM power model. VAMPIRE enables studies that could *not* be performed using prior DRAM power models. For example, we show that a new power-aware data encoding scheme can reduce DRAM power consumption by an average of 12.2% (up to 28.6%). We will release VAMPIRE and all of our raw measurement data online [135]. We hope that the findings in this work and our new power model will inspire new research directions, new ideas, and rigorous and more accurate evaluations in power-and energy-aware memory system design.

ACKNOWLEDGMENTS

We thank our shepherd Thomas Wenisch, the anonymous reviewers, and SAFARI Research Group members for feedback. Thanks to Naveen Kakarla for his assistance with the experimental validation of VAMPIRE. We acknowledge the generous support of our industrial partners (Google, Intel, Microsoft, NVIDIA, Samsung, and VMware) and the United States Department of Energy. This research was supported in part by the Semiconductor Research Corporation and the National Science Foundation (grants 1212962 and 1320531).

REFERENCES

[1] N. Aggarwal, J. F. Cantin, M. H. Lipasti, and J. E. Smith, "Power-Efficient DRAM Speculation," in *HPCA*, 2008.
[2] A. Agrawal, A. Ansari, and J. Torrellas, "Mosaic: Exploiting the Spatial Locality of Process Variation to Reduce Refresh Energy in On-Chip eDRAM Modules," in *HPCA*, 2014.
[3] A. R. Alameldeen and D. A. Wood, "Adaptive Cache Compression for High-Performance Processors," in *ISCA*, 2004.
[4] A. M. Amin and Z. A. Chishti, "Rank-Aware Cache Replacement and Write Buffering to Improve DRAM Energy Efficiency," in *ISLPED*, 2010.
[5] V. Anagnostopoulou, S. Biswas, H. Saadeldeen, A. Savage, R. Bianchini, T. Yang, D. Franklin, and F. T. Chong, "Barely Alive Memory Servers: Keeping Data Active in a Low-Power State," *ACM JETC*, 2012.
[6] A. Bakhoda, G. Yuan, W. W. L. Fung, H. Wong, and T. M. Aamodt, "Analyzing CUDA Workloads Using a Detailed GPU Simulator," in *ISPASS*, 2009.
[7] B. M. Beckmann and D. A. Wood, "TLC: Transmission Line Caches," in *MICRO*, 2003.
[8] I. Bhati, Z. Chishti, and B. Jacob, "Coordinated Refresh: Energy Efficient Techniques for DRAM Refresh Scheduling," in *ISLPED*, 2013.
[9] I. Bhati, Z. Chishti, S. Lu, and B. Jacob, "Flexible Auto-Refresh: Enabling Scalable and Energy-Efficient DRAM Refresh Reductions," in *ISCA*, 2015.
[10] M. Bi, R. Duan, and C. Gniady, "Delay-Hiding Energy Management Mechanisms for DRAM," in *HPCA*, 2010.
[11] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, "gem5: A Multiple-ISA Full System Simulator with Detailed Memory Model," *CAN*, vol. 39, June 2011.
[12] M. N. Bojnordi and E. İpek, "DESC: Energy-Efficient Data Exchange Using Synchronized Counters," in *MICRO*, 2013.
[13] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: A Framework for Architectural-Level Power Analysis and Optimizations," in *ISCA*, 2000.
[14] Y. Cai, S. Ghose, Y. Luo, K. Mai, O. Mutlu, and E. F. Haratsch, "Vulnerabilities in MLC NAND Flash Memory Programming: Experimental Analysis, Exploits, and Mitigation Techniques," in *HPCA*, 2017.
[15] Y. Cai, Y. Luo, S. Ghose, E. F. Haratsch, K. Mai, and O. Mutlu, "Read Disturb Errors in MLC NAND Flash Memory: Characterization and Mitigation," in *DSN*, 2015.
[16] Y. Cai, Y. Luo, E. F. Haratsch, K. Mai, and O. Mutlu, "Data Retention in MLC NAND Flash Memory: Characterization, Optimization, and Recovery," in *HPCA*, 2015.
[17] Y. Cai, G. Yalcin, O. Mutlu, E. F. Haratsch, A. Cristal, O. Unsal, and K. Mai, "Flash Correct and Refresh: Retention Aware Management for Increased Lifetime," in *ICCD*, 2012.
[18] Y. Cai, G. Yalcin, O. Mutlu, E. F. Haratsch, A. Cristal, O. Unsal, and K. Mai, "Error Analysis and Retention-Aware Error Management for NAND Flash Memory," *Intel Technol. J.*, May 2013.

[19] Y. Cai, S. Ghose, E. F. Haratsch, Y. Luo, and O. Mutlu, "Error Characterization, Mitigation, and Recovery in Flash-Memory-Based Solid-State Drives," *Proceedings of the IEEE*, 2017.

[20] Y. Cai, S. Ghose, E. F. Haratsch, Y. Luo, and O. Mutlu, "Reliability Issues in Flash-Memory-Based Solid-State Drives: Experimental Analysis, Mitigation, Recovery," in *Inside Solid State Drives (SSDs)*, 2nd ed. Springer Nature, 2018.

[21] Y. Cai, E. F. Haratsch, O. Mutlu, and K. Mai, "Error Patterns in MLC NAND Flash Memory: Measurement, Characterization, and Analysis," in *DATE*, 2012.

[22] Y. Cai, E. F. Haratsch, O. Mutlu, and K. Mai, "Threshold Voltage Distribution in MLC NAND Flash Memory: Characterization, Analysis, and Modeling," in *DATE*, 2013.

[23] Y. Cai, O. Mutlu, E. F. Haratsch, and K. Mai, "Program Interference in MLC NAND Flash Memory: Characterization, Modeling, and Mitigation," in *ICCD*, 2013.

[24] Y. Cai, G. Yalcin, O. Mutlu, E. F. Haratsch, O. Unsal, A. Cristal, and K. Mai, "Neighbor-Cell Assisted Error Correction for MLC NAND Flash Memories," in *SIGMETRICS*, 2014.

[25] K. Chandrasekar, S. Goossens, C. Weis, M. Koedam, B. Akesson, N. Wehn, and K. Goossens, "Exploiting Expendable Process-Margins in DRAMs for Run-Time Performance Optimization," in *DATE*, 2014.

[26] K. Chandrasekar, B. Akesson, and K. Goossens, "Improved Power Modelling of DDR SDRAMs," in *DSD*, 2011.

[27] K. Chandrasekar, C. Weis, Y. Li, S. Goossens, M. Jung, O. Naji, B. Akesson, N. Wehn, and K. Goossens, "DRAMPower: Open-Source DRAM Power & Energy Estimation Tool," http://www.drampower.info.

[28] K. K. Chang, "Understanding and Improving the Latency of DRAM-Based Memory Systems," Ph.D. dissertation, Carnegie Mellon Univ., 2017.

[29] K. K. Chang, D. Lee, Z. Chishti, A. Alameldeen, C. Wilkerson, Y. Kim, and O. Mutlu, "Improving DRAM Performance by Parallelizing Refreshes with Accesses," in *HPCA*, 2014.

[30] K. K. Chang, P. J. Nair, D. Lee, S. Ghose, M. K. Qureshi, and O. Mutlu, "Low-Cost Inter-Linked Subarrays (LISA): Enabling Fast Inter-Subarray Data Movement in DRAM," in *HPCA*, 2016.

[31] K. K. Chang, A. G. A. G. Yağlıkçı, S. Ghose, A. Agrawal, N. Chatterjee, A. Kashyap, D. Lee, M. O'Connor, H. Hassan, and O. Mutlu, "Understanding Reduced-Voltage Operation in Modern DRAM Devices: Experimental Characterization, Analysis, and Mechanisms," in *SIGMETRICS*, 2017.

[32] K. K. Chang, A. Kashyap, H. Hassan, S. Ghose, K. Hsieh, D. Lee, T. Li, G. Pekhimenko, S. Khan, and O. Mutlu, "Understanding Latency Variation in Modern DRAM Chips: Experimental Characterization, Analysis, and Optimization," in *SIGMETRICS*, 2016.

[33] N. Chatterjee, M. O'Connor, D. Lee, D. R. Johnson, M. Rhu, S. W. Kecker, and W. J. Dally, "Architecting an Energy-Efficient DRAM System for GPUs," in *HPCA*, 2017.

[34] K. Chen, S. Li, N. Muralimanohar, J. H. Ahn, J. B. Brockman, and N. P. Jouppi, "CACTI-3DD: Architecture-Level Modeling for 3D Die-Stacked DRAM Main Memory," in *DATE*, 2012.

[35] X. Chen, L. Yang, R. Dick, L. Shang, and H. Lekatsas, "A High-Performance Microprocessor Cache Compression Algorithm," *TVLSI*, 2010.

[36] J. Y. Choi, "LPDDR4: Evolution for New Mobile Worlds," in *MEMCON*, 2013.

[37] E. Cooper-Balis and B. Jacob, "Fine-Grained Activation for Power Reduction in DRAM," *IEEE Micro*, 2010.

[38] H. David, C. Fallin, E. Gorbatov, U. R. Hanebutte, and O. Mutlu, "Memory Power Management via Dynamic Voltage/Frequency Scaling," in *ICAC*, 2011.

[39] V. De La Luz, M. Kandemir, and I. Kolcu, "Automatic Data Migration for Reducing Energy Consumption in Multi-Bank Memory Systems," in *DAC*, 2002.

[40] V. De La Luz, M. Kandemir, N. Vijaykrishnan, A. Sivasubramaniam, and M. J. Irwin, "DRAM Energy Management Using Software and Hardware Directed Power Mode Control," in *HPCA*, 2001.

[41] V. De La Luz, A. Sivasubramaniam, M. Kandemir, N. Vijaykrishnan, and M. J. Irwin, "Scheduler Based DRAM Energy Management," in *DAC*, 2002.

[42] Q. Deng, D. Meisner, L. Ramos, T. F. Wenisch, and R. Bianchini, "MemScale: Active Low-Power Modes for Main Memory," in *ASPLOS*, 2011.

[43] Q. Deng, D. Meisner, L. Ramos, T. F. Wenisch, and R. Bianchini, "Active Low-Power Modes for Main Memory with MemScale," in *MICRO*, 2012.

[44] B. Diniz, D. Guedes, J. W. Meira, and R. Bianchini, "Limiting the Power Consumption of Main Memory," in *ISCA*, 2007.

[45] X. Dong, C. Xu, Y. Xie, and N. P. Jouppi, "NVSim: A Circuit-Level Performance, Energy, and Area Model for Emerging Nonvolatile Memory," *TCAD*, June 2012.

[46] J. Dusser, T. Piquet, and A. Seznec, "Zero-Content Augmented Caches," in *ICS*, 2009.

[47] M. Ekman and P. Stenström, "A Robust Main-Memory Compression Scheme," in *ISCA*, 2005.

[48] R. Elmore, K. Gruchalla, C. Phillips, A. Purkayastha, and N. Wunder, "An Analysis of Application Power and Schedule Composition in a High Performance Computing Environment," National Renewable Energy Laboratory, Tech Report NREL/TP-2C00-65392, 2016.

[49] X. Fan, C. S. Ellis, and A. R. Lebeck, "Memory Controller Policies for DRAM Power Management," in *ISLPED*, 2001.

[50] M. Ferdman, A. Adileh, O. Kocberber, S. Volos, M. Alisafaee, D. Jevdjic, C. Kaynak, A. D. Popescu, A. Ailamaki, and B. Falsafi, "Clearing the Clouds: A Study of Emerging Scale-Out Workloads on Modern Hardware," in *ASPLOS*, 2012.

[51] C. F. Gauss, *Theoria Motus Corporum Coelestium in Sectionibus Conicis Solem Ambientium.* F. Perthes et I. H. Besser, 1809.

[52] H. Hassan, G. Pekhimenko, N. Vijaykumar, V. Seshadri, D. Lee, O. Ergin, and O. Mutlu, "ChargeCache: Reducing DRAM Latency by Exploiting Row Access Locality," in *HPCA*, 2016.

[53] H. Hassan, N. Vijaykumar, S. Khan, S. Ghose, K. Chang, G. Pekhimenko, D. Lee, O. Ergin, and O. Mutlu, "SoftMC: A Flexible and Practical Open-Source Infrastructure for Enabling Experimental DRAM Studies," in *HPCA*, 2017.

[54] Hewlett Packard Enterprise, "CACTI 7.0," https://github.com/HewlettPackard/cacti.

[55] T. M. Hollis, "Data Bus Inversion in High-Speed Memory Applications," *TCAS II*, 2009.

[56] U. Holzle and L. A. Barroso, *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines.* Morgan & Claypool, 2009.

[57] S. Hong, "Memory Technology Trend and Future Challenges," in *IEDM*, 2010.

[58] A. Hwang, I. Stefanovici, and B. Schroeder, "Cosmic Rays Don't Strike Twice: Understanding the Nature of DRAM Errors and the Implications for System Design," in *ASPLOS*, 2012.

[59] M. Inoue, T. Yamada, H. Kotani, H. Yamauchi, A. Fujiwara, J. Matsushima, H. Akamatsu, M. Fukumoto, M. Kubota, I. Nakao, N. Aoi, G. Fuse, S. Ogawa, S. Odanaka, A. Ueno, and H. Yamamoto, "A 16-Mbit DRAM with a Relaxed Sense-Amplifier-Pitch Open-Bit-Line Architecture," *JSSC*, 1988.

[60] JEDEC Solid State Technology Assn., *JESD79-3F: DDR3 SDRAM Standard*, 2012.

[61] JEDEC Solid State Technology Assn., *JESD79-3-1A.01: Addendum No.1 to JESD79-3 - 1.35V DDR3L-800, DDR3L-1066, DDR3L-1333, DDR3L-1600, and DDR3L-1866*, 2013.

[62] JEDEC Solid State Technology Assn., *JESD21C, Module 4.20.18: 204-Pin DDR3 SDRAM Unbuffered SO-DIMM Design Specification*, 2014.

[63] JEDEC Solid State Technology Assn., *JESD209-3C: Low Power Double Data Rate 3 SDRAM (LPDDR3) Standard*, 2015.

[64] JEDEC Solid State Technology Assn., *JESD209-4B: Low Power Double Data Rate 4 (LPDDR4) Standard*, 2017.

[65] M. Jung, D. M. Mathew, É. F. Zulian, C. Weis, and N. Wehn, "A New Bank Sensitive DRAMPower Model for Efficient Design Space Exploration," in *PATMOS*, 2016.

[66] M. Jung, D. M. Mathew, C. C. Rheinländer, C. Weis, and N. Wehn, "A Platform to Analyze DDR3 DRAM's Power and Retention Time," *IEEE Design and Test*, 2017.

[67] M. Kandemir, O. Ozturk, and M. Karakoy, "Dynamic On-Chip Memory Management for Chip Multiprocessors," in *CASES*, 2004.

[68] M. Kandemir, U. Sezer, and V. De La Luz, "Improving Memory Energy Using Access Pattern Classification," in *ICCAD*, 2001.

[69] M. Kandemir, T. Yemliha, S. W. Son, and O. Ozturk, "Memory Bank Aware Dynamic Loop Scheduling," in *DATE*, 2007.

[70] D. Kaseridis, J. Stuechelia, and L. K. John, "Minimalist Open-Page: A DRAM Page-Mode Scheduling Policy for the Many-Core Era," in *MICRO*, 2011.

[71] B. Keeth, R. J. Baker, B. Johnson, and F. Lin, *DRAM Circuit Design: Fundamental and High-Speed Topics.* Wiley-IEEE Press, 2007.

[72] Keysight Technologies, Inc., *34134A AC/DC DMM Current Probe: User's Guide*, https://literature.cdn.keysight.com/litweb/pdf/34134-90001.pdf, 2009.

[73] Keysight Technologies, Inc., *Keysight Truevolt Series Digital Multimeters: Operating and Service Guide*, https://literature.cdn.keysight.com/litweb/pdf/34460-90901.pdf, 2017.

[74] S. Khan, D. Lee, Y. Kim, A. R. Alameldeen, C. Wilkerson, and O. Mutlu, "The Efficacy of Error Mitigation Techniques for DRAM Retention Failures: A Comparative Experimental Study," in *SIGMETRICS*, 2014.

[75] S. Khan, D. Lee, and O. Mutlu, "PARBOR: An Efficient System-Level Technique to Detect Data Dependent Failures in DRAM," in *DSN*, 2016.

[76] S. Khan, C. Wilkerson, D. Lee, A. R. Alameldeen, and O. Mutlu, "A Case for Memory Content-Based Detection and Mitigation of Data-Dependent Failures in DRAM," *CAL*, 2016.

[77] S. Khan, C. Wilkerson, Z. Wang, A. R. Alameldeen, D. Lee, and O. Mutlu, "Detecting and Mitigating Data-Dependent DRAM Failures by Exploiting Current Memory Content," in *MICRO*, 2017.

[78] H. S. Kim, M. Kandemir, N. Vijaykrishnan, and M. J. Irwin, "Characterization of Memory Energy Behavior," *WWC*, 2000.

[79] J. Kim, M. Patel, H. Hassan, and O. Mutlu, "The DRAM Latency PUF: Quickly Evaluating Physical Unclonable Functions by Exploiting the Latency–Reliability Tradeoff in Modern DRAM Devices," in *HPCA*, 2018.

[80] Y. Kim, "Architectural Techniques to Enhance DRAM Scaling," Ph.D. dissertation, Carnegie Mellon Univ., 2015.

[81] Y. Kim, M. Papamichael, O. Mutlu, and M. Harchol-Balter, "Thread Cluster Memory Scheduling: Exploiting Differences in Memory Access Behavior," in *MICRO*, 2010.

[82] Y. Kim, V. Seshadri, D. Lee, J. Liu, and O. Mutlu, "A Case for Exploiting Subarray Level Parallelism (SALP) in DRAM," *ISCA*, 2012.

[83] Y. Kim, W. Yang, and O. Mutlu, "Ramulator: A Fast and Extensible DRAM Simulator," *CAL*, 2015.

[84] Y. Kim, R. Daly, J. Kim, C. Fallin, J. H. Lee, D. Lee, C. Wilkerson, K. Lai, and O. Mutlu, "Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors," in *ISCA*, 2014.

[85] A. R. Lebeck, X. Fan, H. Zeng, and C. Ellis, "Power Aware Page Allocation," in *ASPLOS*, 2000.

[86] C. J. Lee, V. Narasiman, O. Mutlu, and Y. N. Patt, "Improving Memory Bank-Level Parallelism in the Presence of Prefetching," in *MICRO*, 2009.

[87] D. Lee, "Reducing DRAM Energy at Low Cost by Exploiting Heterogeneity," Ph.D. dissertation, Carnegie Mellon Univ., 2016.

[88] D. Lee, S. Ghose, G. Pekhimenko, S. Khan, and O. Mutlu, "Simultaneous Multi-Layer Access: Improving 3D-Stacked Memory Bandwidth at Low Cost," *ACM TACO*, 2016.

[89] D. Lee, S. Khan, L. Subramanian, S. Ghose, R. Ausavarungnirun, G. Pekhimenko, V. Seshadri, and O. Mutlu, "Design-Induced Latency Variation in Modern DRAM Chips: Characterization, Analysis, and Latency Reduction Mechanisms," in *SIGMETRICS*, 2017.

[90] D. Lee, Y. Kim, V. Seshadri, J. Liu, L. Subramanian, and O. Mutlu, "Tiered-Latency DRAM: A Low Latency and Low Cost DRAM Architecture," in *HPCA*, 2013.

[91] D. Lee, L. Subramanian, R. Ausavarungnirun, J. Choi, and O. Mutlu, "Decoupled Direct Memory Access: Isolating CPU and IO Traffic by Leveraging a Dual-Data-Port DRAM," in *PACT*, 2015.

[92] D. Lee, Y. Kim, G. Pekhimenko, S. Khan, V. Seshadri, K. Chang, and O. Mutlu, "Adaptive-Latency DRAM: Optimizing DRAM Timing for the Common-Case," in *HPCA*, 2015.

[93] C. Lefurgy, K. Rajamani, F. Rawson, W. Felter, M. Kistler, and T. Keller, "Energy Management for Commercial Servers," *Computer*, 2003.

[94] A.-M. Legendre, *Nouvelles Méthodes pour la Détermination des Orbites des Comètes.* F. Didot, 1805.

[95] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi., "McPAT: An Integrated Power, Area and Timing Modeling Framework for Multicore and Manycore Architectures." in *MICRO*, 2009.

[96] C. H. Lin, D. Y. Shen, Y. J. Chen, C. L. Yang, and M. Wang, "SECRET: Selective Error Correction for Refresh Energy Reduction in DRAMs," in *ICCD*, 2012.

[97] J. Liu, B. Jaiyen, Y. Kim, C. Wilkerson, and O. Mutlu, "An Experimental Study of Data Retention Behavior in Modern DRAM Devices: Implications for Retention Time Profiling Mechanisms," in *ISCA*, 2013.

[98] J. Liu, B. Jaiyen, R. Veras, and O. Mutlu, "RAIDR: Retention-Aware Intelligent DRAM Refresh," in *ISCA*, 2012.

[99] S. Liu, K. Pattabiraman, T. Moscibroda, and B. G. Zorn, "Flikker: Saving DRAM Refresh-Power Through Critical Data Partitioning," in *ASPLOS*, 2011.

[100] C.-K. Luk, R. Cohn, R. Muth, H. Patil, A. Klauser, G. Lowney, S. Wallace, V. J. Reddi, and K. Hazelwood, "Pin: Building Customized Program Analysis Tools with Dynamic Instrumentation," in *PLDI*, 2004.

[101] Y. Luo, S. Ghose, Y. Cai, E. F. Haratsch, and O. Mutlu, "HeatWatch: Improving 3D NAND Flash Memory Device Reliability by Exploiting Self-Recovery and Temperature Awareness," in *HPCA*, 2018.

[102] Y. Luo, S. Ghose, Y. Cai, E. F. Haratsch, and O. Mutlu, "Improving 3D NAND Flash Memory Lifetime by Tolerating Early Retention Loss and Process Variation," in *SIGMETRICS*, 2018.

[103] C. Lyuh and T. Kim, "Memory Access Scheduling and Binding Considering Energy Minimization in Multi-Bank Memory Systems," in *DAC*, 2004.

[104] K. T. Malladi, F. A. Nothaft, K. Periyathambi, B. C. Lee, C. Kozyrakis, and M. Horowitz, "Towards Energy-Proportional Datacenter Memory with Mobile DRAM," in *ISCA*, 2012.

[105] K. T. Malladi, I. Shaeffer, L. Gopalakrishnan, D. Lo, B. C. Lee, and M. Horowitz, "Rethinking DRAM Power Modes for Energy Proportionality," in *MICRO*, 2012.

[106] D. M. Mathew, M. Schultheis, C. C. Rheinländer, C. Sudarshan, C. Weis, N. Wehn, and M. Jung, "An Analysis on Retention Error Behavior and Power Consumption of Recent DDR4 DRAMs," in *DATE*, 2018.

[107] D. M. Mathew, Éder F. Zulian, S. Kannoth, M. Jung, C. Weis, and N. Wehn, "A Bank-Wise DRAM Power Model for System Simulations," *RAPIDO*, 2017.

[108] J. Meza, Q. Wu, S. Kumar, and O. Mutlu, "Revisiting Memory Errors in Large-Scale Production Data Centers: Analysis and Modeling of New Trends from the Field," in *DSN*, 2015.

[109] MFactors, "JET-5467A Product Page," http://www.mfactors.com/jet-5467a-ddr3-sodimm-extender-with-current-sensing/.

[110] Micron Technology, Inc., "DDR3 Point-to-Point Design Support," Technical Note TN-41-13, 2013.

[111] Micron Technology, Inc., "Calculating Memory System Power for DDR3," Technical Note TN-41-01, 2015.

[112] Micron Technology, Inc., "DDR4 Point-to-Point Design Guide," Technical Note TN-40-40, 2018.
[113] J. Mukundan, H. Hunter, K. H. Kim, J. Stuecheli, and J. F. Martinez, "Understanding and Mitigating Refresh Overheads in High-Density DDR4 DRAM Systems," in *ISCA*, 2013.
[114] K. P. Muller, B. Flietner, C. L. Hwang, R. L. Kleinhenz, T. Nakao, R. Ranade, Y. Tsunashima, and T. Mii, "Trench Storage Node Technology for Gigabit DRAM Generations," in *IEDM*, 1996.
[115] O. Mutlu, "The RowHammer Problem and Other Issues We May Face as Memory Becomes Denser," in *DATE*, 2017.
[116] O. Mutlu and T. Moscibroda, "Parallelism-Aware Batch Scheduling: Enhancing Both Performance and Fairness of Shared DRAM Systems," in *ISCA*, 2008.
[117] O. Mutlu, "Memory Scaling: A Systems Architecture Perspective," in *IMW*, 2013.
[118] T. Ohsawa, K. Kai, and K. Murakami, "Optimizing the DRAM Refresh Count for Merged DRAM/Logic LSIs," in *ISLPED*, 1998.
[119] O. Ozturk and M. Kandemir, "Data Replication in Banked DRAMs for Reducing Energy Consumption," in *ISQED*, 2006.
[120] A. Patel, F. Afram, S. Chen, and K. Ghose, "MARSSx86: A Full System Simulator for x86 CPUs," in *DAC*, 2011.
[121] M. Patel, J. Kim, and O. Mutlu, "The Reach Profiler (REAPER): Enabling the Mitigation of DRAM Retention Failures via Profiling at Aggressive Conditions," in *ISCA*, 2017.
[122] I. Paul, W. Huang, M. Arora, and S. Yalamanchili, "Harmonia: Balancing Compute and Memory Power in High-Performance GPUs," in *ISCA*, 2015.
[123] K. Pearson, "Notes on Regression and Inheritance in the Case of Two Parents," *Proc. Royal Soc. London*, 1895.
[124] G. Pekhimenko, E. Bolotin, N. Vijaykumar, O. Mutlu, T. C. Mowry, and S. W. Keckler, "A Case for Toggle-Aware Compression for GPU Systems," in *HPCA*, 2016.
[125] G. Pekhimenko, T. Huberty, R. Cai, O. Mutlu, P. B. Gibbons, M. A. Kozuch, and T. C. Mowry, "Exploiting Compressed Block Size as an Indicator of Future Reuse," in *HPCA*, 2015.
[126] G. Pekhimenko, V. Seshadri, Y. Kim, H. Xin, O. Mutlu, M. A. Kozuch, P. B. Gibbons, and T. C. Mowry, "Linearly Compressed Pages: A Low-Complexity, Low-Latency Main Memory Compression Framework," in *MICRO*, 2013.
[127] G. Pekhimenko, V. Seshadri, O. Mutlu, M. A. Kozuch, P. B. Gibbons, and T. C. Mowry, "Base-Delta-Immediate Compression: Practical Data Compression for On-Chip Caches," in *PACT*, 2012.
[128] M. K. Qureshi, D. H. Kim, S. Khan, P. J. Nair, and O. Mutlu, "AVATAR: A Variable-Retention-Time (VRT) Aware Refresh for DRAM Systems," in *DSN*, 2015.
[129] Rambus, Inc., "RDRAM Memory Architecture," https://www.rambus.com/memory-and-interfaces/rdram-memory-architecture/.
[130] S. Rixner, W. J. Dally, U. J. Kapasi, P. Mattson, and J. D. Owens, "Memory Access Scheduling," in *ISCA*, 2000.
[131] P. Rosenfeld, E. Cooper-Balis, and B. Jacob, "DRAMSim2: A Cycle Accurate Memory System Simulator," *CAL*, 2011.
[132] SAFARI Research Group, "Characterization Results of Modern DRAM Devices Under Reduced-Voltage Operation — GitHub Repository," https://github.com/CMU-SAFARI/DRAM-Voltage-Study.
[133] SAFARI Research Group, "Ramulator: A DRAM Simulator — GitHub Repository," https://github.com/CMU-SAFARI/ramulator.
[134] SAFARI Research Group, "SoftMC — GitHub Repository," https://github.com/CMU-SAFARI/SoftMC.
[135] SAFARI Research Group, "VAMPIRE — GitHub Repository," https://github.com/CMU-SAFARI/VAMPIRE.
[136] B. Schroeder, E. Pinheiro, and W. Webe, "DRAM Errors in the Wild: A Large-Scale Field Study," in *SIGMETRICS*, 2009.
[137] V. Seshadri, D. Lee, T. Mullins, H. Hassan, A. Boroumand, J. Kim, M. A. Kozuch, O. Mutlu, P. B. Gibbons, and T. C. Mowry, "Ambit: In-Memory Accelerator for Bulk Bitwise Operations Using Commodity DRAM Technology," in *MICRO*, 2017.
[138] V. Seshadri, "Simple DRAM and Virtual Memory Abstractions to Enable Highly Efficient Memory Systems," Ph.D. dissertation, Carnegie Mellon University, 2016.
[139] V. Seshadri, Y. Kim, C. Fallin, D. Lee, R. Ausavarungnirun, G. Pekhimenko, Y. Luo, O. Mutlu, P. B. Gibbons, M. A. Kozuch, and T. C. Mowry, "RowClone: Fast and Energy-Efficient In-DRAM Bulk Data Copy and Initialization," in *MICRO*, 2013.
[140] V. Seshadri and O. Mutlu, "Simple Operations in Memory to Reduce Data Movement," in *Advances in Computers, Volume 106*, 2017.
[141] W. Shin, J. Yang, J. Choi, and L.-S. Kim, "NUAT: A Non-Uniform Access Time Memory Controller," in *HPCA*, 2014.
[142] V. Sridharan, N. DeBardeleben, S. Blanchard, K. B. Ferreira, J. Stearley, J. Shalf, and S. Gurumurthi, "Memory Errors in Modern Systems: The Good, the Bad, and the Ugly," in *ASPLOS*, 2015.
[143] V. Sridharan and D. Liberty, "A Study of DRAM Failures in the Field," in *SC*, 2012.
[144] M. R. Stan and W. P. Burleson, "Bus-Invert Coding for Low-Power I/O," *TVLSI*, 1995.
[145] M. R. Stan and W. P. Burleson, "Coding a Terminated Bus for Low Power," in *GLSVLSI*, 1995.
[146] Standard Performance Evaluation Corp., "SPEC CPU2006 Benchmarks," http://www.spec.org/cpu2006.

[147] J. Stuecheli, D. Kaseridis, D. Daly, H. C. Hunter, and L. K. John, "The Virtual Write Queue: Coordinating DRAM and Last-Level Cache Policies," in *ISCA*, 2010.
[148] J. Stuecheli, D. Kaseridis, D. Daly, H. C. Hunter, and L. K. John, "Coordinating DRAM and Last-Level-Cache Policies with the Virtual Write Queue," *IEEE Micro*, 2011.
[149] K. Sudan, N. Chatterjee, D. Nellans, M. Awasthi, R. Balasubramonian, and A. Davis, "Micro-Pages: Increasing DRAM Efficiency with Locality-Aware Data Placement," in *ASPLOS*, 2010.
[150] A. N. Udipi, N. Muralimanohar, and R. Balasubramonian, "Non-Uniform Power Access in Large Caches with Low-Swing Wires," in *HiPC*, 2009.
[151] A. N. Udipi, N. Muralimanohar, N. Chatterjee, R. Balasubramonian, A. Davis, and N. P. Jouppi, "Rethinking DRAM Design and Organization for Energy-Constrained Multi-Cores," in *ISCA*, 2010.
[152] R. Venkatesan, S. Herr, and E. Rotenberg, "Retention-Aware Placement in DRAM (RAPID): Software Methods for Quasi-Non-Volatile DRAM," in *HPCA*, 2006.
[153] N. Vijaykumar, G. Pekhimenko, A. Jog, A. Bhowmick, R. Ausavarungnirun, C. Das, M. T. Kandemir, T. C. Mowry, and O. Mutlu, "A Case for Core-Assisted Bottleneck Acceleration in GPUs: Enabling Flexible Data Compression with Assist Warps," in *ISCA*, 2015.
[154] T. Vogelsang, "Understanding the Energy Consumption of Dynamic Random Access Memories," in *MICRO*, 2010.
[155] L. Wang, J. Zhan, C. Luo, Y. Zhu, Q. Yang, Y. He, W. Gao, Z. Jia, Y. Shi, S. Zhang, C. Zheng, G. Lu, K. Zhan, X. Li, and B. Qiu, "BigDataBench: A Big Data Benchmark Suite From Internet Services," in *HPCA*, 2014.
[156] F. A. Ware and C. Hampel, "Improving Power and Data Efficiency with Threaded Memory Modules," in *ICCD*, 2006.
[157] M. Ware, K. Rajamani, M. Floyd, B. Brock, J. C. Rubio, F. Rawson, and J. B. Carter, "Architecting for Power Management: The IBM POWER7 Approach," in *HPCA*, 2010.
[158] P. R. Wilson, S. F. Kaplan, and Y. Smaragdakis, "The Case for Compressed Caching in Virtual Memory Systems," in *USENIX ATC*, 1999.
[159] Xilinx, Inc., "Virtex-6 FPGA Family," https://www.xilinx.com/products/silicon-devices/fpga/virtex-6.html.
[160] Xilinx, Inc., "ML605 Hardware User Guide," https://www.xilinx.com/support/documentation/boards_and_kits/ug534.pdf, 2012.
[161] Xilinx, Inc., "MIG 7 Series and Virtex-6 DDR2/DDR3 Solution Center - Design Assistant - Memory Controller Efficiency and Possible Improvements," https://www.xilinx.com/support/answers/36719.html, 2017.
[162] J. Yang, Y. Zhang, and R. Gupta, "Frequent Value Compression in Data Caches," in *MICRO*, 2000.
[163] D. H. Yoon, J. Chang, N. Muralimanohar, and P. Ranganathan, "BOOM: Enabling Mobile Memory Based Low-Power Server DIMMs," in *ISCA*, 2012.
[164] D. H. Yoon, M. K. Jeong, and M. Erez, "Adaptive Granularity Memory Systems: A Tradeoff Between Storage Efficiency and Throughput," in *ISCA*, 2011.
[165] T. Zhang, K. Chen, C. Xu, G. Sun, T. Wang, and Y. Xie, "Half-DRAM: A High-Bandwidth and Low-Power DRAM Architecture from the Rethinking of Fine-Grained Activation," in *ISCA*, 2014.
[166] H. Zheng, J. Lin, Z. Zhang, E. Gorbatov, H. David, and Z. Zhu, "Mini-Rank: Adaptive DRAM Architecture for Improving Memory Power Efficiency," in *MICRO*, 2008.
[167] W. Zuravleff and T. Robinson, "Controller for a Synchronous DRAM That Maximizes Throughput by Allowing Memory Requests and Commands to Be Issued Out of Order," U.S. Patent No. 5,630,096, 1997.

APPENDIX: FULL DRAM ENERGY MODEL PARAMETERS

Our new DRAM power model, VAMPIRE (see Section 9), depends on accurately capturing the current consumed by read and write operations when operations are interleaved across different columns and banks, as we discuss in Section 5.2. In Section 5.3, we introduce a linear model (Equation 2) that captures the energy consumed by each read or write operation, as a function of (1) the number of ones in the cache line, (2) the number of bits that are toggled, and (3) whether the operations are interleaved across different banks and/or columns. Because different types of interleaving make use of different switching circuitry (see Figure 18), we require a separate set of model parameters to use in Equation 2 for each type of operation interleaving. Table 5 provides the model parameters (in mA) for each type of interleaving. All of these parameters are derived using linear least-squares regression on data collected from real DRAM modules.

Table 5. Parameters to quantify current consumption (I_{total}) due to data dependency, for the model $I_{total} = I_{zero} + \Delta I_{one} N_{ones} + \Delta I_{toggle} N_{toggles}$, where N_{ones} is the number of ones in the cache line, $N_{toggles}$ is the number of bits that are toggled, and I_{zero}, ΔI_{one}, and ΔI_{toggle} are the parameters.

No Interleaving (Same Bank & Column)

Vendor	Read			Write		
	I_{zero} (mA)	ΔI_{one} (mA)	ΔI_{toggle} (mA)	I_{zero} (mA)	ΔI_{one} (mA)	ΔI_{toggle} (mA)
A	250.88	0.449	0.0000	489.61	-0.217	0.0000
B	226.69	0.164	0.0000	447.95	-0.191	0.0000
C	222.11	0.134	0.0000	343.41	-0.000	0.0000

Column Interleaving Only (same as Table 2 in Section 5.3)

Vendor	Read			Write		
	I_{zero} (mA)	ΔI_{one} (mA)	ΔI_{toggle} (mA)	I_{zero} (mA)	ΔI_{one} (mA)	ΔI_{toggle} (mA)
A	246.44	0.433	0.0515	531.18	-0.246	0.0461
B	217.42	0.157	0.0947	466.84	-0.215	0.0166
C	234.42	0.154	0.0856	368.29	-0.116	0.0229

Bank Interleaving Only

Vendor	Read			Write		
	I_{zero} (mA)	ΔI_{one} (mA)	ΔI_{toggle} (mA)	I_{zero} (mA)	ΔI_{one} (mA)	ΔI_{toggle} (mA)
A	287.24	0.244	0.0200	534.93	-0.249	0.0225
B	228.14	0.159	0.0364	419.99	-0.179	0.0078
C	289.99	0.034	0.0455	304.33	-0.054	0.0455

Bank and Column Interleaving

Vendor	Read			Write		
	I_{zero} (mA)	ΔI_{one} (mA)	ΔI_{toggle} (mA)	I_{zero} (mA)	ΔI_{one} (mA)	ΔI_{toggle} (mA)
A	277.13	0.267	0.0200	537.58	-0.249	0.0225
B	223.61	0.152	0.0364	420.43	-0.179	0.0078
C	266.51	0.099	0.0090	323.22	-0.072	0.0090

Received February 2018; accepted April 2018