# High-Level Power Estimation and Optimization of DRAMs

Karthik Chandrasekar

# Hoog-Niveau Approximatie en Optimalisatie van DRAM Vermogen

**Karthik Chandrasekar**

# High-Level Power Estimation and Optimization of DRAMs

**Proefschrift**

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof.ir. K.C.A.M. Luyben,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen
op vrijdag 03 oktober 2014 om 15.00 uur

door
Karthik Chandrasekar

Ingenieur
Master of Science in Computer Engineering
geboren te Chennai, India

Dit proefschrift is goedgekeurd door de promotor:

Prof. dr. K.G.W Goossens

Copromotor:

Dr. K.B. Åkesson

The work in this thesis was supported by the COMCAS EU project (CA 501).

*To my parents*

# Acknowledgments

I would like to thank my promoter, Prof. Kees Goossens, for being a great advisor and teacher during my PhD. He gave me invaluable guidance and pivotal feedback throughout and continuously drove me to do better and better.

I also wish to express my gratitude to Dr. Benny Akesson, my PhD supervisor for always appreciating every small idea and work that I proposed and spending incalculable hours evaluating and reviewing them.

I would also like to convey my sincere thanks to Dr. Christian Weis for his always ready to help attitude and unlimited ideas on DRAM-related research. In collaborating with him, not only did I gain significant knowledge of DRAMs but also a good friend. This collaboration would not have been as fruitful, if not for Prof. Norbert Wehn, who constantly encouraged and inspired us to improve our work. I cannot thank him enough for all his help. I would also like to specially thank Mathias Jung for sparing his time and effort in taking this collaboration forward and for inviting me to different social events in Kaiserslautern.

I would also like to communicate my appreciation to Prof. Koen Bertels for always being enthusiastic and willing to help the students in his group. I also want to deeply thank the current and former members of the Memory Team at TU Eindhoven, Sven Goossens, Manil Gomony, Yonghui Li, Tim Kouters and Jasper Kuijsten for all the numerous discussions, meetings, reviews and feedback on various aspects of my PhD work. I would also like to thank members of the CompSoc Team from TU Delft and TU Eindhoven Anca, Andrew, Aqeel, Ashkan, Davit, Jude, Martijn, Pavel and Shube for being great team players and good friends. I would also like to appreciate Lidwina, Marja and Rian for their immeasurable support and assistance throughout my PhD.

I would also like to thank my office mates at Eindhoven (in alphabetical order), Alexandre, Francesco, Hamid, Rosilde, Sebastian and Umar for all the crazy times both at work and outside. I would also like to thank all my friends in Delft and Eindhoven, Gokul, Chocka, Shena, Mahi, Silesh, Arvind, Ram, Sangi, Ganesh, Santi and others who made my stay in the Netherlands fun and memorable.

I wish to acknowledge the role of Prof. Venkateswaran Nagarajan of WARFT, who has been my true Guru and mentor and motivated me to take up a PhD.

I am very grateful to my parents and my sister who have been the source of my strength throughout my PhD with their constant encouragement and love.

# Summary

Embedded systems have become an integral part of our life in the last few years in multifarious ways, be it in mobile phones, portable audio players, smart watches or even cars. Most embedded systems fall under the category of consumer electronics, such as televisions, mobile devices, and wearable electronics. With several players competing in this market, manufacturers of embedded systems continue to add more functionality to these devices to make them more user friendly, and often equip them with a very high resolution display and graphics support, and better computing and Internet capabilities. Unfortunately, they are often constrained by tight power/energy budgets, since battery capacity does not improve at the same rate as computing power. While there is clearly much progress to be made in harnessing all the possibilities of embedded systems, limitations in battery capacities, thermal constraints and power/energy budgets surely hinder this progress. Although technology scaling has traditionally addressed both the power minimization and high-performance requirements, with Moore's law nearing its limits, the development of energy-efficient system designs has become critically important. Thus, to be able to continue to provide new and improved features in embedded systems, design-time and run-time power management and minimization holds the key. As a consequence, power optimization has become one of the most defining aspects of designing modern embedded systems.

To design such high-performance and energy-efficient embedded systems, it is extremely important to address two basic issues: (1) accurate estimation of power consumption of all system components during early design stages and (2) deriving power optimization solutions that do not negatively impact system performance.

In this thesis, we aim to address these two issues for one of the most important components in modern embedded systems: DRAM memories. Towards this, we propose a high-precision DRAM power model (DRAMPower) and a set of performance-neutral DRAM power-down strategies.

DRAMPower is a high-level DRAM power model that performs high-precision modeling of the power consumption of different DRAM operations, state transitions and power-saving modes at the cycle-accurate level. To further improve the accuracy of DRAMPower's power/energy estimates, we derive better than worst-case and realistic measures for the JEDEC current metrics instead of vendor-provided worst-case measures from device datasheets.

Towards this, we modify a SPICE-based circuit-level DRAM architecture and power model and derive better than worst-case current measures under nominal operating conditions applicable to a majority of DRAM devices (>97%) with any given configuration (capacity, data width and frequency). Besides these better than worst-case current measures, we also propose a generic post-manufacturing power and performance characterization methodology for DRAMs that can help identify the realistic current estimates and optimized set of timing measures for a given DRAM device, thereby further improving the accuracy of the power and energy estimates for that particular DRAM device.

To optimize DRAM power consumption, we propose a set of performance-neutral DRAM power-down strategies coupled with a power management policy that for any given use-case (access granularity, page policy and memory type) achieves significant power savings without impacting its worst-case performance (bandwidth and latency) guarantees.

We verify the pessimism in DRAM currents and four critical DRAM timing parameters as provided in the datasheets, by experimentally evaluating 48 DDR3 devices of the same configuration. We further derive optimal set of timings using the performance characterization algorithm, at which the DRAM can operate successfully under worst-case run-time conditions, without increasing its energy consumption. We observed up to of 33.3% and 25.9% reduction in DRAM read and write latencies and 17.7% and 15.4% improvement in energy efficiency.

We validate DRAMPower model against a circuit-level DRAM power model and verify it against real power measurements from hardware for different DRAM operations. We observed between 1-8% difference in power estimates, with an average of 97% accuracy. We also evaluated the power-management policy and power-down strategies and observed significant energy savings (close to theoretical optimal) at very marginal average-case performance penalty without impacting any of the original latency and bandwidth guarantees.

# Samenvatting

Embedded systemen zijn de laatste jaren een integraal onderdeel van ons leven geworden; je komt ze tegen op enorm veel verschillende plaatsen, zoals in mobiele telefoons, draagbare muziekspelers, smart watches en zelfs in auto's. De meeste embedded systemen vallen binnen de consumenten elektronica, zoals bijvoorbeeld, televisies, mobiele apparaten, en draagbare elektronica. Verschillende spelers concurreren op deze markt, waardoor fabrikanten van embedded systemen continue steeds meer functionaliteit toevoegen aan deze apparaten om ze gebruiksvriendelijker te maken. Daarnaast worden ze vaak uitgerust met een scherm en ondersteuning voor zeer hoge resolutie graphics, en steeds beter wordende reken- en internetmogelijkheden.

Jammer genoeg worden fabrikanten vaak beperkt door krappe vermogens / energiebudgetten, aangezien de batterijcapaciteit niet zo snel vooruit gaat als het bruikbare rekenvermogen. Hoewel er duidelijk veel voortgang is geboekt in het exploiteren van alle mogelijkheden van gentegreerde system, ondervindt men toch hinder van de beperkingen van de maximale batterijcapaciteit, werkingstemperatuur en vermogens/energiebudgetten. Technologieschaling betekende traditioneel gezien zowel een vermogensminimalisatie als een prestatieverbetering voor veeleisende systemen, maar omdat de wet van Moore tegen zijn limieten aan loopt, wordt de ontwikkeling van energie-efficinte systeemontwerpen van cruciaal belang. De sleutel tot het kunnen blijven verbeteren en uitbreiden van de functionaliteit van embedded systemen, is het beheren en minimaliseren van het opgenomen vermogen, zowel in de ontwerpfase als tijdens de levensduur van het systeem.

Voor het ontwerp van energie-efficinte gentegreerde systemen met hoge prestaties is het zeer belangrijk om twee basisproblemen te adresseren: (1) nauwkeurige approximatie van het opgenomen vermogen van alle systeemcomponenten tijdens de vroege ontwerpfases en (2) vermogensoptimalisatieoplossingen zonder negatieve effecten op de systeemprestaties.

In dit proefschrift adresseren we deze twee problemen voor een van de belangrijkste componenten in moderne embedded systemen: DRAM geheugens. We introduceren een nauwkeurig DRAM vermogensmodel (DRAMPower), en een set prestatie-neutrale DRAM power-down strategien.

DRAMPower is een hoog-niveau DRAM vermogensmodel, dat zeer nauwkeurig het opgenomen vermogen van verschillende DRAM operaties, toestandsovergangen en energiebesparende modi modelleert, op een cycle-nauwkeurige tijdschaal. Om de precisie van DRAMPowers vermogen/energieapproximaties te verbeteren, leiden we realistische beter-dan-worst-case waarden af voor de JEDEC stromen, die de door de fabrikanten aangeleverde worst-case waarden uit de datasheet vervangen.

Om dit te realiseren passen we een SPICE-gebaseerd circuitniveau DRAM architectuur- en vermogensmodel aan, en we leiden hieruit af wat de beter-dan-worst-case stroomwaarden onder nominale omstandigheden zijn. Deze zijn toepasbaar op de meerderheid van de DRAM geheugens (>97%) met een willekeurige configuratie (opslagcapaciteit, databusbreedte, en klokfrequentie).

Naast deze beter-dan-worst-case stroomwaarden introduceren we ook een generieke post-productie vermogens- en prestatiekarakterisatiemethode voor DRAM geheugens die kan helpen bij het identificeren van realistische stroomwaarden en een geoptimaliseerde set van timings voor een specifieke DRAM chip, waardoor de nauwkeurigheid van de vermogens- en energieapproximatie voor deze chip verbeterd wordt.

Om het opgenomen vermogen van DRAM te optimaliseren, stellen we een set met prestatie-neutrale DRAM power-down strategien voor, gekoppeld aan een vermogensmanagementpolicy, die voor iedere gegeven use case (lees- schrijfgranulariteit, page policy en geheugentype) een significante vermogensbesparing oplevert, zonder daarbij af te doen aan de worst-case prestatie-garanties (bandbreedte en latency).

We verifiren dat de gespecificeerde DRAM stromen en vier kritische DRAM timing parameters in de datasheets pessimistisch zijn, door empirisch 48 DDR3 identieke chips te evalueren. Daarnaast leiden we een optimale set timings af met behulp van het prestatiekarakterisatiealgoritme. Bij het gebruik van deze timings functioneert het DRAM nog steeds in worst-case omstandigheden, zonder dat het energieverbruik stijgt. We zien dat een reductie tot respectievelijk 33.3% en 25.9% van de DRAM lees- en schrijflatency mogelijk is, in combinatie met een verbetering van de energie-efficintie van respectievelijk 17.7% en 15.4%.

We verifiren de correcte werking van DRAMPower door een vergelijking met een DRAM model op circuit-niveau, en een vergelijking met echte hardwaremetinging van verschillende DRAM operaties. We observeren een verschil van 1-8% in de vermogensschattingen, met een gemiddelde nauwkeurigheid van 97%. We evalueren ook de vermogensmanagementpolicy en power-down strategien en zien daarbij significante energiebesparingen (dicht bij het theoretische optimum) tegen een zeer marginale average-case presatatieafname, zonder effect op de originele latency- en bandbreedtegaranties.

# Table of contents

# List of Tables

# List of Figures

CHAPTER 1

# INTRODUCTION

Most modern battery-driven consumer electronics, including tablets, mobile phones, and wearable electronics, can be categorized as embedded systems. With each new generation and version of these devices, manufacturers continue to equip them with better computing, visualization and internet capabilities [15, 16], which often hurt their power consumption. Since battery capacities have not improved at the same rate as computing power [49], efficient design-time and run-time power management and minimization techniques are of highest importance in designing embedded systems.

To address these issues, embedded system designers rely on couple of features: (1) accurate power estimation of all system components and (2) efficient power optimization solutions that do not harm performance.

In this thesis, we propose solutions to address both of these two issues for one of the most important components in the system: DRAM memories. Modern embedded systems often include DRAMs [8,9], to optimize system's performance (for instance in display buffers to get better frame rate), but have an adverse effect on its power and energy consumption [2,19,20] (up to 25% increase [14]). To address this, we propose: (1) a high-precision DRAM power model called DRAMPower that uses realistic current measures as inputs and (2) a set of performance-neutral DRAM power-optimization strategies.

Using the DRAM power model, we identify critical DRAM operations and states that contribute significantly to DRAM power consumption and with the performance -neutral power-optimization strategies, we reduce their impact on overall power consumption.

## 1.1 Problem Statement

Although JEDEC [10] and DRAM vendors have continuously improved DRAM architectures [11–13] in terms of bandwidth and power efficiency, the incessantly increasing demand for higher memory performance (bandwidth) and capacity has meant continued significance of DRAMs in overall system power consumption [19], even when idle [20, 21]. With larger and faster DRAMs being incorporated with every new generation of mobile phones and tablets, DRAM power consumption in mobile devices is likely to match that of mobile processors [14]. Figure 1.1 depicts the ratios of power consumption of different components in a generic mobile platform as observed by Siemens and Infineon [14]. As depicted, memories are seen to consume as much power as an application processor.

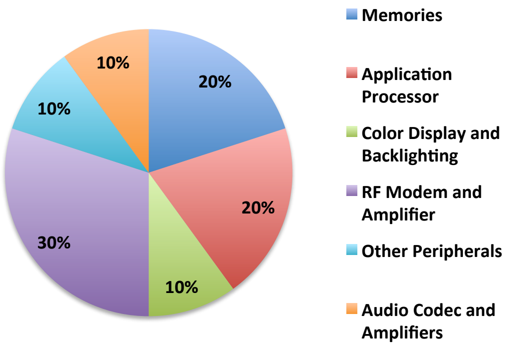


Figure 1.1: Mobile Platform Power Consumption (Adapted from [14])

As expected, DRAMs have become a crucial target for power optimization both in the industry [14, 19, 22–24] and academia [25–37], with solutions ranging from system-level power management down to the circuit-level optimizations, targeting both active and idle power consumption. Refining the earlier statement, the two key factors defining energy-efficient use of DRAMs in embedded systems are: (1) accurate power/energy consumption estimation of DRAMs and (2) efficient power/energy optimization of DRAMs. Together these form the primary focus of this thesis work. Towards this, we propose: (1) a high-precision power model of DRAMs (DRAMPower) and (2) a set of performance-neutral DRAM power optimization strategies.

Before discussing the problems and the proposed solutions in detail, we briefly describe the basics of the DRAM architecture, organization and operations.

### 1.1.1 Background: Generic DRAM Architecture

DRAMs are independent memory devices which for instance, can be used as shared storage between several IPs in a System-on-Chip (SoC) through a DRAM memory controller. DRAMs have a defined architecture, interface, and a set of operating modes. Each of the IPs in an SoC can read or write data into DRAMs by sending read or write requests to the DRAM memory controller, which translates these requests into memory transactions with a set of DRAM commands, data and target memory address. Below, we describe basic DRAM organization, commands and operations. Internally, DRAMs are organized in banks of rows and columns, as shown in Figure 1.2. A bank includes memory elements (cells) arranged in a matrix structure and a row buffer (with sense amplifiers) to store contents of an active memory row. The banks in a DRAM operate in a parallel and pipelined fashion. However, since they all share a single I/O and command bus, only one bank can perform an I/O operation at a particular instance in time and only one DRAM command may be issued to the memory per clock cycle.

To read contents from the memory, an *Activate* command (#1 in Figure 1.2) is first issued by the memory controller (*MC* in the figure) to the DRAM, which opens the requested row and copies data from the DRAM cells in the corresponding row into the row buffer. Then, any number of *Read* or *Write* commands (#3 in Figure 1.2) can be issued to read out or write into specific columns in the row buffer. Subsequently, a *Precharge* command (#2 in Figure 1.2) is issued and the contents of the row buffer are stored back into the corresponding memory row. Reads and writes can also be issued with an *auto-precharge flag* to automatically precharge as soon as the request completes. The number of read/write commands in a transaction is called Burst Count (BC), and the amount of data read out or written into by each command is given by the Burst Length (BL) (e.g. 8 words for DDR3), where each word is defined by the data width of a given device. Furthermore, a memory transaction may also be interleaving over more than one bank, given by the degree of Bank Interleaving (BI). The product of BL, BC and BI parameters determines the data *access granularity* with which the memory controller accesses the memory and has a large impact on both performance and power consumption [97].

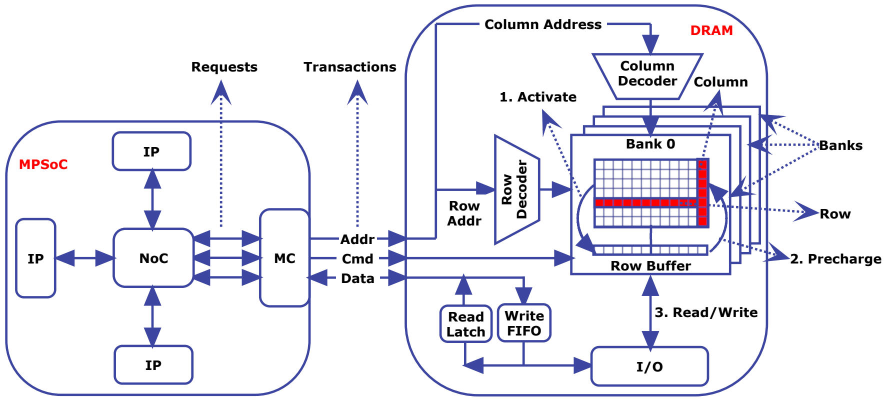


Figure 1.2: DRAM Organization in the System context

If any row is active, the memory is said to be in the *active* state, else it is in the *precharged* state. Switching between a read and a write command, or vice versa, takes a few clock cycles to allow the data bus to switch the I/O direction. Further, to retain data in the memory, all rows in the DRAM need to be refreshed at regular intervals, which is done by issuing a *Refresh* command. Internally, a refresh is a set of activates and precharges to the same row in different banks.

In addition to issuing these commands, it is also possible to transition to power-down state by disabling the clock at run time to reduce power consumption, if the memory is idle. However, the memory must be powered up whenever a refresh command is issued. It is also possible to retain the memory contents without refreshing by employing the Self-Refresh power-saving state, which refreshes the memory at significantly lower power consumption than explicit refreshes.

For proper DRAM operation, the commands discussed above must be issued by the memory controller to the DRAM in a specific order, while satisfying the associated minimal timing constraints (for DDR2 [102] and for DDR3 [103]). For instance, between issuing an *Activate* command and a *Read* command, the minimum timing constraint of *nRCD* cycles should be respected. Some of these constraints that need to be satisfied when issuing commands to a DDR3-800 memory [99] are specified in Table 1.1:

Table 1.1: Micron DDR3-800 Timing Constraints

| *Constraint* | *Description (Minimum Time between)* | *Time (cycles)* |
|---|---|---|
| *nRC* | Two ACTs to the same bank | 20 |
| *nRAS* | An ACT and a PRE to the same bank | 15 |
| *nRCD* | An ACT and a RD/WR to the same bank | 5 |
| *nRP* | A PRE and an ACT to the same bank | 5 |
| *nWTR* | A RD and a WR to the same bank and row | 4 |
| *nRTP* | A RD and a PRE to the same bank | 4 |
| *nCCD* | Two consecutive RDs or WRs | 4 |
| *nRRD* | Two ACTs to different banks | 4 |
| *nCL* | Two RDs to the same bank | 5 |
| *nWR* | A RD and a WR to the same bank and row | 6 |
| *nFAW* | A RD and a WR to the same bank and row | 16 |
| *nRFC* | A REF and an ACT | 44 |

These timing constraints specified by the datasheets are the minimal timings between two commands. However, most DRAM controllers do not always issue commands as soon as these minimal constraints are satisfied. Instead, they schedule commands based on different command-scheduling and row-buffer management policies, where the actual duration between any two issued commands may be greater than the minimum. For instance, the memory controller may employ an open-page policy [50] and delay issuing a precharge to a bank until there is a row-miss on the subsequent access to that bank.

In general, memory controllers employ the open-page policy or the close-page policy [50] based on the assumed presence or absence of data locality in the target application. The former policy keeps the row buffer active to reduce the access time for subsequent accesses to the same memory row in the same bank by not issuing a *Precharge* command at the end of a transaction. The latter policy immediately closes the active row buffer at the end of every bank access with a *Precharge* command, for faster accesses to any other location in the memory in the subsequent transaction.

### 1.1.2 Problem I: DRAM Power Modeling

This section addresses the first of the two problems to be addressed in this thesis: accurate power/energy estimation of DRAMs.

To enable efficient power management, system designers rely on power consumption information provided by DRAM vendors and/or power models developed by DRAM vendors. These power measures/models are required to address three important issues: (1) to design efficient power supplies for DRAMs, (2) to estimate power/energy usage by the DRAMs used in a system and (3) to derive design-time and run-time power optimization policies to reduce DRAM power consumption. The reason for employing these measures/models to address these three issues is their ***accuracy***. *In this work, the accuracy of a given power model is evaluated by comparing its power/energy estimates against real power measurements from a given DRAM device for different DRAM operations.*

JEDEC requires all DRAM vendors to furnish a set of standardized current measures in DRAM device datasheets corresponding to different combinations of standard memory transactions, to obtain approximate power consumption estimates. Although these current measures are adequate to enable designing of suitable power supplies for DRAMs (Issue 1), they are insufficient for accurate power/energy consumption estimation (Issue 2) and efficient DRAM power management (Issue 3), since they do not represent power consumption of individual DRAM operations.

To resolve this issue, DRAM vendors including Micron and Samsung supplement these datasheet current measures with high-level power models [17,18] that include equations to break the datasheet current measures down to measures corresponding to individual DRAM transactions and to obtain a more fine-grained account of power consumption in DRAMs. Although these models provide better details of DRAM power consumption compared to bare datasheet current measures, they have been shown to be ***imprecise in their modeling*** of the different individual DRAM operations, state transitions and power-saving modes [38–40]. This highlights ***the need for high precision modeling***.

Besides the issue of precision in power modeling, the input datasheet measures used by these power models reflect ***worst-case current measures for all DRAM devices*** manufactured with the same configuration (frequency, speed-bin and revision) by a vendor [45, 98–100]. These measures include substantial margins in current measures to address the impact of design-time process-variations and run-time variations in operating temperature and power-supply noise [89]. As a result, the datasheet current measures can differ significantly from the actual observed current measures, when measured on any given DRAM device. *Hence, using datasheet current measures reduces the accuracy of their power and energy estimates for a given DRAM device.* These high-level models employ basic algebraic functions to model power consumption, hence, the worse the accuracy of the input, the worse the accuracy of the output.

To highlight the significant difference between the worst-case datasheet current measures and the nominal current measures of the manufactured lot, we present the average data ($\mu$) and the standard deviation ($\sigma$) in Table 1.2 that shows the impact of process variations on a few DRAM currents [103] as observed by a memory vendor in the production analysis data of a lot of 11,000 DDR3 1Gb memories with 533 MHz frequency and x8 width, manufactured at 70nm.

Table 1.2: Distribution of Current Consumption

| *Current Type* | *Nominal (Average)* $\mu$ *(mA)* | $\sigma$ *%* | *Datasheet (Worst Case)* $\mu + 5\sigma$ *(mA)* |
|---|---|---|---|
| $I_{DD0}$ | 79.1 | 1.4 | 84.7 |
| $I_{DD1}$ | 111.1 | 1.2 | 117.8 |
| $I_{DD2P}$ | 13.1 | 7.1 | 17.7 |
| $I_{DD6}$ | 9.2 | 12 | 14.8 |

This distribution data is represented as probability density function of the different currents in Figure 1.3. In the figure, the reported datasheet (DS) measures are indicated for each current measure. These datasheet measures reflect worst-case case measures for the all devices that are sold from a particular DRAM generation and revision. Such worst-case measures are used to improve the yield of the manufactured devices. The devices that have worse current measures than the datasheet measures are generally rejected from the lot. These correspond to the device in the $+6\sigma$ range.

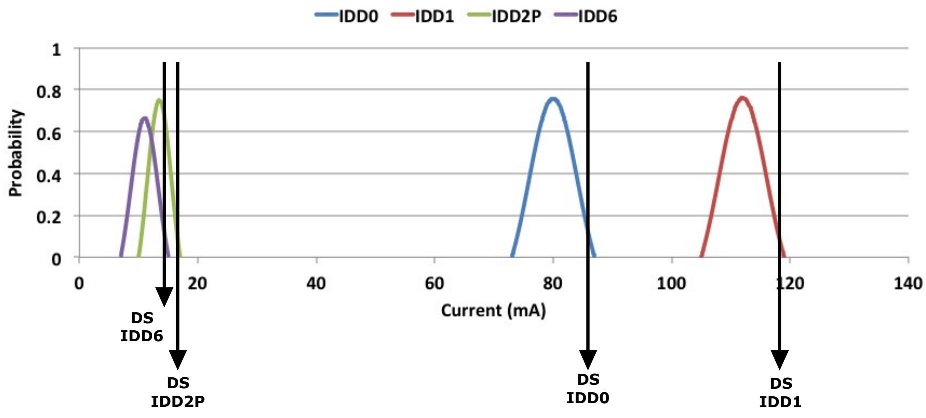


Figure 1.3: Probability Density Function of Current Measures

This current distribution data shows very large difference between the datasheet (DS) current measures and the nominal ($\mu$) current values (by a factor of $5\sigma$), up to 36% and 60% for the low-power modes (power-down: $I_{DD2P}$ and self-refresh: $I_{DD6}$) and up to 7% for the activate-precharge ($I_{DD1}$) current and 6% for the activate-read-precharge ($I_{DD1}$) current [103]. With DRAM memories now being manufactured at technologies below 50nm, these current variations are only expected to worsen, and so is the accuracy of the power models employing these datasheet measures.

To address this issue of worst-case current measures, DRAM vendor RAMBUS [22], Hewlett Packard Research Labs [101], and academic contributors such Weis et al. [46, 47], Keeth et al. [48], have proposed employing detailed circuit-level models of DRAMs to obtain nominal power and energy consumption estimates. By employing such circuit-level models, not only is it possible to (1) model DRAM operations and state transitions more precisely than the high-level current measures-based models, but also to (2) derive more accurate power estimates by using nominal (average-case) current measures in place of the worst-case datasheet measures.

However, there are a few issues in employing these circuit-level models. Firstly, the underlying DRAM architectures employed by these models may not accurately reflect the design choices and optimizations across different DRAM vendors. Furthermore, to make sure the required modifications reflect architectural differences, one needs to have a detailed understanding of the circuit-level behavior of DRAMs, making it very inconvenient for system designers to employ these models. To add to this issue, DRAM vendors do not provide extensive circuit-level details of their DRAM architectures and designs, which makes it extremely difficult to adapt these models to reflect real designs. Finally, although the nominal current measures may be closer to real measurements (for most devices) than the worst-case measures, they only serve as approximate indicative measurements and can still differ from the actual current measures of a given DRAM device by a large extent. Also, using nominal measures only covers 50% of the DRAM devices in the lot.

Considering the difficulties in employing and adapting the circuit-level models, the only feasible alternative is to ***employ current measures-based power models*** similar to those by Micron and Samsung, *but* ***with*** **improved precision in their modeling** of the different DRAM transactions, along with ***realistic or 'better than worst-case' current values*** for a device configuration, instead of the worst-case measures.

### 1.1.3 Problem II: Run-Time DRAM Power Optimization

This section addresses the second of the two problems to be addressed in this thesis: performance-neutral power optimization of DRAMs.

Most modern embedded systems employ DRAMs as a high-bandwidth low-cost memory solution to store active application code and data to enhance system performance. However, DRAMs also significantly impact system power and energy consumption (increase of up to 25% in mobile phones) [14, 19], even when they are idle [20, 21] and are prime candidates for efficient run-time power management and optimization to reduce their energy consumption.

Figure 1.4 depicts DRAM energy consumption components when the DRAM is idle 50% of the time and switching between reads and writes (using a closed-page policy) interspersed with the occasional refresh, for 1Gb DDR3-800 modules from Micron. As can be noticed from the pie chart, idle energy (energy consumed in the precharged idle standby state) contributes to more than 25% of the total energy consumption at 50% idleness, highlighting the need for optimization of power consumption during idle periods.

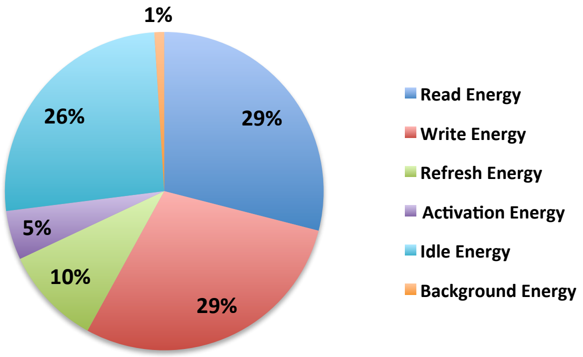


Figure 1.4: Energy Consumption of DRAM Operations

Besides the design-time circuit-level and architectural optimizations for power and energy-efficient DRAM designs employed by JEDEC and DRAM vendors, run-time high-level power optimization solutions are required for efficient system designs. Towards this, innumerable propositions have been made [14, 19, 22–37] to optimize DRAM power consumption, obtaining different degrees of power reduction, often with a corresponding negative impact on performance. With the ever-increasing demand for higher memory bandwidth, employing power optimization strategies that trade off performance for power becomes counter-productive. To optimize DRAM power consumption, two primary avenues are explored: (1) reducing active power by optimizing power consumption of DRAM accesses and refreshes and, (2) reducing standby power by optimizing power consumption when the memory is idle.

To reduce active power consumption, general approaches target: (1) minimizing row-buffer misses and (2) reducing read-write switches. The main goal of these approaches is to improve the average DRAM performance by reducing the overall number of DRAM operations and as a byproduct, reduce the DRAM energy consumption. However, these optimizations can also impact worst-case latencies of individual DRAM transactions, due to: (1) inefficient handling of open rows and (2) re-ordering of transactions.

To reduce standby power consumption, DRAMs have the option of using either (1) power-down or (2) self-refresh modes to power-off the device [103] or (3) frequency scaling to minimize idleness. Unfortunately, the powering-off mechanisms, if speculatively used, may impact performance due to their power-up latencies [103]. Also frequency scaling may impact the latency of individual DRAM operations, due to the slowing down of the memory.

This calls for DRAM power optimization strategies that can efficiently employ any of these approaches, while avoiding or hiding any resulting performance loss. Such guaranteed performance is often required by applications with strict performance requirements (such as high-performance real-time systems), which demand worst-case guarantees from every component in the embedded system and cannot tolerate any impact on the same.

## 1.2 Proposed Solutions

To address both the accurate DRAM power modeling and performance-neutral DRAM power optimization problems, we propose the following:

1. DRAM power modeling - There are two issues to address here: (a) Improved precision in power modeling and (b) Employing *realistic* or *better than worst-case* current values as inputs.

   To address the first issue, we propose high-level cycle-accurate high-precision power model of DRAMs. To address the second issue, we propose to adapt a circuit-level SPICE model to reflect the architecture of a particular DRAM configuration and:

   I. Derive *better than worst-case current measures* that are applicable for a majority of DRAM devices in a particular generation ($\geq$97%), in place of datasheet measures, which are extremely pessimistic. These 'better than worst-case' measures are obtained by introducing device level variations in the circuit-level DRAM model and performing Monte-Carlo analysis to derive $\pm 6\sigma$ distribution of current measures (that reflect impact of process variations). From this distribution, we select the current measures applicable to $\geq$97% of the devices ($+3\sigma$ data point).

   II. Combine it with a post-manufacturing DRAM power and performance characterization methodology to determine more *realistic* current measures for a particular given DRAM device, which may lie anywhere in the $\pm 6\sigma$ distribution of current measures.

2. DRAM power optimization - We propose a couple of performance-neutral DRAM power-down strategies with a run-time power management policy that reduce memory power consumption, while preserving the original worst-case performance guarantees.

### 1.2.1 Improved DRAM Power Modeling and Estimation

When it comes to DRAM power modeling and estimation, the accuracy of power estimation can improve with: (1) An increase in the level of detail (precision) employed in modeling the power consumption of a particular DRAM operation (like activation, precharge, refresh, power-down etc.) and (2) A higher degree of accuracy of the inputs employed by the models (such as its current measures, design specifications etc.).

In this work, we establish the accuracy of a given power model by comparing its power/energy estimates against real power measurements from a given DRAM device, thereby evaluating both the power model's detail of modeling different DRAM operations/transactions and its assumptions on the DRAM current measures it uses as inputs.

In this context, we first establish a generic metric to evaluate the accuracy of a DRAM power model. Let $\mathrm{M}_{(i,j)}$ denote a Power Model *i* and its modeling of operation *j* and $\mathrm{I}_{(i,j)}$ denote its current inputs for the particular operation (e.g. worst-case or nominal currents). The resultant output can be derived as $\mathrm{O}_{(i,j)}$, as shown in Equation (1.1). Accordingly, the power model's accuracy for the particular operation $\mathrm{A}_{(i,j)}$ can be given by Equation (1.2), where $\mathrm{O}_{(ref,j)}$ gives the reference power consumption estimate obtained from direct power measurements on a particular DRAM device for the particular operation *j*. The aggregate average of this accuracy measure over all DRAM operations *(J)*, indicates the overall accuracy of the power model as given by Equation (1.3), where $J$ is the set of all important DRAM operations. In general, since worst-case currents represent the entire lot of DRAM devices of a particular generation and configuration and nominal currents represent 50% of the devices of the lot, the accuracy of models using these current measures is likely to be higher for the proportion of population they represent.

$$O_{(i,j)} = M_{(i,j)}\big(I_{(i,j)}\big) \tag{1.1}$$

$$A_{(i,j)} = 1- \mid 1 - (O_{(i,j)}/O_{(ref,j)}) \mid \tag{1.2}$$

$$A_{(i)} = \big(\sum_{j \epsilon J} A_{(i,j)}\big)/ \mid J \mid \tag{1.3}$$

As stated before, the state-of-the-art transaction-level models based on current measures such as those from Micron, are imprecise in their modeling of different DRAM operations and state transitions and employ worst-case current measures as inputs, whereas the circuit-level power models [22, 47, 48, 101] may not accurately reflect the architectural distinctions between different DRAM generations, across DRAM vendors and their design optimizations. As a result, both these models are expected to fair poorly in their accuracy metric.

Considering the issues with employing circuit-level models, the only feasible alternative is to employ high-level power models similar to those by Micron and Samsung, which are based on JEDEC-specified currents that reflect individual vendor's architectural differences, but with (1) improved precision in modeling of the different DRAM operations ($M_{(i,j)}$) and (2) use of 'better than worst-case' and 'realistic' current measures ($I_{(i,j)}$) instead of the worst-case datasheet current measures, to obtain more accurate power consumption estimates ($O_{(i,j)}$).

To improve the modeling precision, we propose a *high-level power model of DRAMs* referred to as *DRAMPower*, which models the power consumption of different DRAM operations, state transitions and power-saving modes with high precision and analyzes memory command timings at cycle-accurate level, resulting in more accurate power estimates compared to transaction-level models.

To address the issue of assumptions on input currents, we propose to adapt a circuit-level model to reflect the architectural details of a given DRAM configuration and:

(I) Derive *better than worst-case current measures* that are applicable to $>$ 97% of the devices with the particular configuration ($+3\sigma$ of the population).

(II) Combine it with a generic post-manufacturing DRAM power and performance characterization methodology that identifies *realistic* current estimates for any given DRAM device of any configuration.

Both the solutions for power modeling and assumptions on current inputs, improve the accuracy of the power and energy estimates of DRAMPower. We further employ the actual measured current values for a given DRAM device as inputs to DRAMPower, and show a very high degree of accuracy in its power estimates over different DRAM operations, in comparison to real measurements from hardware.

In Table 1.3, we depict the modeling detail and current inputs employed by the state-of-the-art power models, viz., Micron (M0) and Weis et al., (M1), as we improve DRAMPower's (M2) accuracy by introducing high-precision cycle-accurate modeling and use of 'better than worst-case' current measures (I2 [$+3\sigma$]) in place of Datasheet current measures (I0 [$+5\sigma$]) and nominal current measures (I1 [$\mu$]). These comparisons are valid for >97% of the DRAM devices of a given configuration.

Additionally, we also propose a post-manufacturing power and performance characterization mechanism to obtain more *realistic* current measures (I3) for a given DRAM device. These realistic current measures are identified at the peak performance of the given device, and are much closer to the real current measures of the device, if it were to be operated at its peak performance. To fairly assess the accuracy of the high-level power models (Micron and DRAMPower), we also employ *measured current ($I_{DD}$) values* (I4) for a given DRAM device as inputs to both these models (M0/I4 and M2/I4, respectively), and then compare their power estimates over different DRAM operations to real measurements from hardware ($O_{ref}$), thereby performing a fair comparison between the three sets of power estimates. These comparisons are also shown in Table 1.3.

In Figure 1.5, we present an overview of the different levels of detail and current measures employed by different DRAM power models (including DRAMPower) and the *relative degree of accuracy of their power and energy consumption estimates compared to real power measurements from hardware* on a 512MB DDR3-800 DIMM using 1Gb-Micron DDR3-800 devices. As shown in the figure, the accuracy of DRAMPower improves by employing 'better than worst-case' current measures ($I_{DDs}$) [53] in place of the worst-case datasheet current measures,

Table 1.3: Accuracy of State-of-the-art power models vs. DRAMPower

| | Datasheet $I_{DDs}$ (I0) | Nominal $I_{DDs}$ (I1) | Better than Worst-Case $I_{DDs}$ (I2) | Measured $I_{DDs}$ (I4) |
|---|---|---|---|---|
| | (Chapter 2) | (Chapter 3) | (Chapter 3) | (Chapter 4) |
| Trans-Level(M0) | *Micron* [17] | | | *Micron* |
| Circuit-Level(M1) | | *Weis et al.* [46] | | |
| Cycle-Acc.(M2) | DRAMPower | | DRAMPower | DRAMPower |
| Applicable Population % | 100% | 50% | 97% | 1 device |
| Standard Deviation | $+5\sigma$ | $\mu$ | $+3\sigma$ | $-5\sigma$ |

to achieve power estimation accuracy closer to that of the circuit-level models. It further improves the accuracy of its power estimation by employing nominal (I1) and realistic currents (I3) which are closer to the real measures from a given device. Finally, when employing measured currents (I4), it achieves around 97% accuracy compared to power measures from hardware over different DRAM operations. Micron's model also improves its accuracy when employing I4 to achieve around 82% accuracy.

In Figure 1.6, we present an overview of the DRAM power models (M0 to M2) and the current measures they employ (I0 to I4) and their relative degree of accuracy in terms of power estimates compared to real measurements from hardware ($O_{ref}$). As can be noticed in the figure, DRAMPower (M2) using 'better than worst-case' current measures (I2) is shown to be more accurate compared to existing power model/current measure combinations (M0(I0) and M1(I1)) for >97% of the devices. For a particular DRAM device under consideration, DRAMPower using realistic current measures (I3) and measured current values (I4) improves in terms of power estimation accuracy and still evaluates better than the Micron model.

## 1.2.2 Performance-Neutral Power Optimization of DRAMs

DRAM power management mechanisms target two power modes: active and idle, since both are equally important to optimize. Active power management, solutions range from exploiting locality to re-ordering transactions, and reducing refreshes, all of which primarily target minimizing the number of DRAM operations and as a consequence reduce the overall energy consumption. Although these solutions improve performance and reduce power consumption on average, they can also impact worst-case latencies of individual DRAM transactions, since: (1) the memory rows are kept open for long, and (2) re-ordering of transactions can delay individual transactions.

Conservative strategies, such as the one proposed by Goossens et al. in [96],

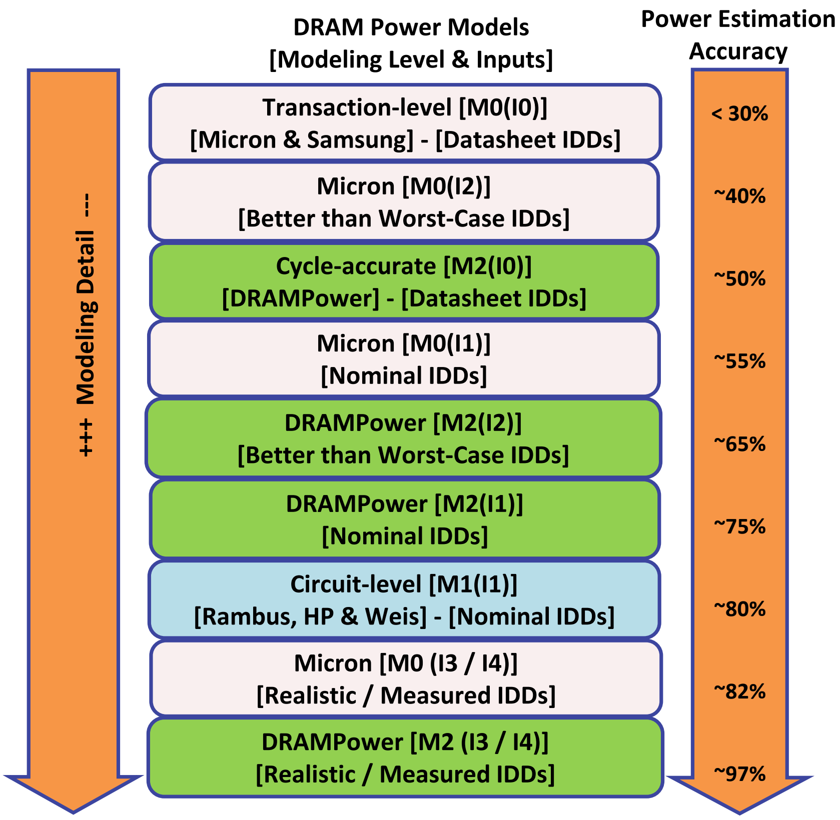

DRAM Power Models
[Modeling Level & Inputs]
Power Estimation Accuracy
+++ Modeling Detail ---
Transaction-level [M0(I0)]
[Micron & Samsung] - [Datasheet IDDs]
< 30%
Micron [M0(I2)]
[Better than Worst-Case IDDs]
~40%
Cycle-accurate [M2(I0)]
[DRAMPower] - [Datasheet IDDs]
~50%
Micron [M0(I1)]
[Nominal IDDs]
~55%
DRAMPower [M2(I2)]
[Better than Worst-Case IDDs]
~65%
DRAMPower [M2(I1)]
[Nominal IDDs]
~75%
Circuit-level [M1(I1)]
[Rambus, HP & Weis] - [Nominal IDDs]
~80%
Micron [M0 (I3 / I4)]
[Realistic / Measured IDDs]
~82%
DRAMPower [M2 (I3 / I4)]
[Realistic / Measured IDDs]
~97%

Figure 1.5: DRAM Power Models vs. Accuracy

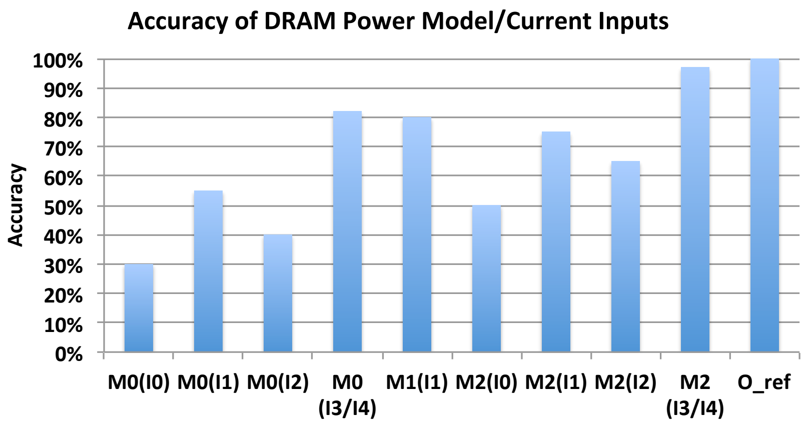


Figure 1.6: Accuracy of DRAM Power Models and Current Measures

can be employed to optimize average-case performance without affecting worst-case guarantees, while reducing active energy consumption. Such worst-case performance guarantees are required by applications with high-performance and real-time performance requirements that must not be violated. Additionally, as a result of reducing overall DRAM accesses, the active power optimization policies tend to increase DRAM idleness and these idle periods must also be optimized for reduced power consumption.

When it comes to idle power management, most solutions employ either the power-down or the self-refresh power saving modes to power off the device when it is idle [24,55–58] or scale down the DRAM frequency to minimize idleness [23,59–61]. The down-side to using these power saving modes is that they can negatively impact both the average-case performance and worst-case latencies of transactions due to their power-up latencies, if speculatively used. Frequency scaling also incurs a performance penalty due to overhead involved in the process and hence, also can affect both average-case and worst-case performance.

In comparison to active power management, idle power optimization poses a bigger challenge, since not only can the speculative use of power saving modes reduce system performance, but can also increase the overall energy consumption. Hence, there is a need to derive idle power optimization strategies that can efficiently employ any of the power saving modes to reduce idle power consumption without affecting the original worst-case performance guarantees, while avoiding or minimizing any impact on average-case performance.

Considering the challenges in reducing idle power consumption in DRAMs, with the aim of deriving performance-neutral run-time power optimization strategies, we propose: (1) *a conservative and an aggressive DRAM power-down strategy* and (2) *a power management policy* for DRAM memory controllers that employs one of these two strategies at run time, *preserving the original worst-case performance guarantees while achieving significant power savings.*

The conservative and aggressive power-down strategies exploit the idle memory service cycles identified by real-time DRAM arbiters like Round-Robin and TDM, to initiate use of the power-down mode and differ primarily in their decision to power-up the memory.

While the conservative strategy acts cautiously and powers up the DRAM by the end of every arbiter service cycle (time period required by the DRAM to serve a request), the aggressive strategy actively merges contiguous idle service cycles to keep the memory in the powered-down state for longer continuous periods, as depicted in Figure 1.7. It does so by snooping the arbiter/bus at the front-end of the memory controller to look ahead for upcoming requests to the memory before deciding to power-up. It must be noted that the aggressive strategy, as a result of the snooping, also manages to power-up the memory in time for the next request to be served without affecting the requester's (memory client) original worst-case memory performance (latency and bandwidth) guarantees.

While the conservative strategy avoids any latency penalties, the aggressive policy efficiently bounds and hides the penalties within the original guaranteed

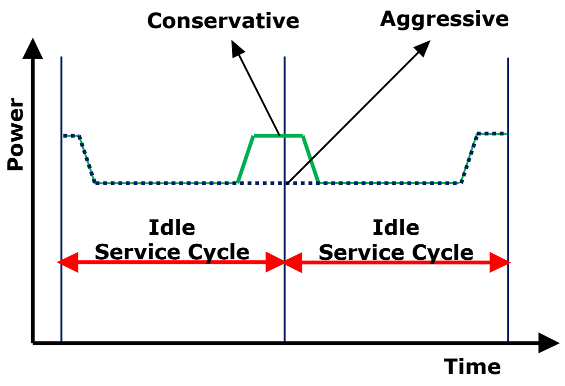


Figure 1.7: Conservative vs. Aggressive Power-Down Strategies

latency, thereby avoiding any impact on the worst-case guaranteed performance. However, the aggressive strategy cannot always be applied for all use-cases and does marginally impact the average-case performance, which the conservative strategy avoids. To assure that both these strategies are used correctly without impacting the worst-case performance guarantees, we also propose a power management policy for the memory controller that evaluates both these power-down strategies for their applicability and potential for energy savings, for a given system use case, based on different memory access parameters such as access granularity, page policy and memory service cycle durations and applicable power-down modes.

This power management policy assures that the power-down strategies do not violate the original DRAM performance guarantees. For instance, for a given use case, if the power-up penalty of the aggressive policy cannot be hidden within the original latency bounds, it chooses to employ conservative power-down, ensuring no violation of the original performance guarantees. Thus, together, both the power management policy and the two power-down strategies are worst-case performance-neutral.

Both the power-down strategies and the power management policy can be employed together with any of the real-time memory controllers presented in [62–67]. Hence, by employing the proposed performance-neutral power-down strategies with the run-time power management policy, the memory controller can effectively and efficiently power-down the DRAM memory when it is idle, without impacting the original DRAM worst-case performance guarantees.

## 1.3 Contributions

As highlighted in the previous sections, the goal of this thesis work is two-fold. (1) To derive accurate DRAM power and energy consumption estimates. (2) To derive efficient DRAM power optimization solutions without trading off worst-case performance for lower power consumption.

Towards this, we propose two major solutions: (1) A high-level cycle-accurate high precision DRAM power model that uses better than worst-case or realistic current measures to achieve accurate power and energy consumption estimates. (2) A run-time power management policy and two DRAM power-down strategies to optimize DRAM idle power consumption without affecting its worst-case performance guarantees. There are five significant contributions of this thesis:

1. **High-Precision DRAM Power Modeling**

   We propose a high-level cycle-accurate DRAM power model (DRAMPower) that enables high-precision power consumption modeling of different DRAM operations, state-transitions and power-saving modes. Towards this, we employ cycle-level DRAM command information, analyze the actual timings between the commands and accurately account for the power consumed during memory state-transitions (Chapter 2). We identify the differences in our modeling approach to existing power models based on current measures.

2. **Variation-Aware DRAM Power Estimation**

   To further improve the accuracy of DRAMPower's power/energy estimates, we derive better than worst-case measures for the JEDEC current metrics instead of vendor-provided worst-case measures from device datasheets. Towards this, we modify an NGSPICE-based circuit-level DRAM architecture and power model to accommodate the effects of design-time and run-time variations and derive a distribution of current measures (Chapter 3) applicable to all DRAM devices with any given configuration (capacity, data-width and frequency). From these measures we derive *better than worst-case* current estimates applicable to a majority (>97%) of the manufactured devices with that configuration (+3$\sigma$ values in the distribution).

   We then propose a generic post-manufacturing power characterization methodology for DRAMs to derive *realistic* current estimates for a given DRAM device. To do so, we assess a DRAM's actual performance characteristics and identify the equivalent impact on power consumption. When employing this methodology, we empirically determine the actual impact of manufacturing process-variations for a given DRAM device, thereby identifying the excess margins for this device, in the datasheet current measures (Chapter 3). As a consequence of this effort, we also identify the best-case performance metrics for a given DRAM device, enabling its optimized usage, both in terms of performance and power consumption.

3. **Open-Source DRAMPower Tool**

   The DRAMPower model has been released as an open source DRAM power and energy estimation tool at *www.drampower.info* [68] for fast and accurate DRAM power and energy estimation for DDR2/3/4, LPDDR/2/3 and Wide IO DRAM memories based on JEDEC standards.

The tool can be employed at two levels of abstraction: (1) Command-level and (2) Transaction-level. To facilitate use of transaction-level traces, DRAMPower includes an optional DRAM command scheduler (developed by Yonghui Li at TU Eindhoven [132]), which dynamically schedules and logs DRAM commands, corresponding to the incoming memory transactions, as if it was a regular memory controller. It assumes a closed-page policy, employs FCFS scheduling across transactions and uses ASAP scheduling for DRAM commands. The tool supports all basic DRAM memory commands including read, write, refresh, activate and precharge, besides the power-down and self-refresh modes.

4. **Validating DRAMPower**

   We validate the DRAMPower model against power measurements from real hardware (for a DDR3 DIMM) and compare its power estimates against those of Micron's power model [17]. Towards this, we employ measured current values from a DDR3 DIMM as inputs to DRAMPower and Micron's model. We determine these measures by implementing the standardized JEDEC current measurement test loops and measuring voltage drop across a shunt resistor. We then implement several test cases covering different DRAM operations, and state-transitions, and compare the power estimates of DRAMPower against those of the Micron model and the actual measurements from hardware (Chapter 4).

   With these experiments, we highlight the significance of high-precision modeling in Chapter 2 by comparing DRAMPower against Micron's model.

5. **Performance-Neutral DRAM Power Optimization**

   We propose two DRAM power-optimization strategies to power-down the DRAM when it is not in use, while making sure that the worst-case performance guarantees of the DRAM memory are not affected. To do so, we employ a performance-neutral run-time power management policy that ensures that both these strategies are used correctly and efficiently without violating any latency/bandwidth bounds. The power management policy on its part evaluates both the power-down strategies for their applicability and potential for energy savings, based on the selected memory access granularity, memory page policy and memory service cycle durations. The two power-down strategies only differ in their powering-up policy and frequency of powering-up, with the aggressive strategy reducing the number of power-ups to the minimum required number and yet powering-up the memory in time for the next request (Chapter 5).

Together, these five important contributions successfully achieve the goals of this thesis and play a part in energy-efficient usage of DRAMs.

## 1.4 Organization of this Thesis

The rest of thesis is organized as follows:

Chapter 2 describes the details of the proposed DRAM power model (DRAMPower), its modeling differences compared to other power models based on current measures (especially Micron's) and the adaptations made to it to address most DRAM generations from DDR2 to DDR4, LPDDR to LPDDR3 and Wide IO DRAMs. The chapter also briefly discusses the tool-flow, command scheduler and command trace analysis of the open-source DRAMPower tool.

Chapter 3 describes the proposed post-manufacturing DRAM power and performance characterization methodology that identifies the excess margins in DRAM current and performance measures in the datasheets for any given DRAM device. This chapter also describes the modifications made to the baseline NGSPICE model to incorporate impact of process variations on DRAM power and performance, and derives better than worst-case current measures for a majority of DRAM devices in a generation with a given configuration.

Chapter 4 includes the tests and experiments used to verify and validate DRAMPower against real hardware measurements and compares its estimates against those of Micron's model.

Chapter 5 describes the proposed performance-neutral power optimization strategies and the power-management policy that enables optimization of idle DRAM power without impacting the original performance guarantees. The chapter describes the latency and bandwidth guarantees provided by real-time DRAM memory controllers and analyses the impact of the proposed power-down strategies on these measures, highlighting their worst-case performance neutrality.

Chapter 6 describes the conclusions drawn from this work and sheds light on possible future extensions to improve this work, both in terms of power estimation and optimization.

# Chapter 2

# Cycle-Accurate DRAM Power Modeling

DRAM memories contribute significantly to the overall system power and energy consumption and require effective power management for their energy-efficient use. The key prerequisite to their efficient power/energy management is to use accurate DRAM power and energy consumption estimates. Hence, system designers require high-precision power models that accurately estimate power and energy consumption of the different DRAM operations, state transitions and power-saving modes.

All DRAM vendors furnish a set of standard current measures corresponding to different combinations of memory operations specified by JEDEC. These measures are employed by high-level power models, which break them down into measures corresponding to individual DRAM operations. However, existing high-level power models lack precision in their modeling of the different DRAM operations, and hence do not report accurate power measures. Alternatively, circuit-level power models can be employed for power estimation, since they perform accurate modeling of these operations, transitions and modes. However, the underlying DRAM architectures employed by these circuit-level models do not accurately reflect architectural distinctions between different DRAM generations, and vendor-specific designs, and need to be extensively adapted to reflect similar configuration as a particular DRAM device, timing behavior and current consumption.

Hence, this chapter proposes a high-level cycle-accurate power model which employs JEDEC-specified current measures and performs high-precision modeling of DRAM operations to obtain accurate power and energy estimates. We compare and contrast the state of the art in high-level DRAM power models against our proposed power equations, which improve the precision of the modeling of the different DRAM operations, state transitions and power-saving modes. Most of the equations presented in this Chapter have been previously published in our papers at DSD 2011 [40] and DATE 2013 [41].

## 2.1 Related Work

The most popular DRAM power model is provided by Micron [17], which derives power equations for different DRAM operations using the JEDEC-specified datasheet current measures. However, it has been found to be inaccurate or insufficient for several reasons including:

1) It *does not consider* the power consumed during the *state transitions* from an arbitrary DRAM state to the power-down and self-refresh states, reporting optimistic power saving numbers for these modes. This also includes any mandatory precharges required before such power-down or self-refresh states can be employed. Schmidt et al., empirically verified this shortcoming of Micron's power model in [38]. Furthermore, it does not take into account the *power consumed during the pre-refresh clock cycles used to precharge all banks before executing a Refresh*, as a part of Refresh power.

2) It employs the *minimal timing constraints* between successive commands from DRAM datasheets [98], [99] and not the *actual duration* between them as issued by a DRAM controller, which may well be greater than the minimum constraints. Direct scaling of the power estimates obtained from Micron's power model gives pessimistic power consumption values for basic DRAM operations, such as reads and writes.

3) It cannot directly provide power consumption values when an *open-page policy* or a *multi-bank-interleaved memory access policy* [51] or a *multi-rank memory system* is employed. This is because, (a) *it assumes a close-page policy* by default, (b) when multiple banks are activated in parallel, *it employs inaccurate scaling of power consumption* and requires adaptations for proper power and energy estimation and, (c) *it does not address power estimation of multi-rank memory systems.*

Schmidt et al., in [38] and [39] empirically measured the power values from a DRAM and showed that Micron's power model provided approximate and worst-case power consumption numbers and over-estimated the actual savings of the Self-Refresh mode for DRAMs. They also attributed these discrepancies to the fact that Micron's power model does not cover the state transitions to the Self-Refresh or the other modes and verified this using different benchmarks.

These critical issues with Micron's power model impact the accuracy and the validity of the power values reported by it. This chapter addresses all of the aforementioned issues by proposing an improved DRAM power model (DRAMPower) for all DRAMs. As stated in Chapter 1, the precision of the power model using the JEDEC-specified current measures, is one of the factors that define the accuracy of the power estimates. The proposed power model takes into account all possible state transitions from any arbitrary DRAM state to the power-down and self-refresh states. Our generic power model accepts a cycle-accurate DRAM command trace of any length (from a single transaction to an application trace) from any memory controller, supporting both open and close-page policies and any degree of bank-interleaving memory access scheme.

Our proposed DRAMPower model employs the *actual duration* between commands obtained from any such DRAM command trace together with the JEDEC-specified current and voltage values. Other existing DRAM power models such as Rawson [42], Joshi et al. [43] and Ji et al. [44], propose DRAM power modeling similar to Micron, but none of them identified or addressed the state transitions issue and hence, do not provide any improved power estimation numbers. Rawson [42] even simplified Micron's model further with approximate power equations, which were less precise compared to Micron's model. Joshi et al. [43] estimated energy per read/write transaction and assumed all transactions have a fixed timing behavior, ignoring the scaling issue. Ji et al. [44] did not model the memory power-saving states or state transitions, and hence, do not provide better power consumption estimates.

## 2.2 Background on DRAM currents

In this section, we describe the different DRAM currents, when and how they are measured, and the state of the banks and changes to the DRAM settings, when they are measured. These currents are also described in detail in [103]. The measures for these currents for a MICRON 512MB DDR3-800 DIMM are provided in Table 2.1

(1) $I_{DD0}$ **(One Bank Active-Precharge Current)**: Measured across ACT and PRE commands to one bank. Other banks are retained in precharged state.

(2) $I_{DD1}$ **(One Bank Active-Read-Precharge Current)**: Measured across ACT, RD and PRE commands to one bank, while other banks are retained in the precharged state. This measurement is performed twice, targeting two different memory locations and toggling of all data bits.

(3) $I_{DD2N}$ **(Precharge Standby Current)**: Measured when all banks are closed (in the precharged state).

(4) $I_{DD2P0}$ **(Precharge Power-Down Current - Slow-Exit)**: Measured during power-down mode with CKE (Clock Enable) Low and the DLL locked but off, while the external clock is On and all banks are closed (precharged).

(5) $I_{DD2P1}$ **(Precharge Power-Down Current - Fast-Exit)**: Measured during power-down mode with CKE (Clock Enable) Low and the DLL locked and on, while the external clock is On and all banks are closed (precharged).

(6) $I_{DD3N}$ **(Active Standby Current)**: Measured when at least one bank is open (in the active state).

(7) $I_{DD3P}$ **(Active Power-Down Current)**: Measured during power-down mode with CKE (Clock Enable) Low and the DLL locked, while the external clock is On and at least one bank is open (active state).

(8) $I_{DD4R}$ **(Burst Read Current)**: Measured during Read (RD) operation, assuming seamless read data burst with all data bits toggling between bursts and all banks open, with the RD commands cycling through all the banks.

(9) $I_{DD4W}$ **(Burst Write Current)**: Measured during Write (WR) operation, assuming seamless write data burst with all data bits toggling between

bursts and all banks open, with the WR commands cycling through all the banks and the ODT (On Die Termination) stable at HIGH.

(10) $I_{DD5}$ **(Refresh Current)**: Measured during Refresh (REF) operation, with REF commands issued every *nRFC* cycles.

(11) $I_{DD6}$ **(Self Refresh Current)**: Measured during self-refresh mode with CKE Low and the DLL off and reset, while the external clock is Off and all banks are closed (precharged).

(12) $I_{DD1W}$(One Bank Active-Write-Precharge Current): This current is not a JEDEC standard measure, however, its reference measures can be calculated by substituting write current ($I_{DD4W}$) instead of read current ($I_{DD4R}$) in $I_{DD1}$ current and corresponds to activation-write-precharge current.

Table 2.1: DDR3 Current Measures

| *Current* | *Type* | *Measure (mA)* |
|---|---|---|
| $I_{DD0}$ | One Bank Active-Precharge Current | 360 |
| $I_{DD1R}$ | One Bank Active-Read-Precharge Current | 440 |
| $I_{DD1W}$ | One Bank Active-Write-Precharge Current | 410 |
| $I_{DD2N}$ | Precharge Standby Current | 180 |
| $I_{DD2P0}$ | Precharge Power-Down Current - Slow-Exit | 40 |
| $I_{DD2P1}$ | Precharge Power-Down Current - Fast-Exit | 100 |
| $I_{DD3N}$ | Active Standby Current | 200 |
| $I_{DD3P}$ | Active Power-Down Current | 100 |
| $I_{DD4R}$ | Burst Read Current | 840 |
| $I_{DD4W}$ | Burst Write Current | 840 |
| $I_{DD5}$ | Refresh Current | 800 |
| $I_{DD6}$ | Self-Refresh Current | 24 |

## 2.3 Our Approach

In this chapter, we present equations to accurately model power consumption of different DRAM operations and estimate power savings for the different power-down modes and the self-refresh mode. For this, we employ the actual timing durations between successive commands issued by a DRAM controller (obtained using a DRAM command trace), instead of the minimal timing constraints from the datasheets, employed by Micron [17].

We also take into account the power consumed during the *state transitions* from any arbitrary state into the power-down, self-refresh and refresh states. In short, we propose a generic power model that supports any row-buffer management policy (open-page or close-page), any command scheduling policy, any degree of bank parallelism or interleaving and multi-rank DRAM systems.

To identify appropriate current consumption values for the different state transitions to the power-down or self-refresh mode, we observe: (a) the state the memory is in (active/precharged) before entering the power-down/self-refresh mode,

(b) the state it will be in (active/precharged) after powering back up, based on the power-saving mode selected, and (c) the changes to the CKE (Clock Enable) signal. We obtain the timing requirements for those state transitions from the JEDEC specified requirements, identify the duration of the transitions from the trace and accurately calculate their energy consumption. Using this approach, we obtain the power consumption values for any such transition and operation.

To highlight the improvement in the accuracy of modeling of DRAM operations in DRAMPower, in Figure 2.1, we present the difference in the modeling of the self-refresh power-saving mode between Micron's power model (indicated by red line), DRAMPower (indicated by white line) and the actual measures.

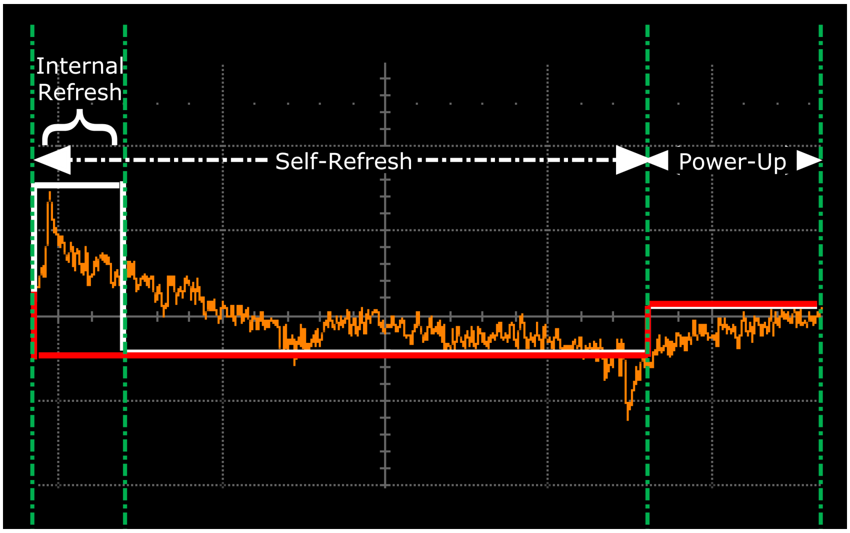


Figure 2.1: Micron (Red) vs. DRAMPower (White) vs. Measurements (Orange)

As can be noticed in the figure, Micron's power model ignores the internal implicit refresh instantiated at the beginning of the self-refresh period, which may prove critical (in terms of power consumption) for shorter self-refresh periods. This effect is captured by DRAMPower unlike Micron's model. Similarly, state transitions to power-down modes or auto-refreshes and use of dynamic command scheduling policies are captured by our model, more accurately.

When it comes to regular transactions, Micron's model employs the minimal timing constraints (Table 1.1) like *nRC* (minimum duration between two Activate commands to the same bank), as the transaction length to calculate power consumption for the transaction (Figure 1.2).

We instead propose to employ the actual *transaction length* denoted by *nTL* for every individual transaction, as observed in the command trace of a DRAM memory controller, to calculate the exact power consumption for that particular transaction. *Note:* In the context of the DRAMPower model, *a read or* *write transaction ends when the data transfer corresponding to the read/write command finishes or the associated auto-precharge (if any) finishes, whichever is later, defining its transaction length.* Similarly, *an idle/power-down transaction length is defined by the duration of the continuous period of idle/power-down clock cycles (including power-up periods).* In implementation, any overlapping cycles between transactions are to be accounted for separately to assure no duplicate calculations.

Additionally, the user may also specify a user-defined window of analysis to derive the power consumption over that time period. Note that however, the minimum transaction length for any operation is defined by the minimum timing constraints associated with that operation. We further clarify on the minimum and actual transaction lengths associated with every operation, as and when we discuss them. In a nutshell, our approach employs the actual observed timings between commands as transaction lengths, addresses DRAM state transitions, and is applicable to all memory controller policies and schedules. We derive our power model (DRAMPower) on the basis of JEDEC specified current and timing measures. In the next sections, we address each of the drawbacks in Micron's model, with improved and new power equations from our model. We first present equations targeting DDR2/3/4 devices and later adapt them to reflect LPDDR/2/3 and Wide IO DRAMs.

## 2.4 Modeling Basic Operations

As stated before, Micron's model employs the *minimal timing constraints* between successive commands from DRAM datasheets [98], [99] and not the *actual duration* between them as issued by a DRAM controller, which may well be greater than the minimum constraints. Micron has identified the basic power components that add up and contribute to overall memory power consumption [17]. These include background power components such as Active Background ($Act_{BG}$) and Precharged Background ($Pre_{BG}$) power, and active power components such as power of Activate ($ACT$), Precharge ($PRE$), Read ($RD$), Write ($WR$) and Refresh ($REF$) commands. Unfortunately, Micron employs the minimal constraints to calculate power consumption of these active power components. As a result, the direct scaling of the power estimates to multiple operations (multiplying) gives pessimistic (higher) power consumption values for basic DRAM operations. Furthermore, Micron's model does not directly provide power consumption values when an *open-page policy* or a *multi-bank or multi-rank memory access* [51] is employed. This is because, *it assumes a close-page policy* by default and suggests equal scaling for both activation and precharge operations even when different timings are used by the memory controller. In reality the ratios of scaling of activation/precharge operations can vary depending on the scheduler and hence, equal scaling can result in higher or lower power estimates. Also, when multiple banks or ranks are accessed in parallel, Micron's power model requires adaptations for proper power and energy estimation. These adaptations are explained in detail later in this section.

In this section, we first present DRAMPower's alternatives (identified by subscript 'D') to Micron's equations (identified by subscript 'M') for these basic power components, considering the actual timings between commands. We then derive equations to represent the different power-saving modes and state transitions. A general rule of thumb is that the background power components (static power) scale up with increase in duration, since they are always consumed whenever the memory is 'ON' (leakage). On the other hand, the active power components (dynamic power) scale down with increase in duration, since they contribute to power consumption only for the period when they are used (based on the switching activity), and get averaged over the actual transaction length ($nTL$). The basic power components that add up for a sample read transaction with burst length 8 (using a close-page policy) are shown in Figure 2.2. The clock cycles in which these power components are consumed are indicated by 'X', for instance, *P(RD)* is consumed over 4 cycles of data transfer.

| *Command* | ACT | NOP | NOP | RD | NOP | NOP | NOP | NOP | PRE | NOP | NOP |
|---|---|---|---|---|---|---|---|---|---|---|---|
| *P($ACT_{BG}$)* | X | X | X | X | X | X | X | X | | | |
| *P($PRE_{BG}$)* | | | | | | | | | X | X | X |
| *P(ACT)* | X | X | X | X | X | X | X | X | | | |
| *P(PRE)* | | | | | | | | | X | X | X |
| *P(RD)* | | | | | X | X | X | X | | | |

Figure 2.2: Basic Power Components in a Read Transaction of length nTL

## 2.4.1 Activate and Precharge Command Power

$I_{DD0}$ is specified as the average current consumed by the memory when it executes an $ACT$ command (to transfer the data from the memory array to the row buffer) and a $PRE$ command (to charge the bit lines and restore the row buffer contents back to the memory array), within the minimum timing constraints. The $I_{DD0}$ current value also includes the active background current $I_{DD3N}$ for the minimum period for which the row is active ($nRAS$) and the precharge current $I_{DD2N}$ for the minimum period for which the row is precharged ($nRP$ = $nRC$ - $nRAS$, where $nRC$ is the total active and precharged time). Hence, these should be subtracted from $I_{DD0}$ for the respective durations and averaged over the transaction length ($nTL$) to identify the average power consumed only due to the $ACT$ and $PRE$ commands. Figure 2.3 depicts the overall current consumption curves, when an ACT command is issued followed by a PRE command within minimum timing constraints (repeated four times), as measured from hardware. These curves only serve as indicators for these operations and the actual measures and scale vary depending on the memory being used.

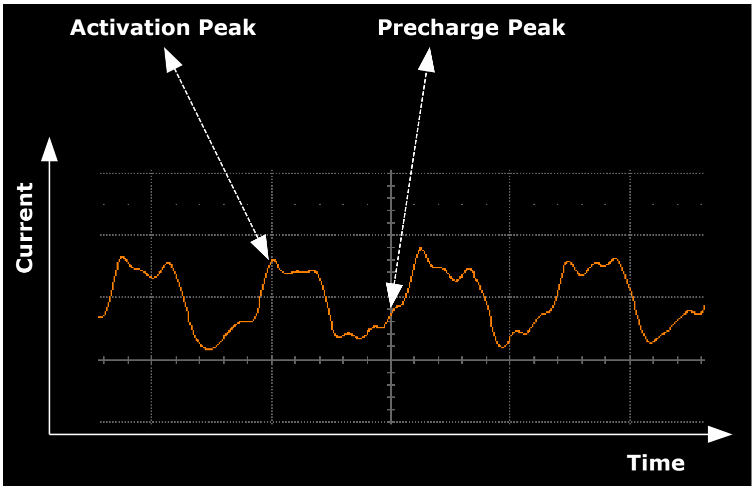


Figure 2.3: ACT-PRE Measurement

The unmodified Micron power model specifies these two power components as one, assuming by default, a close-page policy, as shown in Equation (2.1).

$$P_M(AP) = (I_{DD0} - (\sum_{n=1}^{nRAS} I_{DD3N} + \sum_{n=1}^{nRC\text{-}nRAS} I_{DD2N})/nRC) \times V_{DD} \tag{2.1}$$

However, we split the two power components as $P_D(ACT,\ nTLa)$ and $P_D(PRE,\ nTLp)$ and provide estimates based on the ratio of the number of active cycles to precharge cycles in the transaction, as shown in Equations (2.2) and (2.3), respectively.

This partitioning enables us to provide power estimates when using the open-page policy (where the memory may be retained longer in the active state) and for accurate scaling of appropriate power estimates (corresponding to ACT or PRE sub-operation durations), observed as two distinct peaks in Figure 2.4. Note that in Figure 2.4, each ACT is delayed for a few extra clock cycles after the last PRE to allow the memory to stay idle longer in the precharged mode (to show effect of scaling), whereas in Figure 2.3, minimum timings between the two operations are employed.

$$P_D(ACT,\ nTLa) = \sum_{n=1}^{nRAS} (I_{DD0} - I_{DD3N}) \times V_{DD}/nTLa \tag{2.2}$$

$$P_D(PRE,\ nTLp) = \sum_{n=nRAS+1}^{nRC} (I_{DD0} - I_{DD2N}) \times V_{DD}/nTLp \tag{2.3}$$

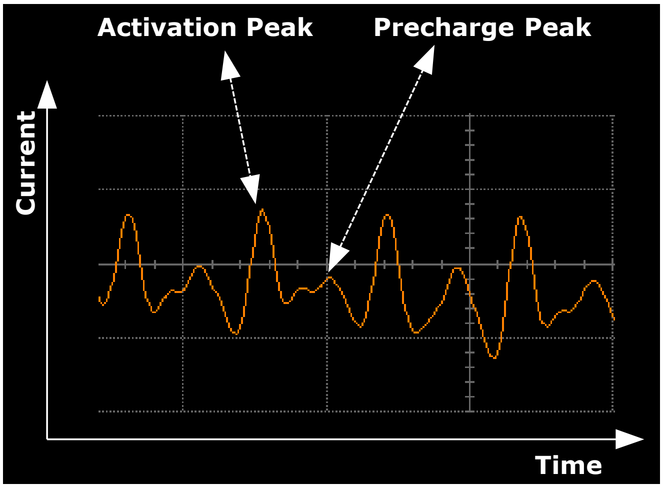


Figure 2.4: ACT-PRE Operation (Spaced Out)

When scaling the power consumption estimates for the ACT and PRE commands, Micron's model scales both activate and precharge power components by the same factor ($nRC_{new}$, scaled sum of active and precharged time), as shown in Equation (2.4), without considering the actual length of time spent individually in the active and precharged states, as shown in Figure 2.5. On the other hand, our equations employ the actual scaling factors for each of the sub-operations as a parameter by performing cycle-accurate analysis to provide better instantaneous power estimates. For instance, $nTLa$ in Equation (2.2) can be $nRAS_{new}(1)$ or $nRAS_{new}(2)$, which are longer than $nRAS$, since activation power is to be computed over the actual activation period.

For the precharge power consumption in Equation (2.3), this period can be defined by the difference between a $nRC_{new}$ and the corresponding $nRAS_{new}$ to indicate the precharge period.

$$P_{M}\_scale(AP) = P_{M}(AP) \times nRC/nRC_{new} \tag{2.4}$$

Furthermore, when scaling this measure when two banks in parallel are activated, as shown in Figure 2.6, Micron's model takes a direct double of the power consumption, as shown in Equation (2.5) for multiple banks. This is incorrect, since it ignores the clock cycles between the two ACT commands, which can range from a minimum of ACT-to-ACT command timing constraint ($nRRD$) to any number of active idle cycles defined by the memory controller leading to a delay in the completion of the second ACT-PRE operation, and resulting in lower average power compared to Micron's estimates. Our equations on the other hand, analyze each of these operations per bank and as a result, avoid any incorrect scaling of the power estimates.

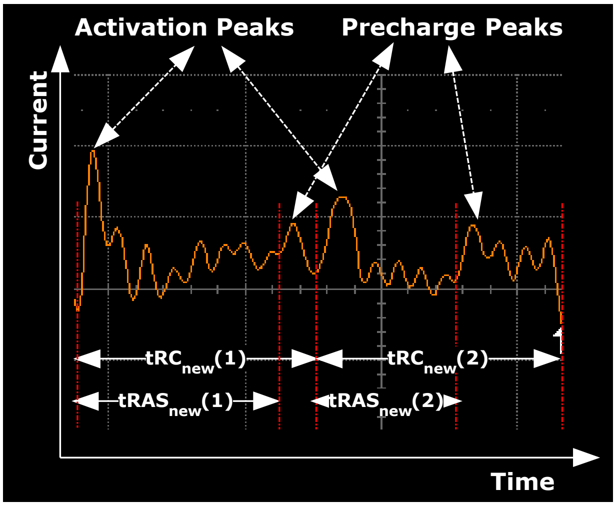


Figure 2.5: ACT-PRE Scaling

$$P_M\_multibank(AP,\ nBanks) = P_M(AP) \times nRC \times nBanks \tag{2.5}$$

Additionally, Micron's model does not provide any power equations to model the Precharge-All (PREA) command, which is often employed when more than one bank has an active row. The PREA command is more efficient in its latency and energy consumption compared to explicit PRE commands to different banks, since it avoids use of explicit PRE commands and takes less time than multiple PRE commands. DRAMPower provides a power equation to evaluate PREA, as shown in Equation (2.6).

Figure 2.7 shows an instance of use of PREA command (LHS in the figure) which precharges 4 banks (at once) compared to using 4 independent PRE commands (RHS in the figure). In this equation as well, the transaction length can be as long as $nRC_{new}$ (including activation and precharge period) or $nRP_{new}$, the actual precharging period.

$$P_D(PREA,\ nTL) = \big(P_D(PRE, nTL) \times n_{open\_banks} + \sum_{n=1}^{nRP} I_{DD2N}\big)/nTL \tag{2.6}$$

## 2.4.2 Read and Write Command Power

A *Read* command consumes $I_{DD4R}$ average current during the cycles of the data transfer, while a *Write* command consumes $I_{DD4W}$. Since these also include the active background current values consumed during the read or the write, $I_{DD3N}$

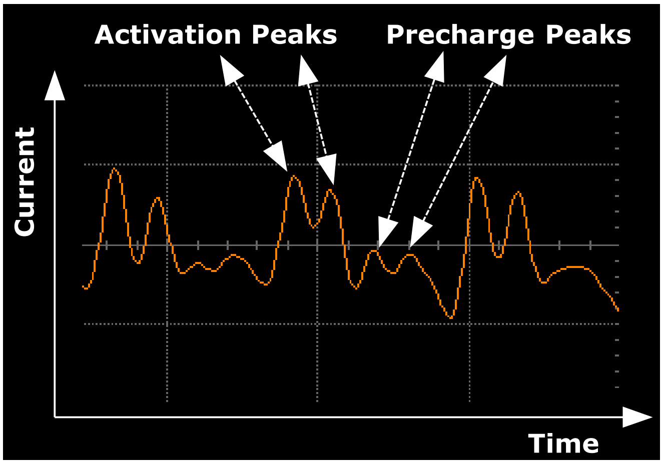


Figure 2.6: Two Bank Activation

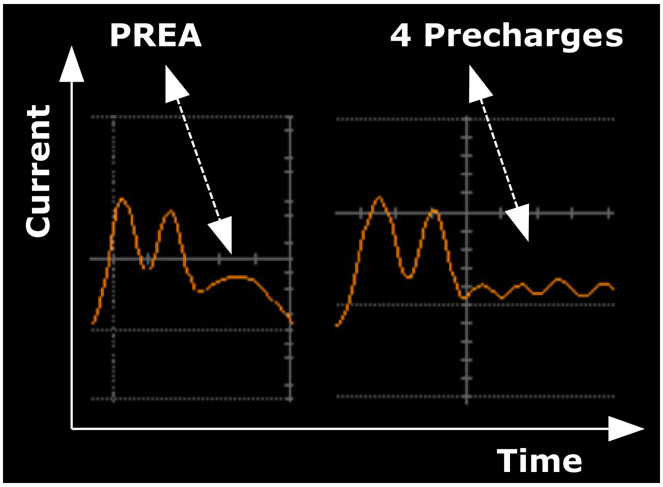


Figure 2.7: PREA clarification

must be subtracted from the $I_{DD4R}$ and $I_{DD4W}$ currents, to identify the power associated only with the *Read* and the *Write* commands, respectively.

To calculate the power associated with the *Read* and *Write* commands, we first sum the current values over the number of cycles the data is on the data bus when reading from or writing to the DRAM, identified here using $nR$ and $nW$, respectively. These cycles of data transfer for a single burst of data can be derived using the ratio of burst length (BL) to data rate (DR). For DDR memories this equates to BL/2. The power values are scaled over the transaction length $nTL$ to get the average power consumed by a *Read* and a *Write*, given by Equations (2.7) and (2.8), respectively.

Micron's model evaluates these operations using minimal timing measures for the $nRC$ parameter as the transaction length, which would not hold if multiple bursts of reads/writes are performed from the same active row in the bank, where the average power consumption would be higher and merely multiplying the read/write power by the number of bursts would not be accurate. Hence, the actual transaction length depending on the number of bursts and the actual activation and precharge period should employed. The power consumption for reads is shown in Figures 2.8. The operations include ACT-READ-PRE sequences (single burst), one after the other to different banks. This is similar for writes.

$$P_D(RD,\ nTL) = \sum_{n=1}^{nR} (I_{DD4R} - I_{DD3N}) \times V_{DD}/nTL \tag{2.7}$$

$$P_D(WR, nTL) = \sum_{n=1}^{nW} (I_{DD4W} - I_{DD3N}) \times V_{DD}/nTL \tag{2.8}$$

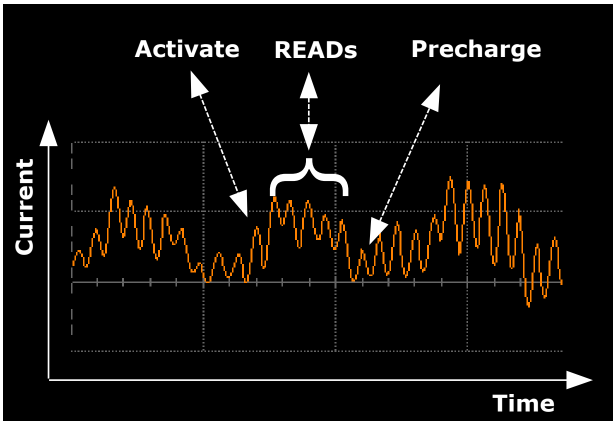


Figure 2.8: READ/WRITE clarification

### 2.4.3 Background, IO and Termination Power

Besides these basic power components, other auxiliary power components are associated with every read and write operation. When a write is issued, the external signal used to drive the data to the memory needs to be terminated on the memory module to avoid distortions of other signals on the memory, using a termination resistor. This termination power is consumed whenever a write is issued and also considers another DRAM rank in the idle mode. Similarly, when a read is issued the power required to drive the data out through the device I/O, must also be accounted for and is referred to as the I/O power.

These power components can be employed directly from Micron's power model. In order to calculate the total power for termination during a write operation, the termination power per data bit, $P_M(W_{DQ})$ from Table 4 in [17], the number of data bits written, $N(W_{DQ})$, and the data strobes $N(W_{DQS})$, must be multiplied. Similarly, to calculate the total power for data I/O during a read operation, the I/O power per data bit, $P_M(R_{DQ})$ from Table 4 in [17], the number of data bits read, $N(R_{DQ})$, and the data strobes $N(R_{DQS})$, must be multiplied. These power measures are computed for DDR3 memories with using the circuit shown in Figure 2.9 and from Table 4 in [17]. In case of single rank DIMMs, I/O and Termination power consumption is likely to be higher, since the appropriate circuitry must be incorporated in the same rank.

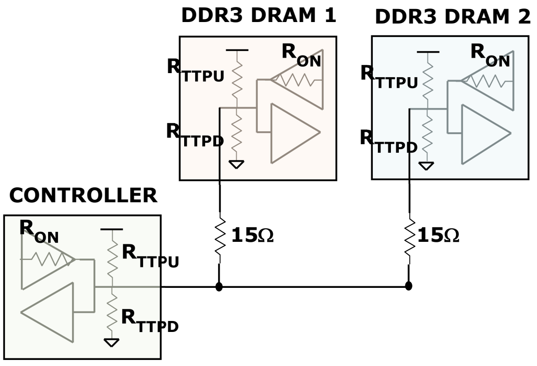


Figure 2.9: I/O and Termination Clarification

Besides these components, the memory consumes certain background power when active. For instance, if all memory banks are in the precharged stand-by state, the memory consumes a precharge background current (static power component) of $I_{DD2N}$ [103], computed as $\mathrm{P}_D(Pre_{BG})$ power. However, even if a single bank is in the active state, the memory consumes an active background current (also static power component) of $I_{DD3N}$, computed as $\mathrm{P}_D(Act_{BG})$ power. These equations are also captured by Micron's model correctly and can be employed as is. To obtain power consumption per clock cycle the memory is in the active or precharged state, the corresponding current can be multiplied with the voltage for those clock cycles.

## 2.5 Modeling State Transitions

As stated earlier, Micron's model *does not consider* the power consumed during the *state transitions* from any arbitrary DRAM state to the power-down and self-refresh states, reporting optimistic power saving numbers for these modes. This also includes any mandatory precharges required before such power-down or self-refresh states can be employed. Schmidt et al., also empirically verified this shortcoming of Micron's power model in [38]. Furthermore, it does not take into account the *power consumed during the pre-refresh clock cycles used to precharge all banks before executing a Refresh*, as a part of Refresh power. In this section, we first look at the equations modeling power-down modes and transitions, then discuss the refresh and self-refresh modes.

### 2.5.1 Transition into Precharge Power-down

Micron's power model does not provide power values for the transition period to power-down modes, resulting in optimistic estimates of power savings. This section corrects this optimism with power equations related to the transition periods from stand-by modes to precharge power-down modes.

When a power-down is issued in the precharge stand-by mode, a time period of *nCPDED* cycles is required to enter the power-down mode. Including this time period, the DRAM will be in power-down mode for *nPD* cycles. Note: nPD is trace dependent and not a JEDEC standard measure. When employing this mode, either a *fast-exit* or a *slow-exit* policy can be selected for DDR3. The fast-exit power-down mode has an exit transition period of *nXP* and the slow-exit power-down mode has an exit transition period of *nXPDLL*, where $nXP \leq nXPDLL$. During *nPD* cycles (power-down time), the memory consumes $I_{DD2P1}$ and $I_{DD2P0}$ currents in the fast-exit mode and slow-exit mode, respectively.

Additionally, to employ the precharge power-down mode, the memory is required to be in the precharged state. To ensure this, a precharge(all) command must be issued, which depending on the number of open banks ($n_{open\_banks}$), can consume significant amount of power and energy over *nRP* cycles (as shown in Figure 2.10), which is not captured by Micron's power model. Equations (2.9) and (2.10), derive the power measures for fast and slow exit precharge power-down modes, respectively. In these equations, the total precharge power-down duration (including transitions and precharging period) is the transaction length ($nTL = nPD + nRP$). These transitions are not accurately accounted for by Micron's power model, which ignores the precharge-all transition step, before entering power-down.

$$P_D(PPD_F, nTL) = \left( P_D(PREA, nRP) + \sum_{n=1}^{nPD} I_{DD2P1} + \sum_{n=1}^{nXP} I_{DD2N} \right) \times V_{DD}/nTL \tag{2.9}$$

$$P_D(PPD_S, nTL) = \left(P_D(PREA, nRP) + \sum_{n=1}^{nPD} I_{DD2P0} + \sum_{n=1}^{nXPDLL} I_{DD2N}\right) \times V_{DD}/nTL \tag{2.10}$$

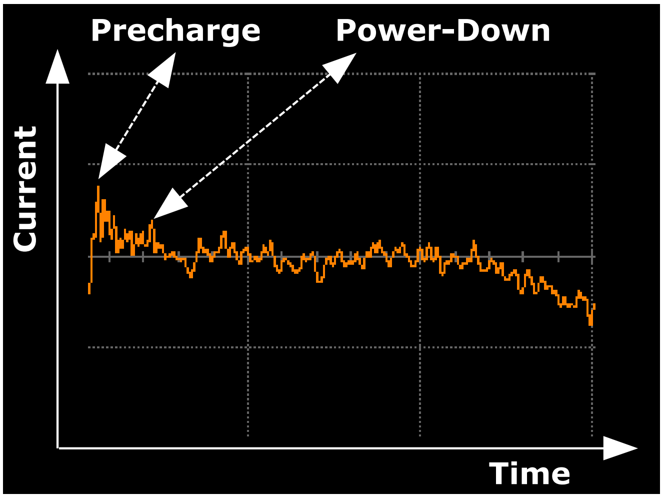


Figure 2.10: Precharge Power-Down Transition

### 2.5.2 Transition into Active Power-down

If a power-down is scheduled after a *Read* or a *Write* (without an auto-precharge), the memory controller must wait at least *nRDPDEN* or *nWRPDEN* cycles, respectively, before entering the active power-down state. During these cycles, active stand-by current of $I_{DD3N}$ is consumed, in addition to the $P_D$*(RD)* and $P_D$*(WR)* power consumed during the *BL/2* cycles of data transfer for read and write, respectively. These are calculated over *nRDPDEN* and *nWRPDEN* cycles, which must be taken as the transaction lengths ($nTL$) of the read and write transitions into power-down. Equations (2.11) and (2.12) give the power for transitioning from a Read and a Write (without auto-precharge) to an active power-down mode. Figure 2.11 depicts this transition from active mode to active power-down. $P_M(R_{DQ})$ and $P_M(W_{DQ})$ measures for I/O power consumption are employed from Table 4 in [17].

$$P_D(R_{PD}, nRDPDEN) = \Big(\sum_{n=1}^{nRDPDEN} I_{DD3N} \times V_{DD} + \sum_{n=1}^{nRDPDEN} P_D(RD,\ nRDPDEN) + \big(P_M(R_{DQ}) \times N(R_{DQ})\big)\Big)/nRDPDEN \quad (2.11)$$

$$P_D(W_{PD}, nWRPDEN) = \Big(\sum_{n=1}^{nWRPDEN} I_{DD3N} \times V_{DD} + \sum_{n=1}^{nWRPDEN} P_D(WR,\ nWRPDEN) + \big(P_M(W_{DQ}) \times N(W_{DQ})\big)\Big)/nWRPDEN \quad (2.12)$$

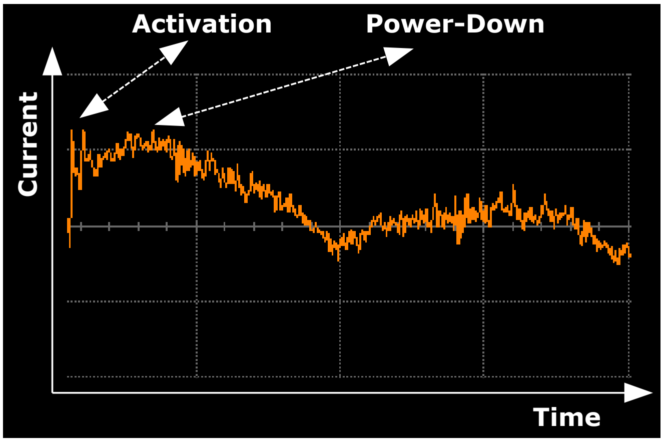


Figure 2.11: Active Power-Down Transition

Once the transition is complete, the DRAM enters the active power-down state. When an active power-down is issued, a time period of *nCPDED* cycles is required to enter the power-down mode. Including this time period, the DRAM will be in power-down mode for *nPD* cycles. When employing this mode, it has an exit transition period of *nXP* cycles. During *nPD* (power-down time), the memory consumes $I_{DD3P}$ current and during *nXP* cycles (power-up time), the memory consumes $I_{DD3N}$ current, as shown in Equation (2.13). The transaction length is defined over the entire transition period, power-down period and power-up period.

$$P_D(APD,\ nTL) = \Big(P_D(R_{PD}, nRDPDEN) + \big(\sum_{n=1}^{nPD} I_{DD3P} + \sum_{n=1}^{nXP} I_{DD3N}\big) \times V_{DD}\Big)/nTL \quad (2.13)$$

If a power-down is scheduled after *Read* is issued with an auto-precharge, the waiting time before entering a precharge power-down mode is also defined by *nRDPDEN* and hence, Equation (2.11) holds for this transition as well. However, if a *Write* is issued with an auto-precharge, the memory controller must wait *nWRAPDEN* cycles before issuing the (precharge) power-down. The active stand-by current of $I_{DD3N}$ is consumed during the *nWRAPDEN-1* cycles before the auto-precharge is issued and $I_{DD2N}$ is consumed for the precharge cycle. In addition, $P_D(WR)$ is consumed during the *BL/2* cycles of data transfer and $P_M(W_{DQ})$ measures for I/O power consumption can be employed from Table 4 in [17]. Also, $P_D(PRE)$ is consumed for the auto-precharge with transaction length *tWRAPDEN*. Equation (2.14) gives the power for a transition from a write with an auto-precharge to a precharge power-down mode. Equations (2.9) and (2.10) can be employed with these transitions after auto-precharge to compute the power-down mode power consumption.

$$P_D(WA_{PD}, nWRAPDEN) = \Bigg( \Big( \sum_{n=1}^{nWRAPDEN\text{-}1} (I_{DD3N}) + I_{DD2N} \Big) \times V_{DD} + P_D(PRE,\ 1) + P_D(WR,\ nWRAPDEN) + \big( P_M(W_{DQ}) \times N(W_{DQ}) \big) \Bigg) / nWRAPDEN \quad (2.14)$$

These transitions (overhead) from read and write or active mode to power-down are also ignored by Micron's power model, when estimating power consumption of the power-down mode.

### 2.5.3 Refresh Transition

A refresh operation is used to retain the data in the DRAM by recharging the capacitors in the memory cells. A refresh can be executed only when all the banks of the memory are in the *precharged state*. A refresh thus consists of a single *Refresh* command preceded by a precharge-all (PREA) command (at least *nRP* cycles before refresh) to precharge all the open banks each before executing the refresh. If all banks are already in the precharged idle state, no explicit precharges are required.

Accordingly, $P_D(PREA,\ nTL)$ is consumed (with a transaction length of *nRP*) depending on the number of precharges required. Micron's model fails to consider the power consumed during pre-refresh clock cycles, as a part of refresh power. The refresh command by itself, consumes $I_{DD5} - I_{DD3N}$ current over the refresh cycles (*nRFC*). Another aspect of Refresh ignored by Micron's model is that during the last *nRP* cycles of the *nRFC* period, the memory is in the precharged mode. During this period it consumes $I_{DD2N}$ current instead of $I_{DD3N}$ (active background current), which is consumed during the rest of the *nRFC* - *nRP* cycles, as shown in Figure 2.12.

The refresh and pre-refresh power components add up over *nREF* ( = *nRP* + *nRFC*) cycles to give the total refresh power, as shown in Equation (2.15).

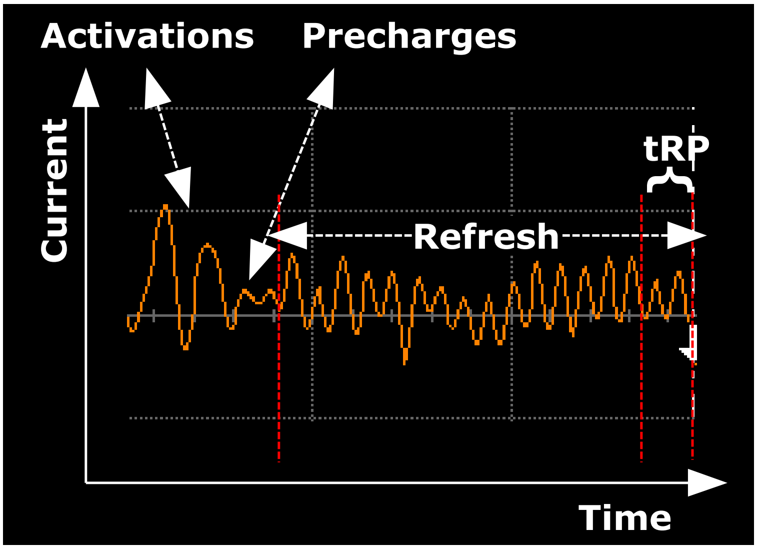


Figure 2.12: Refresh Transition

$$P_D(REF, nREF) = \Big(P_D(PREA, nRP) + \Big(\sum_{n=1}^{nRFC}(I_{DD5} - I_{DD3N}) + \sum_{n=1}^{nRFC\text{-}nRP} I_{DD3N} + \sum_{n=1}^{nRP} I_{DD2N}\Big) \times V_{DD}\Big)/nREF \quad (2.15)$$

### 2.5.4 Self-Refresh Mode Transition

The Self-Refresh mode is used in DRAMs to retain data even when the clock is stopped. In this state, the rest of the memory system is powered down, but the memory internally performs refreshes to maintain its contents without an external clock. In order to switch to the Self-Refresh mode, it must be ensured that the DRAM is idle and all its banks are in the precharge state (using PREA) with *nRP* cycles satisfied to ensure completion of precharging operations.

After issuing the *Self-Refresh* command, the Clock Enable Signal (CKE) must be kept 'LOW' to maintain the memory in the Self-Refresh mode. Additionally, an explicit refresh must be issued at the start of the self-refresh period (Figure 2.13).

The minimum time that the DRAM must remain in Self-Refresh mode is given by *nCKESR*. This includes *nCPDED* to block all the input signals, *nCK-SRE* (Self-Refresh entry time), *nCKSRX* (Self-Refresh exit time) and a minimum *tCKE* period within which the DRAM memory must initiate at least one Refresh command. When exiting the Self-Refresh mode, it must be ensured that the clock is stable and then, the CKE can be changed to 'HIGH'. Once asserted, a timing constraint of at least *nXSDLL* cycles must be satisfied before any other valid command is issued to the memory. The total time required for the Self-Refresh to finish is given by *nSREF* (as derived from the trace).

The $I_{DD6}$ current is consumed for the time period spent in the self-refresh mode as defined in the trace (*nSR*), which excludes the time spent in finishing the explicit auto-refresh (as depicted in Figure 2.13). The auto-refresh consumes $I_{DD5} - I_{DD3N}$ over one refresh period (*nRFC*) from the start of the self-refresh. $I_{DD2N}$ current is consumed when exiting the self-refresh state for the *nXSDLL* exit period.

If the auto-refresh finishes before the self-refresh exit begins, during these auto-refresh cycles (denoted by *nSR_REF*), $I_{DD3P0}$ current is consumed in the background, instead of the $I_{DD6}$ self-refresh current. However, if the self-refresh exit begins before the end of the explicit auto-refresh, the remaining cycles of the auto-refresh operation (denoted by *nEX_REF*) carry forward to the self-refresh exit period. In this case, the $I_{DD3N}$ current is consumed in the background during these remaining *nEX_REF* cycles.

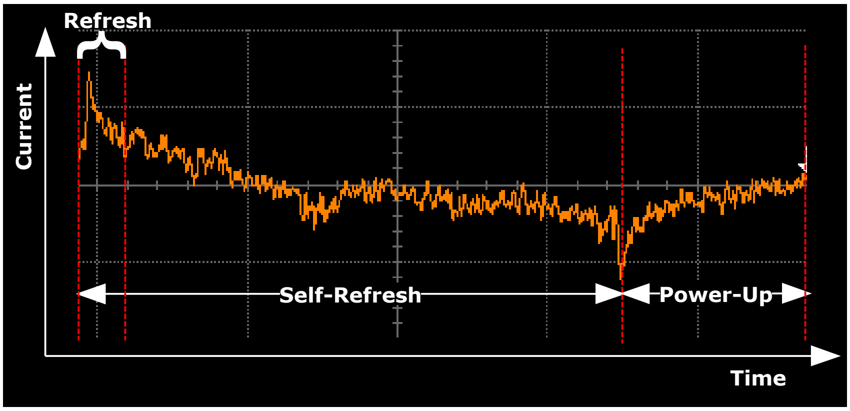


Figure 2.13: Self-Refresh Clarification

Equation (2.16) derives the power consumption of the Self-refresh mode for a transaction length *nTL* equal to the sum of self-refresh period, explicit refresh period and self-refresh exit period. Micron's model merely employs $I_{DD6}$ current during self-refresh and $I_{DD2N}$ current during self-refresh exit.

$$P_D(SR,\ nTL) = \Bigg(\Big(\sum_{n=1}^{nSR} I_{DD6} + \sum_{n=1}^{nXSDLL} I_{DD2N} + \sum_{n=1}^{nRFC} (I_{DD5} - I_{DD3N}) + \sum_{n=1}^{nSR\_REF} I_{DD2P0} + \sum_{n=1}^{nEX\_REF} I_{DD3N}\Big) \times V_{DD}\Bigg)/nTL \tag{2.16}$$

### 2.5.5 Transaction and Trace Power Computation

To estimate power consumption of an entire trace or a transaction, we provide a generic power equation which applies to the whole of the trace. This equation can be employed for any window of analysis (trace length) from a single transaction (including idle transactions) to the entire application trace and is valid for any degree of bank-parallelism and any memory access (open/close page) policy. The initial state of all DRAM banks is assumed to be precharged and ready for operation. This is a safe assumption, since at the end of DRAM initialization, all DRAM banks are precharged.

Before deriving the equation, we list out all the transaction lengths defined in the last sections in Table 2.2 for quick reference.

Table 2.2: Transaction Lengths

| *Transaction* | *Transaction Length* | *nTL Definition* |
|---|---|---|
| Activation | Active Period of a Trans. | $nRAS_{new}$ |
| Precharge | Precharge Period of a Trans. | $nRC_{new}$ - $nRAS_{new}$ |
| Precharge-All | Precharging Period | $nRP$ |
| Reads/Writes | Defined by number of bursts | $nRC_{new}$ |
| Precharged PD | Transition Time + PD Time | $nRP + nPD + nXPDLL$ |
| Active PD (from Read) | Transition Time + PD Time | $nRDPDEN + nPD + nXP$ |
| Refresh | Transition + Refresh | $nRP + nRFC$ |
| Self-Refresh | Transition + Self-Refresh | $nRFC + nSR + nSR_{REF} + nEX_{REF}$ |

This equation (shown in Equation (2.17)) is highly parameterized and together with the individual components described earlier fits a transaction of any length. All the power values obtained from the power equations described before are used, while $P_M(R_{DQ})$ (I/O power) and $P_M(W_{DQ})$ (Termination power) are obtained from Micron's power model [17] for the total numbers of data bits read or written. These parameters can be obtained from any memory controller for every memory transaction.

In Equation (2.17), $i$ refers to the target bank, $j$ refers to the id of a particular transaction in the trace, $nBanks$ refers to the number of banks accessed in parallel by a given transaction in the trace and $N(Act_{BG})$ and $N(Pre_{BG})$ refer to the number of active and precharge cycles in the trace, respectively. $N(ACT)_{(i,j)}$ and $N(PRE)_{(i,j)}$, refer to the number of $ACT$ and $PRE$ commands, and $N(RD)_{(i,j)}$ and $N(WR)_{(i,j)}$ refer to the number of reads and writes per bank ($i$) per transaction ($j$). $N(REF)$ refers to the number of Refreshes in the trace. It should be noted that each *Read* and *Write* corresponds to a burst count *(BC)* of one and hence, transactions with burst counts greater than one are defined by as many read and write commands.

$$
\begin{aligned}
P_D(\mathit{Trace}) = \sum_{j=1}^{\#Trans} \Big( & P_D(Act_{BG}) \times N(Act_{BG}) + P_D(Pre_{BG}) \times N(Pre_{BG}) + \\
& \sum_{i=0}^{nBanks(j)\text{-}1} \big( P_D(ACT, nTL(i,j)) \times nTL(i,j) \times N(ACT)_{(i,j)} + \\
& P_D(PRE, nTL(i,j)) \times nTL(i,j) \times N(PRE)_{(i,j)} + \\
& P_D(RD, nTL(i,j)) \times nTL(i,j) \times N(RD)_{(i,j)} + \\
& P_D(WR, nTL(i,j)) \times nTL(i,j) \times N(WR)_{(i,j)} \big) + \\
& P_M(R_{DQ}) \times (N(R_{DQ}) + N(R_{DQS})) + P_M(W_{DQ}) \times (N(W_{DQ}) + N(W_{DQS})) + \\
& P_D(REF,\ nREF(j)) \times nREF(j) + P_D(PD, nTL(j)) \times nTL(j) + \\
& P_D(SR, nTL(j)) \times nTL(j) \Big) \Big/ \sum_{j=1}^{Trans} nTL(j)
\end{aligned}
\tag{2.17}
$$

The measures for $P_D(REF,\ nREF)$, $P_D(PD,\ nTL_j)$ and $P_D(SR,\ nTL_j)$ correspond to power consumption due to state transitions, into refresh, selected power-down mode and self-refresh mode, besides the power consumption of the respective operation. Transactions that are not bank specific, do not have the suffix $i$.

Since this power equation is highly parameterized, it can be employed for any transaction from any given memory controller. For instance, if a memory controller employs two bank accesses with a close-page policy, for a read transaction with a burst count (BC) of 4, $i = 2$, $j = 1$, $N(ACT)_{(i)} = 2$, $N(PRE)_{(i)} = 2$, $N(RD) = 8$, $N(WR) = 0$, $N(REF) = 0$, $N(R_{DQ}) = 1024$ (128 bits per 8 word burst x 8 bursts - x16 wide DRAM device) and $N(W_{DQ}) = 0$. If an open-page policy is employed, the $N(ACT)_{(i)}$ and $N(PRE)_{(i)}$ values are determined by the need for activating and precharging the particular banks. If power-down or self-refresh modes are employed, the power-down durations are accounted for by the respective power equations.

## 2.6 Adapting to Mobile and Wide IO DRAMs

In comparison to the power equations presented for off-chip DDR3 DRAMs, when it comes to Mobile DRAMs (LPDDR/2/3) and Wide IO 3D-DRAMs, the power model should reflect the following:

(1) It should explicitly consider the multiple voltage sources in mobile and Wide IO DRAMs for different parts of the memory device.

(2) It should reflect the changes in DRAM timing parameters due to removal

of DLLs in mobile and Wide IO DRAMs. This applies especially to the power-down and self-refresh power-saving modes.

(3) It should calculate the I/O power consumption directly from datasheets using $V_{DDQ}$ domain current estimates, since the DRAM has moved 'on-chip' in Wide IO 3D-DRAMs. In the case of LPDDRs (mobile DRAMs), appropriate IO circuitry [121, 128] must be employed as recommended by the DRAM vendor [17].

When it comes to basic memory operations, such as, Activate (ACT), Precharge (PRE), Read (RD), Write (WR) and Refresh (REF), mobile and Wide IO DRAMs are not very different compared to off-chip DRAMs, except for the use of multiple voltage sources and the computation of I/O power consumption.

Hence, we propose a generic power estimation model in Equation (2.18) for all basic DRAM operations and memory states that takes into account the different voltage sources, including $V_{DD1}$, $V_{DD2}$, $V_{DDCA}$ and $V_{DDQ}$.

As can be noticed from the equation, it adds up the corresponding power estimates for all the voltage sources (calculated using the associated current measures) for the relevant memory operations. In the equation, $i$ is used to represent the $V_{DD1}$ and $V_{DD2}$ voltage domains. Note that the current measures corresponding to the $V_{DDCA}$ and $V_{DD2}$ sources have been added up and represented by $V_{DD2}$ (in Equation (2.18) and Table 2.3), since they are both tied to the 1.2V supply.

$$P_D(OP, nTL) = \sum_{n=1}^{nTL}\left(\sum_{i=1}^{2}\left(I_{DDi} \times V_{DDi}\right) + \left(I_{DDQ} \times V_{DDQ}\right)\right)/nTL \tag{2.18}$$

Table 2.3 gives the values of currents and minimum timings (in cc) for the respective memory operations that should be substituted in this generic power equation. Accurate scaling of the power estimates for the basic memory operations, has been presented and described in the previous section. The table also lists background currents consumed when the memory is in the active or precharged states. The I/O current numbers ($I_{DDQ}$) reported for the read/write operations corresponding to the $V_{DDQ}$ source account for the I/O power consumption in the generic power model in Equation (2.17).

In Equation (2.18) and Table 2.3, $nTL$ corresponds to the period for which the corresponding transaction must be active. For instance, $nTL$ for a read and a write command is given by $nRD$ and $nWR$, respectively, which correspond to the period of data transfer during the respective read and write operations. However, $nTL$ equates to $nRAS$, $nRP$ and $nRFC$ for ACT, PRE and REF, commands respectively, which are JEDEC-specified minimum timing constraints to be satisfied for these operations to finish [50].

If these operations continue to be active beyond these minimum timing constraints, appropriate scaling of power numbers must be employed as shown in

Table 2.3: Average Power Consumption of Basic Memory Operations

| *Cmd/State* | $I_{DD1}$ | $I_{DD2}$ | $I_{DDQ}$ | $nTL$ (cc) |
|---|---|---|---|---|
| ACT | $I_{DD0\_1} - I_{DD3N\_1}$ | $I_{DD0\_2} - I_{DD3N\_2}$ | - | $nRAS$ |
| PRE | $I_{DD0\_1} - I_{DD2N\_1}$ | $I_{DD0\_2} - I_{DD2N\_2}$ | - | $nRP$ |
| RD | $I_{DD4R\_1} - I_{DD3N\_1}$ | $I_{DD4R\_2} - I_{DD3N\_2}$ | $I_{DD4R\_Q}$ | $nRD$ |
| WR | $I_{DD4W\_1} - I_{DD3N\_1}$ | $I_{DD4W\_2} - I_{DD3N\_2}$ | $I_{DD4W\_Q}$ | $nWR$ |
| REF | $I_{DD5\_1} - I_{DD3N\_1}$ | $I_{DD5\_2} - I_{DD3N\_2}$ | - | $nRFC$ |
| Active | $I_{DD3N\_1}$ | $I_{DD3N\_2}$ | - | $nACT_{BG}$ |
| Precharged | $I_{DD2N\_1}$ | $I_{DD2N\_2}$ | - | $nPRE_{BG}$ |

the previous section. The $nACT_{BG}$ and $nPRE_{BG}$ timings correspond to the total time period spent in the active and precharged modes, respectively, when performing the basic DRAM operations. These are employed to estimate the background power consumption during these operations. $nTL$ refers to the total operation time window considered when estimating power for the particular operation. It is equal to $nTL$ (as defined before) for all operations except activate and precharge commands, for which it is at least equal to the $nRC$ timing constraint [50] (and may be longer depending on the scaling). Note that for accurate power and energy estimation, the actual command timings from the given memory trace must be employed instead of the minimum timing constraints, and the average power numbers must be appropriately scaled, as in the case of regular off-chip DRAMs.

When modeling power consumption of the power-saving modes in mobile and Wide IO DRAMs, the power model must now take into account the difference in their timings and current measures due to the absence of DLLs, in addition to the introduction of multiple voltage domains (as in the case of basic operations). To enable this, $nXPDLL$ must be replaced by $nXP$ in the power-down equations and $nXSDLL$ must be replaced by $nXS$ in the self-refresh power equation.

## 2.7 DRAMPower Tool

In this section, we describe the different phases in the DRAMPower tool-flow. As stated earlier, the tool employs our high-precision cycle-accurate DRAM power model proposed in this chapter with current measures from vendor datasheets and reports accurate DRAM power and energy measures for traces of any length.

### 2.7.1 DRAMPower Tool Flow

The DRAMPower tool can be employed with two interfaces: (1) DRAM Command traces and (2) Transaction traces. If transaction traces are used, the tool invokes the memory command scheduler (pipelining not supported) developed by Yonghui Li at TU Eindhoven [132], which dynamically schedules DRAM commands assuming closed-page policy, corresponding to the incoming memory transactions. On the other hand, if the tool is employed with DRAM command traces, users must provide command traces (generated by a DRAM memory controller) similar to the ones generated by the scheduler. Once the DRAM command trace is made available (either by the user or by the command scheduler), DRAMPower performs command trace analysis and derives different timing and power metrics for the given trace as described below.

DRAMPower begins by identifying the different memory commands in the command trace (both implicit and explicit) as depicted in Figure 2.14, their target bank and issued time-stamp and defines a complete memory command list by expanding them all (implicit precharges). The user can optionally use a 'NOP' or 'END' command at the end of trace file, with the last cycle as time-stamp to indicate the end of simulation for better analysis accuracy. (This is also explicitly used by the command scheduler). This memory command list is then forwarded to the timing analysis phase of the tool, which identifies the number of activates, (auto) precharges, reads, writes, refreshes, power-downs, self-refreshes, precharged and active cycles and clock cycles in power-down and self-refresh modes. Additionally, it also filters out the number of cycles the memory is idle in the active and the precharged modes. These measures are then forwarded to the DRAM power model through updated counters, which computes the total energy and average power consumed by all basic DRAM operations and power-saving modes using the memory power and timing specification as per Equation (2.17).

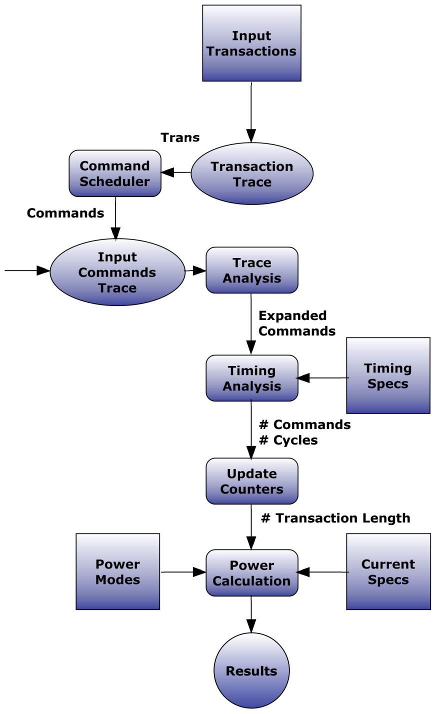


Figure 2.14: Phases in DRAMPower Tool Flow

To perform the trace timing analysis and power consumption computation, the tool uses the target DRAM's architectural, timing and current/voltage details from datasheets in the form of XMLs. The DRAMPower tool has been integrated in the GEM5 simulation environment [120].

### 2.7.2 Command Scheduler

If a transaction trace is used with DRAMPower, it invokes the dynamic DRAM command scheduler (developed by Yonghui Li at TU Eindhoven [132]) to translate incoming memory transactions to equivalent memory commands. The transaction trace should include for every memory transaction, the transaction time-stamp (in clock cycles), the transaction type (READ/WRITE) and the logical memory address (32-bits) generated by the requester in HEX (0x). The DRAM is byte-addressable and uses a flexible and efficient memory map as follows:

**{row}-{bank}-{column}-{BI}-{BC}-{BGI}-{BL}**

Here, BI gives the degree of bank interleaving, BC gives the burst size (count), BGI gives the degree of bank group interleaving (for DDR4) and BL gives the burst length used by the device. The BC and BL address bits are derived from the column address bits, whereas the BI and BGI address bits are derived from the bank address bits.

To generate memory commands for each memory transaction, the command scheduler employs: (1) a closed-page policy (automatically precharges the accessed memory rows) (2) FCFS scheduling across incoming memory transactions (without pipelining), and (3) ASAP scheduling for DRAM commands (i.e. schedules commands as soon as the timing constraints described in Section II are met).

The scheduler supports different request sizes and degrees of bank interleaving and bank group interleaving (for DDR4). Based on these options, it derives the required number of read/writes commands for the given transaction, also referred to as burst count. It uses the memory map described above to translate logical addresses to physical addresses. Users can also select speculative usage of power-down or self-refresh modes (if needed) for idle periods between transactions. The scheduler logs these generated DRAM commands for command trace analysis.

## 2.8 Conclusion

In this chapter, we proposed DRAMPower, a high-level DRAM power model that employs JEDEC-specified current metrics and performs high-precision modeling of the power consumption of different DRAM operations, state transitions and power-saving modes at the cycle-accurate level. We also highlighted the differences between DRAMPower and other existing high-level power models like Micron's and provided updated and new power equations to achieve more accurate power and energy estimates. Finally, we also described the tool flow of the open-source DRAMPower tool and how it employs DRAMPower model for its analysis and computations.

Chapter 3

# Variation-Aware Power Estimation and Performance Characterization of DRAMs

Having described our high-precision cycle-accurate DRAM power model, in this chapter we propose solutions to address the issue of pessimism (for a majority of the devices) in the current measures provided by DRAM vendors in datasheets. Parts of work described in this chapter have been published previously in our papers at DAC 2013 [53] and DATE 2014 [54].

## 3.1 Introduction

Manufacturing-time process (P) variations and run-time variations in voltage (V) and temperature (T) can affect a circuit's performance (delays/timings) and power consumption severely [71–75] and DRAMs are no exception. To counter the effects of these variations, DRAM vendors provide: (1) substantial design-time timing margins to guarantee correct DRAM functionality under worst-case conditions and (2) significant power margins to address the fastest devices and the impact of run-time variations and to improve their yield. Unfortunately, with technology scaling these design margins in power and timings have become large and very pessimistic for the majority of the manufactured DRAMs. While run-time variations are specific to operating conditions and their margins difficult to optimize (since they can vary across use-cases), process variations are manufacturing-time effects and excessive process-margins can be reduced on a per-device basis, if properly identified.

Towards this, we propose a generic post-manufacturing performance characterization methodology for DRAM devices that identifies their actual achievable performance and realistic power estimates under worst-case operating conditions (worst-case power supply (noise) and highest temperature). As a result, we (1) improve the overall performance and energy efficiency of DRAM devices by optimizing timing margins and (2) obtain realistic power measures when retaining original timing margins.

We further evaluate and demonstrate this methodology on 48 DDR3 devices (from 12 identical DIMMs from one vendor [92]), derive their actual delays and current estimates and verify their correct functionality under worst-case operating conditions. In doing so, we achieve up to 33.3% and 25.9% reduction in DRAM Read and Write latencies, respectively. The minimum increase in memory bandwidth (BW) is by 50% for Reads and 35% for Writes. The DRAM energy consumption furthermore reduces by 17.7% when reading and by 15.4% when writing, resulting in improved energy efficiency.

## 3.2 Sources of Variation

When estimating the timing and current consumption margins, vendors consider three primary variation sources (besides aging [88]) that can affect a DRAM's performance: (1) Process (P), (2) Voltage (V) and (3) Temperature (T), also referred to as PVT variations [89].

Process variations are observed due to manufacturing-time disparities in device parameters, such as channel mobility, channel length and oxide thickness [81]. Their impact on DRAM timings and power can vary severely and randomly across all devices produced with the same configuration [87]. Hence, vendors add significant process (timing and current) margins to cover their worst-case impact on the entire manufacture lot.

When it comes to run-time variations in supply voltage and operating temperature, these have a defined and deterministic effect on all manufactured DRAMs, as opposed to the random and distributed effects of process variations.

Voltage variations are represented by noise in the power-supply (reducing operating voltage), which increases the transistor propagation delays in the device. To address this, DRAM vendors define an acceptable operating voltage range (between 1.425V and 1.575V for DDR3 [99, 103]) and add appropriate noise-margins (power and timings) to assure correct functionality in the presence of maximum noise in power supply.

Temperature variations are observed due to two factors: (1) power dissipation of the DRAM during operation (self-heating) and (2) ambient temperature. High operating temperatures also increase the propagation delays and drastically impact leakage power. Hence, DRAM vendors define an operating temperature limit of up to +85°C for commercial DRAMs and add temperature-margins (power and timings) to enable correct DRAM functionality at that temperature.

Besides these variations, aging also impacts DRAM performance [88]. However, all devices susceptible to aging and infant failures are discarded by vendors and hence, all shipped DRAMs are guaranteed to work reliably. This filtering is done by the 'burn-in' test [90], which pre-ages the devices by stress testing them at +125°C and 1.9V and identifies the devices likely to fail early (in $\leq$ 10 years).

In a nutshell, while sufficient voltage (noise) and temperature margins are required to guarantee correct functionality under extreme operating conditions, process-margins can be generously over-dimensioned [91] for a majority of the manufactured DRAM devices. In other words, most DRAMs can perform better (in terms of latency and power consumption) than their datasheet specifications.

In this chapter, we provide an insight into the possible effects of process variations on DRAM performance and power consumption to help improve the accuracy of DRAM power models and enable the efficient use of DRAMs at run time. Towards this, we propose three important contributions: (1) We demonstrate the impact of process variations on DRAM performance and power consumption by performing Monte-Carlo simulations on a detailed DRAM cross-section modeled in NGSPICE [104] and derive *better than worst-case* current measures applicable for a majority of the DRAM devices, from the $\pm 6\sigma$ current distribution obtained from the SPICE simulations. (2) We propose a methodology (set of algorithms) to empirically determine this impact for a given DRAM device, by assessing its actual performance characteristics during the DRAM calibration phase [103] at system boot-time. We employ these performance estimates to identify the equivalent impact on power consumption and to derive *realistic* current estimates. (3) We extend the Monte-Carlo analysis to examine the impact of DRAM architecture parameters, such as capacity, width and frequency, on latency variations and current estimates to derive current measures for DRAMs with different configurations.

Using these methods, we derive possible current distributions for DRAM memories of any configuration, and also determine the actual performance measures and realistic current measure distributions for a given DRAM memory using the characterization step at system boot-time. The derived performance measures can be used to improve the performance of the given DRAM memory and the realistic current measure distributions can be employed in place of the worst-case datasheet values to obtain variation-aware DRAM power and energy estimates when employing typical or optimized timing measures.

## 3.3 Background - DRAM Modeling in SPICE

This section details the background work on modeling a generic DRAM architecture at the circuit-level in NGSPICE [104] as proposed by TU Kaiserslautern in [47,126]. We adapted this circuit-level model as a part of this work to support Monte-Carlo analysis [105] to derive the impact of process variations on DRAM power and performance and to verify DRAMPower power and energy estimates.

### 3.3.1 Baseline DRAM Cross-Section Model

DRAMs consist of memory cells arranged in arrays of rows and columns, organized as a set of memory banks. Each bank is also equipped with a set of row buffers that

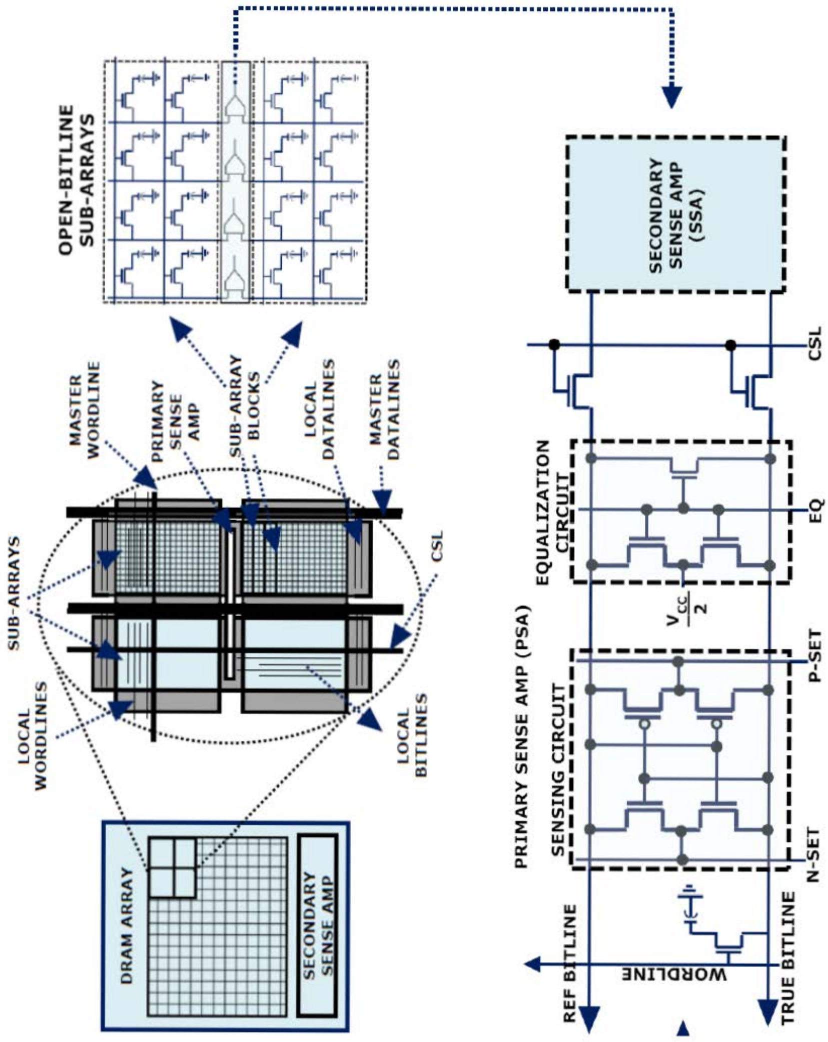


Figure 3.1: DRAM Cross-Section Model

are used as intermediates for reading from or writing into the memory cells. To access a DRAM, a memory controller issues a set of commands in a specific order to perform a given operation [50]. For instance, when reading from the DRAM, an activate command is issued to transfer the data from the cells through the bitlines to a row buffer and then the row buffer is partly read based on the memory's interface width and burst length. Similarly, when writing, data is first written to the row buffer and then a precharge command is sent to store the charge into the DRAM cells. For efficient design, the row buffers are shared between successive rows in a bank, as only one of which can be accessed at a time.

The basic DRAM cell is modeled as a transistor-capacitor (1T1C) pair and stores a single bit of data in the capacitor as a charge. As shown in Figure 3.1, the transistor is controlled by a local wordline (lwl) at its gate, which connects the capacitor to the local bitline (lbl) when turned on (activated). Before reading the data from the memory cell, the bitlines in the memory array are precharged (set to halfway voltage level) using an equalization circuit. When connected, the cell capacitors change the precharged (PRE) voltage levels on the bitlines very slightly. Hence, a set of primary sense amplifiers (PSA) (or row buffer) distributed across memory sub-arrays are used to detect the minute changes and pull the active bitline voltage all the way to logic level 0 or 1. Once the bitline voltage is amplified, it also recharges the capacitors as long as the transistors remain on. The primary sense amplifiers hold the data till all column accesses to the same row are completed, when a precharge is issued. In our model, we used the open bitline array structure and hence differential sense amplifiers (in PSA), which use a reference bitline from a neighboring inactive array segment to detect the minute difference in active bitline voltage. When the Read (RD) command is issued, the data/charge is read out using column select lines (CSLs) from the row buffer (PSA). The data is then switched via master datalines from the PSA to the secondary sense amplifiers (SSA), which connect to the I/O buffers. Once finished, the wordlines can be switched off, safely restoring the charge in the memory cells, before starting to precharge (PRE) the bitlines again.

The memory arrays are organized in a hierarchical structure of memory sub-arrays for efficient wiring. A memory sub-array consists of 256K cells connecting up to 512 cells per local bitline and per local wordline. 16 memory sub-arrays connect to one master wordline forming 4Mb blocks. 16 master wordlines and 16 column select lines (CSLs) connect the 256 memory sub-arrays to form 64Mb memory array macros. The row and column decoders and the master wordline drivers are placed per memory array. The N-Set and P-Set control signal drivers used for activating the primary sense amplifiers are shared among sub-arrays.

### 3.3.2 DRAM Cross-Section SPICE Simulations

In the NGSPICE [104] model of the DRAM cross-section, we employed the BSIM model cards [105] built on Low Power Predictive Technology models (LP-PTM) [106], since there are no openly available technology libraries specific to DRAMs. As a result, the LP-PTM devices had to be adapted to ensure functional and timing correctness of the simulated DRAM cross-section, as observed from the simulations. We modeled the memory cell architecture (of $6F^2$ area), the equalization circuit, the wordline driver, and the sense amplifier in our model described earlier, using the designs suggested in [22], [80] and [50]. The baseline DRAM configuration targets a 1Gb DDR3-1066 (533MHz) x8 memory with core timings of 7-7-7 cc at 45nm. We chose 45nm, since it is the common technology node employed by vendors for DDR3 memories including Samsung, Micron and Hynix. Below, we present the timings and voltages of the different signals corresponding to basic DRAM operations: ACT-RD-PRE in Figure 3.2. The aim is to validate functionality and timing for these DRAM operations as per [50].

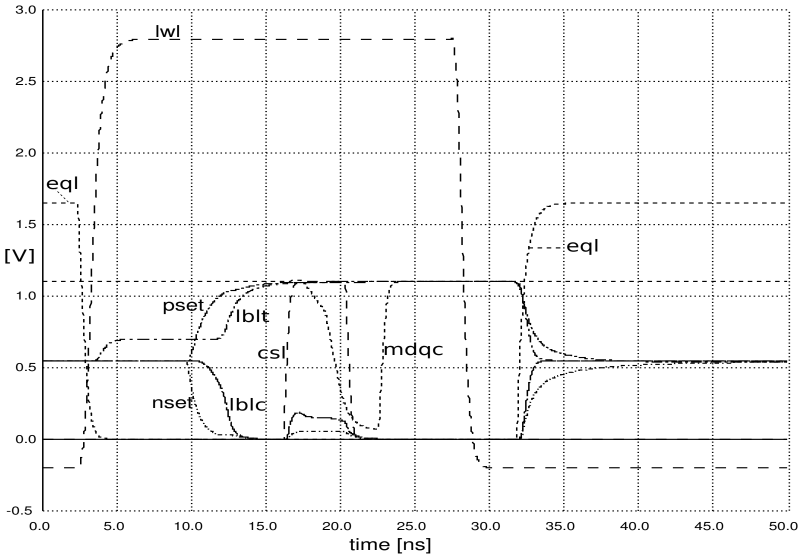


Figure 3.2: ACT-RD-PRE behavior in DRAM Cross-Section

As depicted in the figure, first the equalization circuit (eql) forces both the true (active) bitline (lblt) and the complementary (reference) bitline (lblc) to the same reference voltage (0.55V). This is followed by the local wordline (lwl) going high to begin the activation process that switches the relevant transistors on and connects the cell capacitors to the corresponding local bitlines. Simultaneously, the equalization circuit (eql) de-activates to enable sensing of the change in bitline voltage due to the charge transfer. As the wordline high reaches the required voltage of 2.8V at around 5ns, the pre-sensing phase begins to create a minimum voltage difference (around 200mV) between the reference (lblc) and active (lblt) bitlines at the PSA. This is followed by the activation of the sensing circuit by N-Set and P-Set control signals, which drives the active bitline (lblt) to logic level 1 (the charge stored in the cell corresponds to 1 here) and the reference bitline (lblc) to 0 at around 15ns. Both the lblt and pset signals are driven to the core voltage of 1.1V, while lblc and nset signals are driven to 0V.

This is followed by the Read operation depicted by the rising column select line (csl) voltage at 18ns. Following this, the charges detected at all the PSAs in a memory sub-array are transferred via their respective local datalines to a master dataline, which is reflected by the current drawn from the NMOS components of the PSA (lblc) and the gradual drop in voltage level of master dataline complement signal (depicted by mdqc). Once the mdqc drops by around 200mV (in relation to the core voltage), the data is sensed at the SSA at around 18ns.

Once the Read operation finishes, (data received by SSA) the mdqc (master dataline complement) is precharged back to its reference voltage (1.1V) at around 24ns and the local wordline is switched off at around 28ns. After a short delay to close the transistor and avoid destroying the charge in the cell, the sensing circuit in the PSA is deactivated and the bitline equalization re-starts at around 33ns. This precharges both the local bitlines back to reference voltage levels, finishing at 50ns, as expected for a DDR3-1066 memory [103].

Similar to the modeling of the Read operation, the Write operation is modeled to copy data from the IO buffers to the SSA and then on to the DRAM cells, within predefined timings. To model a Refresh operation, the SPICE model translates it approximately into a set of 'n' internal row activations and precharges (without explicit commands), where 'n' is the number of banks in the DRAM (i.e. refreshing 1 row per bank). Hence, a Refresh operation is modeled as activating and precharging 'n' rows in tRC time per row, each ACT-PRE operation to a new row initiated after tRRD time (activation to activation time) after the last row. To model DRAM power-down, the clock receiver is turned OFF and the CKE is disabled. To model Self-Refresh, almost all components in the DRAM are turned off. When powering back up, the switched off receivers are turned ON again. We observed accurate functionality and timing as per [50] for all DRAM operations, thus, validating the timing correctness and functionality of the modeling of the DRAM cross-section.

## 3.4 Baseline Monte-Carlo Analysis

We employ the DRAM cross-section described in the last section, and observe the impact of variation on delay and power consumption using Monte-Carlo analysis, in the this section.

We present the results from Monte-Carlo analysis on our verified 1Gb DDR3-1066 x8 DRAM cross-section, described in Section 3.3.2. Towards this, we vary global device parameters such as channel length, channel mobility, and oxide thickness and local device threshold voltage ($V_{th}$) (primarily the variations in line edge roughness (LER) [81]), besides the interconnect parameters including wire width and wire thickness, within pre-defined variation ranges. We obtained the variability ranges (scaling metric ($\sigma$) in the corresponding Gaussian distributions) for these parameters from the ITRS technology requirements on Design for Manufacturability [107] and Modeling and Simulation [108] and the variation models of transistors from [81, 82]. We also introduce spatial-correlations in the variations among neighboring transistors, due to expected similarity in the parametric variations. Using these variability values, we performed Monte-Carlo runs on 1000 circuit instances reflecting the variations in all the device and interconnect parameters. From our observations, the variation in the device $V_{th}$ parameter had the biggest impact on the circuit delay and current consumption [81], since it is directly influenced by the variations in the global device parameters. As expected, the active (dynamic) DRAM currents and frequency increased linearly against the variations in the $V_{th}$ parameter [84, 85], while the leakage currents increased exponentially. Hence, we analyzed the variations in leakage currents on the natural logarithmic scale [84] to obtain the $\sigma$ values of their distributions corresponding to those of the $V_{th}$ parameter.

The variations in the local and global device parameters at 45nm based on [81, 82,107,108], as used in our simulations are presented in Table 3.1. These measures correspond to the variability introduced in the device parameters per $\sigma$ change in their Gaussian distributions. In the table, the 'w' corresponds to nominal gate width and 'l' corresponds to nominal gate length, the $\sigma$% value gives the relative variation to the nominal values ($\mu$) obtained from the PTM models [106], while the $\sigma$ values correspond to the absolute values of variation.

Table 3.1: Transistor Process Parameter Variations

| *Tech* *nm* | *Mobility* $\sigma$ *(%)* | $V_{th}$ *(LER)* $\sigma$ *(V)* | *Length* $\sigma$ *(m)* | $T_{ox}$ $\sigma$ *(%)* |
|---|---|---|---|---|
| 45 | 8.2 | 3.0e-9/$\sqrt[2]{(w\times l)}$ | 45e-9×0.03 | 1.67 |

The impact of process variation on the timing behavior of the DRAM cross-section as observed from 1000 Monte-Carlo simulations considering $\pm 1\sigma$ variations in the device and interconnect parameters, are presented in Figure 3.3. In this figure, we present the effects on the local wordline activation (lwl), and the sensing of the true (lblt) and complementary (lblc) bitlines by the PSA.

As can be observed from the figure, there is a significant impact on the (delays) timings of the operations associated with the wordline and bitlines. For instance, the local word line reaches its required potential (upon activation) of 2.8V at between 4ns and 6ns instead of at 5ns, which was the case for the baseline configuration without any variation (shown in Figure 3.2). Similarly, the bitlines reach their potential (upon sensing by the PSA) between 13ns and 17ns, compared to around 15ns in the baseline configuration (Figure 3.2). The variations in the bitlines and wordline impact the activation latency given by the core-timing parameter *tRCD*, thereby impacting both the DRAM delays (latencies) and power consumption. The worst-case effect on these timing measures are shown in Table 3.2 (Worst delays at lowest operating voltage). We refer to the sum of these four critical timing metrics (*tRCD*, *tRTP*, *tRP* and *tWR*) as the *functional latency* (FL) of the DRAM device.

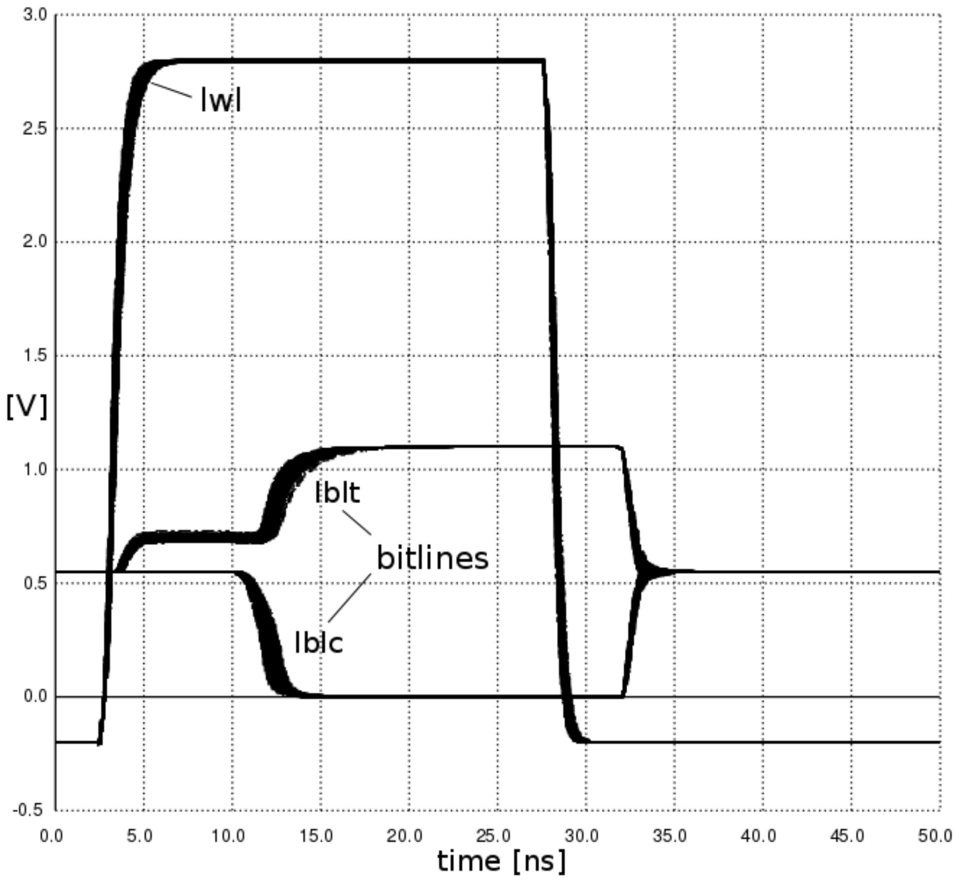


Figure 3.3: Variation Impact on Bitline and Wordline

Table 3.2: Variation Impact on Timing Measures @ +85°C and 1.425V

| *Timing* | $\mu$ *ns* | $\sigma\%$ | $+1\sigma$ *ns* | $+3\sigma$ *ns* | $+5\sigma$ *ns* |
|---|---|---|---|---|---|
| *tRCD* | 10 | 10 | 10 | 13.5 | 15 |
| *tRP* | 7.5 | 15 | 9 | 11.5 | 15 |
| *tRTP* | 7.5 | 5 | 7.875 | 8.75 | 10 |
| *tWR* | 5 | 20 | 6 | 9 | 15 |

To translate the effect on timings in terms of power and different characteristic DRAM currents, we identify the dynamic (active and background) currents as: $I_{DD0}$, $I_{DD1}$, $I_{DD2N}$, $I_{DD3N}$, $I_{DD4R}$, $I_{DD4W}$, and $I_{DD5}$, and the static (leakage) currents as: $I_{DD2P0}$ and $I_{DD6}$ (when the clock is disabled). In Table 3.3, we show the impact of process variation on the different currents for a baseline 1Gb DDR3-1066 (533MHz) x8 DRAM memory, under worst-case operating conditions for power consumption, which are given by 1.575V and +85°C (Highest power at highest voltage).

Table 3.3: Variation Impact on Current Measures @ +85°C and 1.575V

| *Current* | $\mu$ *mA* | $\sigma\%$ | *+1$\sigma$ mA* | *+3$\sigma$ mA* | *+5$\sigma$ mA* |
|---|---|---|---|---|---|
| $I_{DD0}$ | 98.4 | 2.37 | 100.7 | 105.5 | 110.6 |
| $I_{DD1}$ | 104.3 | 2.32 | 106.7 | 111.7 | 116.9 |
| $I_{DD2N}$ | 37.7 | 4.77 | 39.5 | 43.4 | 47.6 |
| $I_{DD3N}$ | 41.5 | 5.71 | 43.8 | 49.1 | 54.7 |
| $I_{DD4R}$ | 118.1 | 2.96 | 121.6 | 128.9 | 136.6 |
| $I_{DD4W}$ | 123.4 | 2.75 | 126.7 | 133.9 | 141.2 |
| $I_{DD5}$ | 146.1 | 2.15 | 149.6 | 155.7 | 164.2 |
| $I_{DD2P0}$ | 8.41 | 13.69 | 9.56 | 12.3 | 15.9 |
| $I_{DD6}$ | 8.04 | 14.02 | 9.17 | 11.9 | 15.5 |

In this table, we present the nominal measures along with the $+1\sigma$, $+3\sigma$ and $+5\sigma$ estimates for the different $I_{DD}$ currents obtained after 1000 runs of Monte-Carlo analysis. We also provide $\sigma\%$ value to get relative variation for the different currents. As can be noticed from the table, the $+5\sigma$ estimates (that are closer to datasheet estimates [99]) are significantly higher than the nominal ($\mu$) values for the different $I_{DD}$ currents.

Combining the effects on performance and power, next we present the impact of $\pm1\sigma$ variations on basic memory operations including Activation, Read, Precharge and Power-down. In Figure 3.4, we present a Q-Q (quantile) plot comparing the distributions observed in active and leakage currents (power) and the delays *tRCD* and *tData* (combining *tRD* with Read to Precharge period), corresponding to $\pm1\sigma$ variations, when simulating a set of activation-read-precharge operations. If the two distributions used for the comparison study are similar, all the points in the QQ plot must lie close to the line x = y. Since we employed a Gaussian distribution in modeling the device voltage variations, we expect a Gaussian distribution in both the observed performance and power measures for these operations. By overlapping the two sets of measures and normalizing, the similarity between the distributions of both measures can be identified as a linear relation between them, as observed in the figure.

Using the current distributions from Table 3.3 and the distribution of the functional latencies from Table 3.2, in Figure 3.5, we overlap them to obtain the complete performance-power-variation relation in the $\pm6\sigma$ form. Here, the functional latency range for DDR3-1066 devices is between 30ns and 55ns (sum of the

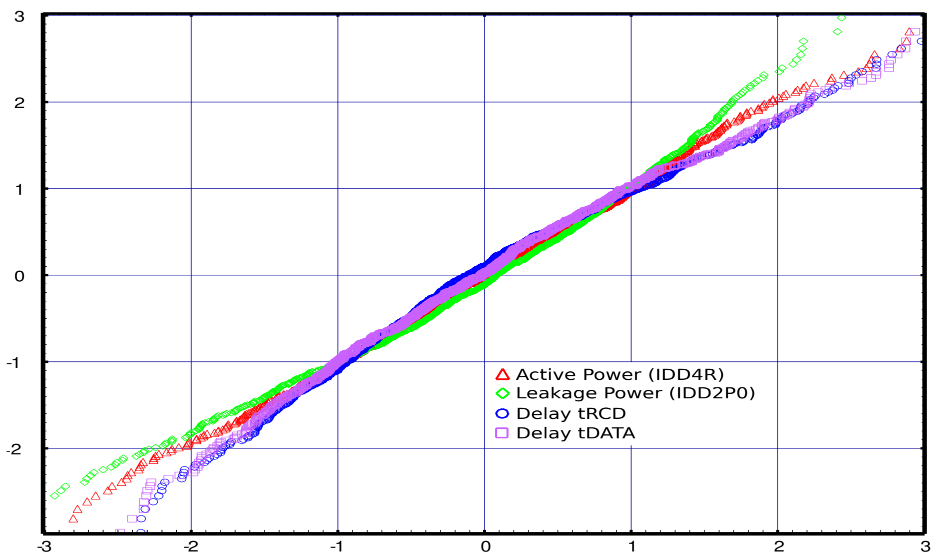


Figure 3.4: Impact on Currents and Timing

timing parameters). *The datasheet (DS) current measures are identified at +5$\sigma$ position (as stated in Chapter 1) and the datasheet (DS) timing measures are identified at -6$\sigma$ position (speed-bin).* (Lower value of functional latency corresponds to faster devices.)

As can be noticed from the plot in Figure 3.5, the current measures observed at the datasheet timings (speed-bin defined) are much lower than the worst-case datasheet current measures observed for high speed devices. Hence, to be conservative and yet 'better than worst-case', we propose to employ +3$\sigma$ current measures in place of the worst-case measures, since they cover more than 97% of the devices manufactured with the same configuration. These $\sigma$ measures can be reverse-engineered from any DRAM device datasheet using the current variation factor ($\sigma$%) as shown in Table 3.3.

Having looked at a specific DRAM configuration, 1Gb DDR3-1066 devices, the next section presents the current distributions for different DRAM configurations, varying their frequency, capacity and data-width.

## 3.5 System Parameters Impact on Variation

DRAM vendors sort the memories by three system parameters: frequency, capacity and data-width. In this section, we present results from more Monte-Carlo simulations to analyze the impact on $\mu$ and $\sigma$% for the different currents when these system parameters change. Our baseline configuration targeted 1Gb DDR3-533MHz (1066) x8 memories. In this experiment, we alter the system parameters individually to simulate different configurations.

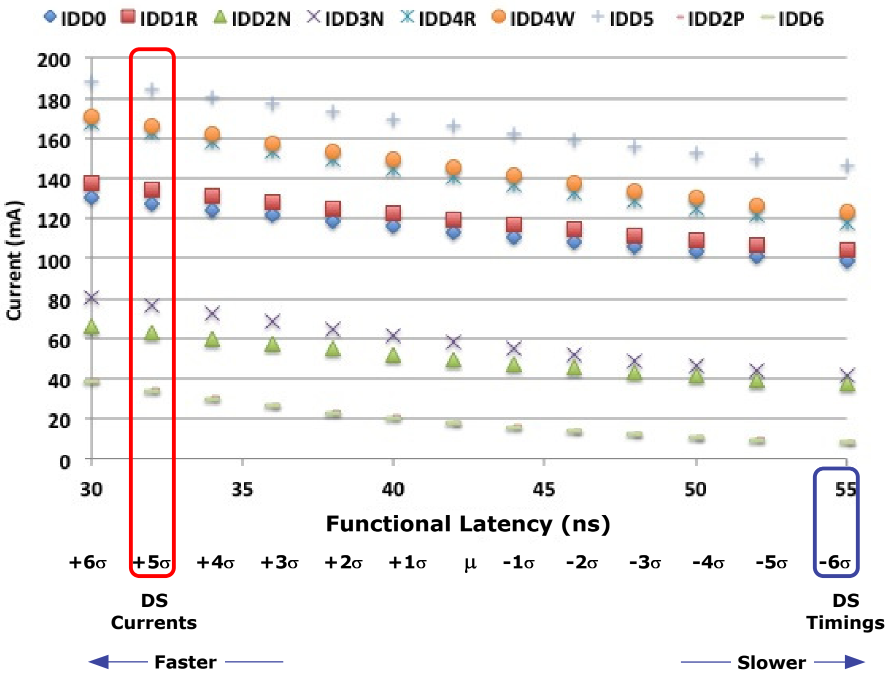


Figure 3.5: Functional Latency Vs. Current Consumption

Accordingly, we change: (1) the frequency to 800MHz and simulate a 1Gb-800MHz-x8 memory, (2) the capacity to 2Gb and simulate a 2Gb-533MHz-x8 memory, and (3) the data-width to x16 to simulate a 1Gb-533MHz-x16 memory and observe the impact on $\mu$ and $\sigma\%$ in Table 3.4.

Table 3.4: System Parameters Vs. Currents

| | Baseline | | Freq | | Capacity | | Width | |
|---|---|---|---|---|---|---|---|---|
| Config | 1Gb-533-x8 | | 1Gb-800-x8 | | 2Gb-533-x8 | | 1Gb-533-x16 | |
| $I_{DD}$ Type | $\mu$ mA | $\sigma$ % | $\mu$ mA | $\sigma$ % | $\mu$ mA | $\sigma$ % | $\mu$ mA | $\sigma$ % |
| $I_{DD0}$ | 98.4 | 2.4 | 112 | 2.6 | 99.3 | 2.5 | 98.4 | 2.37 |
| $I_{DD1}$ | 104 | 2.3 | 118 | 2.2 | 105 | 2.5 | 113 | 2.22 |
| $I_{DD2N}$ | 37.7 | 4.8 | 42.7 | 4.5 | 46.5 | 6.1 | 37.7 | 4.77 |
| $I_{DD3N}$ | 41.5 | 5.7 | 56.7 | 4.5 | 49.9 | 5.3 | 41.5 | 5.71 |
| $I_{DD4R}$ | 118 | 2.9 | 153 | 3.5 | 127 | 3.3 | 208 | 3.14 |
| $I_{DD4W}$ | 123 | 2.7 | 159 | 4.1 | 132 | 3.7 | 213 | 2.6 |
| $I_{DD5}$ | 146 | 2.1 | 161 | 2.4 | 184 | 2.2 | 146 | 2.15 |
| $I_{DD2P0}$ | 8.4 | 13.7 | 8.4 | 13.7 | 16.6 | 16.1 | 8.4 | 13.7 |
| $I_{DD6}$ | 8 | 14 | 8 | 14 | 13.7 | 19.9 | 8 | 14 |

As shown in the results, when increasing the frequency from 533MHz to 800MHz, all currents except the leakage currents scale up linearly due to their dependency on the clock. When increasing the memory density, all currents scale up linearly due to the doubling of the number of memory cells and primary sense amplifiers. However, when the data-width is doubled, while retaining the same page-size (1KB), only the currents reflecting data transfer, viz., $I_{DD1}$, $I_{DD4R}$ and $I_{DD4W}$ are affected, since only the number of data bits accessed during the column accesses increases. Similarly, when a combination of system parameters change, the impact on current measures can be estimated directly from the results in Table 3.4 by adding the corresponding impact on $\mu$ and multiplying by $\sigma$%, for one parameter at a time, considering the most influential parameter first (conservative and determined by % change in $\mu$), as derived in Table 3.5.

Table 3.5: Multi-Parameter Impact on Currents

| | F&C | | F&W | | C&W | | F&C&W | |
|---|---|---|---|---|---|---|---|---|
| Config | 2Gb-800-x8 | | 1Gb-800-x16 | | 2Gb-533-x16 | | 2Gb-800-x16 | |
| $I_{DD}$ Type | $\mu$ mA | $\sigma$ % | $\mu$ mA | $\sigma$ % | $\mu$ mA | $\sigma$ % | $\mu$ mA | $\sigma$ % |
| $I_{DD0}$ | 113 | 2.5 | 112 | 2.4 | 99.3 | 2.4 | 113 | 2.4 |
| $I_{DD1}$ | 119 | 2.4 | 127 | 2.3 | 115 | 2.3 | 128 | 2.3 |
| $I_{DD2N}$ | 48.6 | 6.3 | 42.7 | 5.0 | 43.7 | 5.1 | 48.6 | 5.1 |
| $I_{DD3N}$ | 52.4 | 5.5 | 56.7 | 5.9 | 47.2 | 6.0 | 52.4 | 6.0 |
| $I_{DD4R}$ | 159 | 3.4 | 244 | 3.2 | 214 | 3.2 | 250 | 3.2 |
| $I_{DD4W}$ | 163 | 3.8 | 250 | 2.7 | 217 | 2.7 | 253 | 2.7 |
| $I_{DD5}$ | 194 | 2.3 | 161 | 2.2 | 179 | 2.2 | 194 | 2.2 |
| $I_{DD2P0}$ | 14.1 | 18.3 | 8.4 | 15.5 | 14.1 | 15.9 | 14.1 | 16.2 |
| $I_{DD6}$ | 11.2 | 22.6 | 8.0 | 15.9 | 11.2 | 16.8 | 11.2 | 17.2 |

By combining these current consumption measures with the performance metrics as shown in Figure 3.5, similar power-performance relations can be obtained for DRAM devices of different capacity, width and frequency. As stated before, $+3\sigma$ current measures can be employed as generic measures for >97% devices in the particular generation. Besides these generic current measures, to obtain realistic current measures for each given DRAM device, it is important to characterize the device before determining the same. Next, we propose a DRAM performance characterization methodology that determines the actual performance metrics of any given DRAM (as against the datasheet measures). These performance metrics can be overlapped with the power-performance relation obtained in Figure 3.5. Using this, we can position the given DRAM on the $\pm 6\sigma$ power-performance distribution curves and derive realistic current measures corresponding to the actual performance metrics of the particular DRAM. We further propose to use $+1\sigma$ current estimates relative to the positioning of the DRAM to be conservative. *As stated earlier, for nominal DRAM usage at the minimum timings, we propose to employ $+3\sigma$ current measures from the distribution.*

## 3.6 DRAM Memory Characterization

In the last section, we observed the expected impact of process variations on DRAM delays and power consumption by performing Monte-Carlo simulations on a detailed DRAM cross-section. In this section, we propose a post-manufacturing performance characterization methodology (set of algorithms) to empirically determine this actual impact for a given DRAM memory. By doing so, we relate a DRAM's actual performance to the impact of process variations on DRAM currents.

We begin by identifying a set of requirements and solutions for a comprehensive DRAM characterization methodology. We then propose three algorithms that can be used at run time to determine the actual timings (pruning the excessive process margins) for a given DRAM device in nominal conditions (1.5V and room temperature). Next, we derive conservative timing margins and compensated timings to address noise and temperature variations using our NGSPICE DRAM model and assure correct DRAM functionality under worst-case operating conditions (1.425V and +85°C).

Since this algorithm helps derive optimized timings only, to determine the impact on power consumption, we overlap these optimized timings on the delay-power relation obtained using the SPICE simulations in Figure 3.5 in the previous section, and identify the corresponding current measures. We also extend this relation to estimate the impact of the three system parameters viz., capacity (C), frequency (F) and width (W), on DRAM performance and power consumption, to enable use with DRAMs of different configurations.

### 3.6.1 Requirements and Solutions

The goal of this work is to *derive a set of run time characterization algorithms that generate a sequence of DRAM commands to conclusively determine if the DRAM continues to operate correctly at the tested PVT operating points.* Towards this, we first define a set of requirements and solutions to derive a comprehensive DRAM characterization methodology:

(1) Requirement: *Assure completion of Read/Write operations*: Data previously written into the row may be partially retained in the associated row buffer, if the following precharging and activation operations are only partially completed. When reading from a DRAM row, it is therefore important to assure completion of these operations and that the correct data is being read from the cells.

Solution: Internally, DRAMs employ an open-bitline array structure for their row buffers [50], which implies that the row buffers are shared between adjacent memory rows (as described in Section 3.3). This pair-wise sharing enables use of bitlines of either of the rows as voltage reference by the differential sense amplifiers in the row buffers, when the other row is being accessed.

Hence, if both of these rows are written into in succession with the negated dataset, it would assure that data in the row buffer for the first row is being over-written with flipped bits of data for the second row. Hence, when reading from the first row, if the correct data is observed, the writing and reading operations with the optimized timings were successful.

(2) Requirement: *Varied datasets*: It is important to detect the impact of the charge in neighboring cells on the cell in the middle [93, 94, 109] due to cross-talk across bitlines, which may happen at high speeds. For instance, if the neighboring cells hold the opposite charge as the cell in the middle, it is important to observe the effect on that cell (if any). It is also important to test each cell for stuck-at faults [93, 94].

Solution: The datasets should include data with: (a) neighboring bits flipped and (b) all bits set to '1' and all bits set to '0'. According to Requirement (1), each dataset should have a corresponding negated dataset. Thus, we derive the following datasets (ds):

ds [0] - {0xA5A5A5A5}   ds [1] - {0x5A5A5A5A}
ds [2] - {0xFFFFFFFF}   ds [3] - {0x00000000}

To properly test a memory location, we should employ all combinations of datasets, while writing into and reading from alternating rows with flipped datasets (Requirement (1)). Hence, we derive the following test sets (ts) in Table 3.6:

Table 3.6: Test Sets

| | ts [i] [0] | ts [i] [1] | ts [i] [2] | ts [i] [3] |
|---|---|---|---|---|
| ts [0] [j] | W [x] [0] | W [x+1] [1] | R [x] [0] | R [x+1] [1] |
| ts [1] [j] | W [x] [1] | W [x+1] [0] | R [x] [1] | R [x+1] [0] |
| ts [2] [j] | W [x] [2] | W [x+1] [3] | R [x] [2] | R [x+1] [3] |
| ts [3] [j] | W [x] [3] | W [x+1] [2] | R [x] [3] | R [x+1] [2] |

Here in a W/R [x][y] test, W/R refers to writing or reading operation, x refers to the memory row in the bank and y refers to the dataset element being written or verified against. The test works in the row-major order.

(3) Requirement: *Test all DRAM cells comprehensively*: Some DRAM cells may perform better than others within or across different DRAM devices on a DIMM [93,94,109]. Hence, it is important to test all the DRAM cells and identify the weakest cells that define overall performance of the DIMM. This is to ensure correctness of functionality of all DRAM cells under optimized timings.

Solution: In the test sets defined before, each Write and Read corresponds to accesses of 64 bytes (512 cells) of data. The entire test set writes into and reads from these 512 cells per row, 4 times each, on two rows in a test set and with different datasets. Once the test set finishes, it moves to the next 512 cells in the same two rows till it reaches the end of the rows, before switching to the beginning of the next two rows in the same bank. These test sets are then repeated over all banks.

(4) Requirement: *Worst-case operating conditions test*: Functional correctness must be assured under worst-case operating conditions (maximum power-supply noise and maximum temperature).

Solution: While pruning the excessive process margins, requisite noise and temperature margins must be retained. These noise and temperature margin compensations must be conservatively derived and verified.

### 3.6.2 DRAM Characterization Algorithms

In this section, we define a set of algorithms that determine the fastest timing measures for a given DRAM device at run time under nominal test conditions (+27°C and 1.5V supply). Algorithm 1 (Memory Check), writes, reads and verifies each of the datasets based on the combinations explored by the different testsets. For each iteration of Algorithm 1, a new Read command pattern (RD_Patt[ ]) and a new Write command pattern (WR_Patt[ ]) is provided by the best timings check (btc) function in Algorithm 2, which reduces a target timing parameter one cycle at a time when generating these command patterns. These command patterns consist of a pre-defined set of scheduled DRAM commands like ACT-RD-PRE based on timing constraints described in Table 1.1, but using the reduced timing values. These operations are performed over the entire memory range, writing and reading four times per memory location, in the order specified before. By doing so, Algorithm 1 verifies correct DRAM functionality, with the new test patterns. This algorithm satisfies the first 3 requirements in Section 3.6.1.

In Algorithm 2, RD_Patt[1] corresponds to the DRAM command issued on clock cycle #1 in the Read test pattern. Once explicit DRAM commands (ACT, READ, WRITE, PRE) are issued, an explicit NOP is issued to indicate last clock cycle in the test patterns. For each DRAM device, the test trigger in Algorithm 3, calls Algorithm 2 for each timing parameter to be optimized. It first targets nRCD and nRP timings, since they concern independent memory operations (activating and precharging). It optimizes one of them and then uses the derived minimum value to optimize the other (ordering is irrelevant). Next, it targets the Read/Write to precharge timings viz., nRTP and nWR. They are dependent on each other since they both inherently include nCL (column access latency). In this case, it first individually minimizes both of them to identify their best measures. Next, it employs the minimal independent measure of nRTP and tries optimizing nWR and then employs the minimal independent measure of nWR and tries optimizing nRTP. Finally, it identifies one of the combinations of nRTP and nWR that gives the minimum sum of the two parameters. If the application is memory Read/Write dominant, the appropriate timing (nRTP or nWR) may be targeted specially for minimization. Each parameter is reduced from its datasheet value to 1 or till the chip returns a FAIL.

In all, Algorithm 1 (Mem_Check) is called 26 times, with each call lasting 4 seconds. The entire memory characterization finishes in less than 2 minutes. These algorithms can be employed once during system boot time. Together these algorithms help derive the fastest timings under nominal conditions at which the DRAMs continue to work. However, these timings must be compensated for noise and temperature variations, which is addressed next.

**Algorithm 1:** Memory Check (Mem_Check)

```
Require: RD_Patt[ ], WR_Patt[ ]
 1: {Comment: FOR {all banks, rows, columns}}
 2: for b = 0 → banks do
 3:    for r = 0 → rows do
 4:       for c = 0 → columns do
 5:          for i = 0 → 3 do
 6:             for j = 0 → 3 do
 7:                mem[b][r][c]= ts[i][j] {Comment: Data written using WR_Patt}
 8:                mem[b][r+1][c]= ts[i][j+1] {Comment: Data written using WR_Patt}
 9:                j+=2
10:                ts[i][j] = mem[b][r][c] {Comment: Data read using RD_Patt}
11:                ts[i][j+1] = mem[b][r+1][c] {Comment: Data read using RD_Patt}
12:             end for{j}
13:             {Comment: Check if data is written/read correctly}
14:             if ts[i][2] ≠ ts[i][0] then
15:                Return FAIL
16:             else if ts[i][3] ≠ ts[i][1] then
17:                Return FAIL
18:             end if
19:          end for{i}
20:       end for{c}
21:    end for{r}
22: end for{b}
23: {Comment: END FOR {all banks, rows, columns}}
24: Return PASS
```

**Algorithm 2:** Best Timings Check (btc)

```
Require: Test_ID, RCD, RP, RTP, WR
 1: Init [ ] = {RCD,RP,RTP,WR}
 2: Min [ ] = {1,1,1,1};
 3: # Define: WL = {Write Latency (Datasheet value)}
 4: # Define: BL = 8 {Burst Length}
 5: for $i = Init[Test\_ID] - 1 \rightarrow Min[Test\_ID]$ do
 6:    Init[Test_ID] = i
 7:    RCD=Init[0], RP=Init[1], RTP=Init[2], WR=Init[3]
 8:    RD_Patt[1] = ACT
 9:    RD_Patt[RCD] = READ
10:    RD_Patt[RCD+RTP] = PRE
11:    RD_Patt[RCD+RTP+RP-1] = NOP
12:    WR_Patt[1] = ACT
13:    WR_Patt[RCD] = WRITE
14:    WR_Patt[RCD+WL+WR+BL/2] = PRE
15:    WR[_PattRCD+WL+WR+BL/2+RP-1] = NOP
16:    {Comment: For other cycles of RD_Patt and WR_Patt, no command is issued.}
17:    if Mem_Check(RD_Patt[ ],WR_Patt[ ]) == FAIL then
18:       Return Init[Test_ID] + 1
19:       Break
20:    end if
21: end for{i}
22: Return Init[Test_ID]
```

**Algorithm 3:** Test Trigger

```
Require: RCD,RP,RTP,WR
 1: for j = 0 → 3 do
 2:    {Comment: For all chips in the DIMM}
 3:    bRCD[j] = btc(0,RCD,RP,RTP,WR)
 4:    bRP[j] = btc(1,bRCD[j],RP,RTP,WR)
 5:    bRTP_ind[j] = btc(2,bRCD[j],bRP[j],RTP,WR)
 6:    bWR_ind[j] = btc(3,bRCD[j],bRP[j],RTP,WR)
 7:    bRTP_wr[j] = btc(2,bRCD[j],bRP[j],RTP,bWR_ind[j])
 8:    bWR_rtp[j] = btc(3,bRCD[j],bRP[j],bRTP_ind[j],WR)
 9:    if (bRTP_wr[j] + bWR_ind[j]) ≤ (bRTP_ind[j] + bWR_rtp[j])then
10:       bRTP[j] = bRTP_wr[j], bWR[j] = bWR_ind[j]
11:    else
12:       bRTP[j] = bRTP_ind[j], bWR[j] = bWR_rtp[j]
13:    end if
14: end for{j}
```

### 3.6.3 Conservative Voltage-Temperature Compensations

The goal of this work is to eliminate the excessive process-margins from a DRAM post-manufacturing, while retaining the requisite temperature and noise margins and assuring functional correctness and optimal performance under worst-case conditions. These can be identified if either the worst-case test conditions are employed on the DIMM or equivalent worst-case circuit-level simulations are performed. Although we have access to such an experimental setup, the same cannot be expected of the users of this proposed methodology. Hence, we propose to derive the impact of noise and temperature on these critical timing measures using the NGSPICE DRAM cross-section model. *Note, we do not include in the impact of process variations, since the aim is to derive voltage and temperature margins.* The target device is a 1Gb DDR3-533MHz x16 device [99] at 45nm.

From our SPICE experiments, we derive the following results: Table 3.5 presents the impact of power-supply noise and temperature variations on the critical DRAM timings (described earlier in the chapter). The power-supply variation impact is derived at +85°C for voltage range between 1.425V and 1.575V. As observed, the delays increase with increase in noise (reduced power-supply). The temperature-variation impact is derived at 1.425V. Operating temperatures between +27°C (nominal) and +85°C (maximum) are simulated and as expected; the higher the temperature, the longer the delays. As before, tRCD refers to nRCD in ns.

Table 3.7: Impact of Noise and Temperature on Timings

| | Noise @ +85°C | | | Temperature @ 1.425V | | |
|---|---|---|---|---|---|---|
| *Timings (ns)* | *@1.575V* | *@1.5V* | *@1.425V* | *@+27° C* | *@+70° C* | *@+85° C* |
| *tWR* | 5.19 | 5.28 | 5.38 | 4.79 | 5.22 | 5.38 |
| *tRP* | 5.60 | 6.01 | 6.68 | 6.48 | 6.64 | 6.68 |
| *tRCD* | 9.64 | 9.83 | 10.02 | 9.00 | 9.72 | 10.02 |
| *tRTP* | 9.64 | 10.13 | 10.66 | 9.12 | 10.09 | 10.66 |

Using these observations, we derive the voltage and temperature margins for 1Gb DDR3 devices in Table 3.8, to be added to the timings obtained at nominal conditions using the algorithms in Section 3.6.2.

We also extend our analysis to study the influence of increasing DRAM capacities on these margin compensations, since that increases the number of row buffers in the DRAM, which affects the timings. We present the results for higher capacity (2Gb) DDR3 memories in Table 3.8 as well. *Since we are minimizing analog delays, the DRAM frequency does not influence the margins, hence, they will be the same for all frequencies.*

As can be noticed, the margins for tRTP increased with capacity, since it may take slightly longer to finish a Read operation due to the increase in the number of row buffers and longer bitlines to traverse, to read the data out.

When employing these derived margins, we propose rounding up the resultant

Table 3.8: Conservative Margins

| | 1Gb | | | 2Gb | | |
|---|---|---|---|---|---|---|
| | *Nominal* | *WC* | *Diff* | *Nominal* | *WC* | *Diff* |
| *Timings* | *1.5V +27° C* | *1.425V +85° C* | *Margin (%)* | *1.5V +27° C* | *1.425V +85° C* | *Margin (%)* |
| *tWR* (ns) | 4.69 | 5.38 | **14.71** | 4.71 | 5.38 | **14.21** |
| *tRP* (ns) | 5.88 | 6.68 | **13.5** | 5.88 | 6.68 | **13.6** |
| *tRCD* (ns) | 8.84 | 10.02 | **13.3** | 8.87 | 10.02 | **13.03** |
| *tRTP* (ns) | 8.84 | 10.66 | **20.55** | 9.74 | 12.79 | **31.36** |

timing measures to integer clock cycles, to derive conservative measures for the timings. These derived conservative margins address the worst-case test requirement (4) in Section 3.6.1.

We employ these conservative margins (compensations for worst-case operating conditions) with the fastest timings observed using the characterization algorithms under nominal conditions to derive conservative yet optimal timings for a given device for worst-case operating conditions. Experiments deriving the same for several DRAM devices are presented in the next chapter. By overlapping these conservative yet optimal timings with the power-performance relation in Figure 3.5 (also derived for worst-case operating conditions), we can derive the expected power consumption estimates (including a $+1\sigma$ margin) for the given device. The sequence of operations described above is summarized in the flow-chart in Figure 3.6.

## 3.7 Related Work

When it comes to studying the impact of process variation in DRAMs, Intel observed performance degradation and power variation in DRAM memories in [59, 76]. However, their test mechanisms are not publicly available. Gottscho *et al.* in [45] also observed variations of around 15% in power consumption across several 1GB DIMMs from the same vendor and around 20% across different vendors. However, they did not establish the causes for the observed extent of power variations and did not test the DIMMs for variations in timings and performance.

Bathen *et al.* in [78] and [79] employed these observations and suggested memory mapping and partitioning solutions to exploit this variability, but also failed to analyze or exploit performance variations. Liu *et al.* in [123] observed longer data retention times in DRAMs than those suggested in datasheets, but did not extend this study to optimize DRAM timings. Desai *et al.* in [77] performed Monte-Carlo analysis on a single DRAM cell and basic circuit components to estimate the variation impact for an entire DRAM memory. They further proposed using adaptive body biasing to improve the yield of DRAMs. Although the variation estimates may be acceptable for the basic circuit components, such an

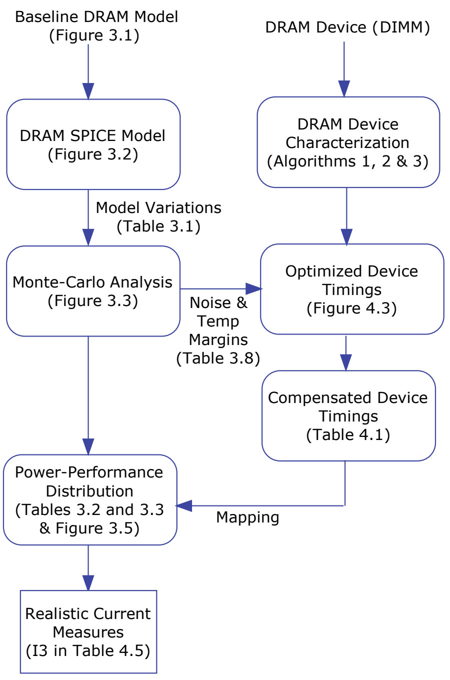

Baseline DRAM Model (Figure 3.1)
DRAM Device (DIMM)
DRAM SPICE Model (Figure 3.2)
DRAM Device Characterization (Algorithms 1, 2 & 3)
Model Variations (Table 3.1)
Monte-Carlo Analysis (Figure 3.3)
Optimized Device Timings (Figure 4.3)
Noise & Temp Margins (Table 3.8)
Compensated Device Timings (Table 4.1)
Power-Performance Distribution (Tables 3.2 and 3.3 & Figure 3.5)
Mapping
Realistic Current Measures (I3 in Table 4.5)

Figure 3.6: Deriving Optimized DRAM Timings and Realistic Current Measures

extrapolation to an entire DRAM is at best a coarse approximation.

Unfortunately, there are no known realistic models or studies that provide acceptable estimates of the expected impact of process variations on DRAM power consumption and performance and no variation data is made available by DRAM vendors, undermining the applicability of the solutions suggested in [59, 77–79]. In this chapter, we derive realistic estimates of the impact of variations on DRAM currents and latencies to enable use of such solutions.

When it comes to DRAM power estimation, none of the existing power models discussed in the last chapter, consider the impact of process variations on power consumption in DRAMs, due to lack of variation analysis and data. In this work, we provide possible distributions of the current measures for different DRAM operations, which can be employed with the JEDEC current measures-based power models, such as [17, 40, 68], to obtain more realistic DRAM power estimates.

In the context of DRAM timing and functionality testing, authors of [93] and [94] proposed industrial DRAM tests and DRAM fault models. However, they only employed default DRAM timings in their tests. JEDEC proposed $I_{DD}$ tests [103] for functional testing of the DRAM under worst-case conditions, but they also did not change the timing measures and only tested a few pre-selected rows in the memory. Memtest86 [109] also did not alter the DRAM timings and only verified if the DRAM accepts and correctly retains arbitrary set of data written to it. Additionally, its testing is not guaranteed to stress the memory, since it depends on the underlying processor/cache architecture.

Moreover, none of these tests considered testing all the timing margins introduced due to process, temperature or voltage variations. In this chapter, we proposed a comprehensive test methodology that varies all critical DRAM timings and tests the entire memory with different datasets and test sets under worst-case operating conditions.

## 3.8 Conclusions

In this chapter, we modified an NGSPICE-based circuit-level DRAM architecture and power model to include the impact of run-time and design-time variations and derived the effect on DRAM performance and power consumption on a $\pm 6\sigma$ scale using Monte-Carlo analysis. From this analysis, we identified *better than worst-case* current measures that are applicable for >97% of the DRAM devices of a given configuration, in place of worst-case datasheet measures. We also proposes a generic post-manufacturing power characterization methodology for DRAMs to derive optimized timings and *realistic* current estimates for a given DRAM device, using the Monte-Carlo analysis. In the next chapter, we verify if our estimates on variation margins hold true on a real DRAM DIMM.

# Chapter 4

# Verifying & Validating DRAMPower

Having discussed the DRAMPower model and the expected impact of variation of DRAM currents and performance, in this chapter, we aim to verify and validate both these claims against real power and performance measurements from hardware. Some of the experiments and results presented in this chapter have been published previously in our paper at DATE 2014 [54].

## 4.1 Introduction

To verify and validate our work, we perform two sets of experiments that:

1. Verify the impact of process variation (process margins) on DRAM currents and the four critical DRAM timing parameters ($nWR$, $nRP$, $nRCD$ and $nRTP$) against the expected impact from Monte-Carlo simulations and derive the fastest and compensated set of timings (using the performance characterization algorithms) at which a DRAM continues to operate successfully under worst-case conditions.

2. Validate DRAMPower and Micron's power model against real power measurements for different DRAM operations by employing (a) datasheet, (b) realistic (SPICE model-based) and (c) measured current values ($I_{DDs}$) as inputs to the two models.

Towards this, we employ 12 identical Micron DDR3-1066 64-bit 512MB single-rank SODIMMs (using 48 x16 DDR3 devices from batch MT4JSF6464HY-1G1B1) and first explore the variation in DRAM timing measures among them using the performance characterization algorithms. We then derive the variation in DRAM current measures in the fastest and slowest DIMMs identified among these 12 tested DIMMs and the impact of speeding up of the fastest DIMM on its current measures, since it represents the highest current consumption in the lot. Finally, we employ the slowest DIMM with the highest current margins (lowest current measures) and evaluate the accuracy of the two power models when employing current measures from datasheets to actual measures.

## 4.2 Experimental Setup

As stated earlier, we employ 12 Micron DDR3 DIMMs with the same configuration for these experiments. To perform these tests, we use Xilinx ML605 boards mounted with the said 512MB DDR3 DIMM module and program the FPGA to: (1) trigger the test algorithms through a MicroBlaze, (2) program the test patterns to a memory controller that uses the Xilinx PHY [110] to communicate to the DRAM, and (3) verify the dataset read back in the memory controller. The DIMM is operated at 400MHz instead of 533MHz, due to limitations of the Xilinx PHY [110]. However it does not have any impact, since the delays being tested are frequency-independent analog timings. However, for a fair comparison, we employ current measures for a DDR3-800 MHz as baseline (which are lower than frequency down-scaled measures of DDR3-1066 and hence more restrictive).

The power measurements are obtained by mounting the DDR3 DIMMs on a JEDEC MO-268 [114] standard-compatible, JET-5466 SODIMM extender board [115] equipped with a current-sensing shunt resistor of 100m$\Omega$, using a high-end Lecroy Wavesurfer 454 Oscilloscope (2GS/s) reporting at 500MHz. We employed two channels on the scope, with the two probes connected across the resistor and using a common ground. We used 1x probes for minimal signal loss. The difference between the measures of the two channels indicates the voltage drop observed. For average power measurements, we take the mean of the voltage drop over more than 100 iterations of the analysis window.

For a fair evaluation, we employ worst-case operating conditions as mandated by JEDEC for these tests. For this, we introduce: (1) maximum noise in the power supply with the help of the shunt resistance on the DDR3 extender board and reduce the supply voltage to 1.42V (maximum supported noise), (2) maximum temperature variations (by forcing a Case Temperature ($T_c$) of +85°C) using current-controlled Peltier elements (explained next) and (3) heavy DRAM usage (by continuously reading and writing from the DRAM for several hours without idling).

To introduce worst-case operating temperature, we locally heat the DRAM module to +85°C without overheating the rest of the ML605 platform. The experimental setup is depicted in Figure 4.1. A couple of small 4W Peltier elements (PEs) [111] are used to heat up two DRAM devices on a DIMM, while temperature sensors (a thermistor and a thermocouple) placed on top of the two DRAM devices provide feedback to keep them at the required temperature. The cold sides of the PEs are connected to a sufficiently large heat-sink to keep them at room-temperature. The hot sides are then connected to the DRAM modules using thermal paste. Both the current-controlled PEs are connected to a digital power supply unit. Two temperature sensors are placed between the PEs and the DRAM module: a pre-calibrated thermocouple for accurate tracking of the temperature and a thermistor used by a control loop to regulate the temperature, as specified by JEDEC [103] and Micron [112] standards.

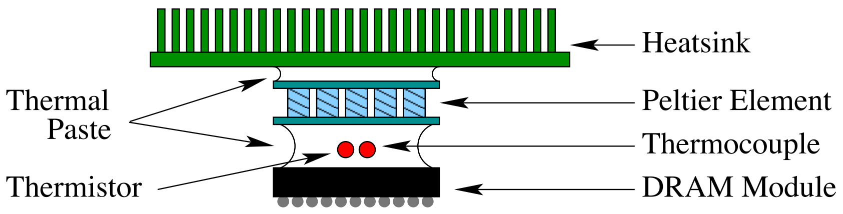


Figure 4.1: Heating Setup

Both the PEs and the thermistor are connected to an Arduino board [113] that reads the dynamic changes to the resistance value of thermistor (based on an amplified voltage drop measure) and implements a simple control loop, which in turn controls the power supply unit that heats up the PEs, thus regulating the temperature on the DRAM module. At the same time, the thermocouple reports accurate temperature measures through a digital multimeter for parallel monitoring of the temperature. Both the Peltier elements and the thermistor are calibrated off-line (before the test is initiated) against the reference thermocouple, which has a known temperature response.

Before starting the experiment, the DRAM module is heated up until it reaches the desired temperature (+85°C). The control loop implemented on the Arduino keeps the device temperature between +85°C and +86°C (+1°C range). If it goes above the temperature threshold the power supply to the PE is switched off, and when it drops below, it is turned on again. This temperature check and update is performed five times per second. The complete experimental setup is shown in Figure 4.2.

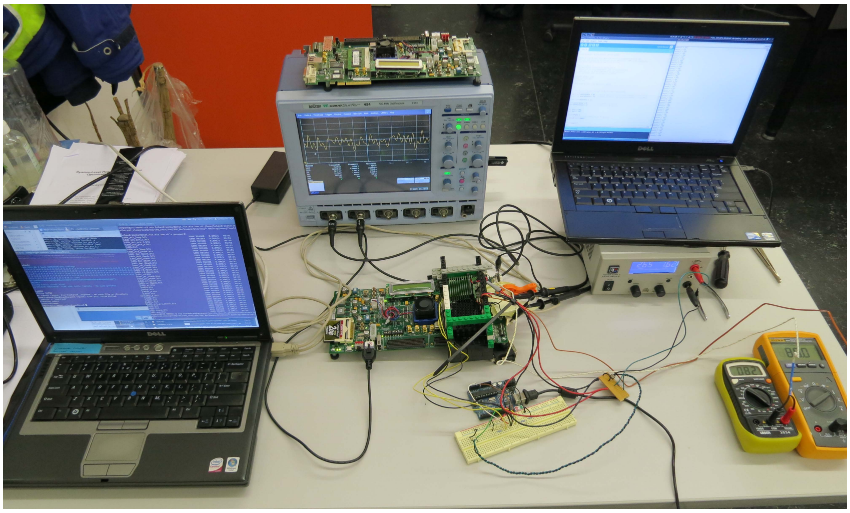


Figure 4.2: Experimental Setup

As can be noticed from the figure, the FPGA board is programmed using the laptop on the left. The Arduino board is continuously fed dynamic thermal resistance information from the thermistor placed on the top of one of the DDR3 modules on the DIMM via a voltage (drop) amplifier (on the breadboard). It forwards this information to the control loop, which in turn controls the output to the power-supply heating up the PEs.

The control loop also outputs its analysis via a display port to the laptop on the right. This control loop is calibrated based on the output of the pre-calibrated thermocouple. The multi-meters constantly read out the temperature on the DDR3 devices (from both the thermistor and thermocouple), while the scope measures the voltage drop across the 100mΩ shunt resistor on the DDR3 extender board. We use these voltage drop and resistance measures to compute the current consumption (I = V/R).

## 4.3 Experiments

In this section, we perform two sets of experiments on real hardware to verify the DRAM performance and power variations expected from the NGSPICE simulations. Additionally, we also perform experiments to validate DRAMPower by comparing its power estimates against real hardware measurements.

### 4.3.1 Verifying DRAM Performance Variations

In these experiments, we begin by evaluating and validating our proposed performance characterization methodology and derive the fastest set of timings for each of 48 the devices on all 12 DIMMs, under nominal test conditions (1.5V/+27°C).

We then identify the fastest DIMM, which represents maximum exploitation of timing margins by our methodology, add the derived requisite timing compensations for temperature and noise and verify if it continues to operate successfully under worst-case conditions (1.42V/+85°C). By validating our methodology for the most exploited DIMM, we show that it will work for the less exploited DIMMs as well. Besides, since the fastest DIMM is also likely to have the highest power consumption, we also show that we do not violate any of the original current measures despite speeding it up but instead, further reduce total energy consumption significantly.

#### Experiment I: Fastest Timings

We employ the performance characterization algorithms to derive the fastest timings for the 48 DDR3 devices tested under nominal conditions. The results for the 48 DDR3 devices (from 12 DIMMs) are presented in Figure 4.3. As can be noticed, the datasheet (DS) measures are very pessimistic compared to the actual measures (under nominal conditions) for all the devices tested. For a fair comparison, we must employ similar worst-case conditions as used for the datasheet measures, which is presented in the next experiment.

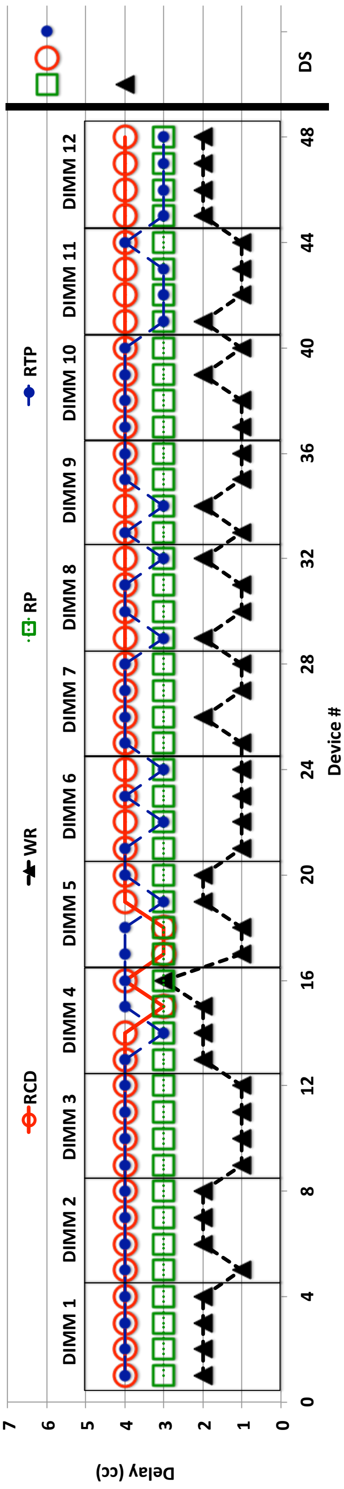


Figure 4.3: Actual vs. Datasheet (DS) Timing Measures - Nominal Conditions

## Experiment II: Timing Compensation and Verification

For these experiments, we add the derived timing margin compensations for noise and temperature (from Table 3.8) to the fastest DIMM (#6), which was most exploited by our method in Experiment I. To identify this DIMM, we summed up the four critical timing parameters for all four devices on each DIMM and selected the one with the lowest sum. We employ the resulting compensated timing measures (rounded-up to integer clock cycle measures and presented in Table 4.1) and verify if they hold true under worst-case conditions.

We first programmed the non-compensated test patterns with the actual fastest (nominal-condition) timing measures of this DIMM on to the MicroBlaze and DRAM controller and triggered the memory test in Algorithm 1. We observed that the test failed immediately, since these measures do not compensate for power-supply noise or temperature (as shown in Table 4.1).

We then programmed the test patterns with the rounded-up compensated timing measures and observed that the device worked correctly without any issues. We continued the test un-interrupted for over 4 hours under the worst-case operating conditions, transferring over 16TB of data to and from the memory in the process. At the end the test finished successfully, as depicted in Table 4.1, thereby verifying the validity of the conservative timing measures derived based on temperature and noise compensations.

Table 4.1: Test Measures and Results

| *Test Type* | *nRCD (cc)* | *nRP (cc)* | *nRTP (cc)* | *nWR (cc)* | *Result* |
|---|---|---|---|---|---|
| Fastest | 4 | 3 | 4 | 1 | FAIL |
| Margins | 13.3% | 13.5% | 20.5% | 14.7% | - |
| Compensated | 5 | 4 | 5 | 2 | PASS |

## Experiment III: Evaluating Performance Benefits

Having tested the 12 DIMMs and verified the conservative timing measures for the fastest DIMM, in this experiment, we evaluate the performance benefits of employing our proposed performance characterization methodology on the fastest (#6) and the slowest (#1) DIMMs. In Tables 4.2 and 4.3, we present the impact on read and write latency and bandwidth, when employing the compensated (for worst-case conditions) and un-compensated (for nominal conditions) timings for the smallest access granularity (64 bytes) for the 64-bit DIMM. The read/write latency measures represent the latency for activating, reading/writing and precharging a bank row. The read/write bandwidth measures indicate the net bandwidth observed in the read/write latency period (64 bytes/latency). As can be observed, under worst-case operating conditions, the fastest DIMM achieved around 61.5% and 35% improvement in read and write bandwidths, and around

38% and 25.9% improvement in read and write latency, respectively. The slowest DIMM also achieved significant improvements, including 61.5% and 28.5% improvements in read and write bandwidths and 38% and 22.2% improvement in read and write latencies.

Table 4.2: Impact on Read Latency and Bandwidth

| | Read Latency (cc) | Latency Reduction (%) | Read Bandwidth (GBps) | Bandwidth Increase (%) |
|---|---|---|---|---|
| Original | 21 | - | 1.22 | - |
| Fastest_Comp | 13 | 38.10 | 1.97 | 61.54 |
| Fastest_Nom | 10 | 52.38 | 2.56 | 110.00 |
| Slowest_Comp | 13 | 38.10 | 1.97 | 61.54 |
| Slowest_Nom | 11 | 47.62 | 2.33 | 90.91 |

Table 4.3: Impact on Write Latency and Bandwidth

| | Write Latency (cc) | Latency Reduction (%) | Write Bandwidth (GBps) | Bandwidth Increase (%) |
|---|---|---|---|---|
| Original | 27 | - | 0.95 | - |
| Fastest_Comp | 20 | 25.93 | 1.28 | 35.00 |
| Fastest_Nom | 17 | 37.04 | 1.51 | 58.82 |
| Slowest_Comp | 21 | 22.22 | 1.22 | 28.57 |
| Slowest_Nom | 18 | 33.33 | 1.42 | 50.00 |

These results show the pessimism in DRAM datasheet timings and the importance of our DRAM characterization methodology. Also, if nominal operating conditions are maintained, these gains increase significantly.

## 4.3.2 Verifying DRAM Power Variations

In these experiments, we first derive a set of standard current measures specified by JEDEC ($I_{DDs}$) for the fastest and slowest DIMMs identified in the previous experiments using the default datasheet timings. By employing the fastest and slowest DIMMs, we cover both the extremes in terms of power consumption for a fair evaluation. The standard JEDEC current measures correspond to different combinations of memory operations and are obtained using standardized measurement test loops [103] under worst-case run-time operating conditions. We compare these measures to the nominal and better than worst-case current estimates derived by the NGSPICE model and the datasheet measures. Next, we also observe the impact of using the compensated timings with the fastest DIMM on its current measures, since shrinking the timing measures may increase them drastically. We chose the fastest DIMM to show the worst-case impact of our methodology.

We show that although the current measures increase, they are still much lower than the datasheet measures for the fastest DIMM. We also show that despite this increase in power consumption, we achieve significant savings in terms of energy, due to the shortened latencies.

## Experiment IV: Measuring $I_{DD}$ currents

We performed a set of eleven current measurements under worst-case operating conditions to obtain the JEDEC standard $I_{DD}$ currents for the fastest and slowest DDR3 DIMMs. We present the screen-shots from the Oscilloscope for the slowest DIMMs and list all $I_{DD}$ measures for both these DIMMs in Table 4.4.

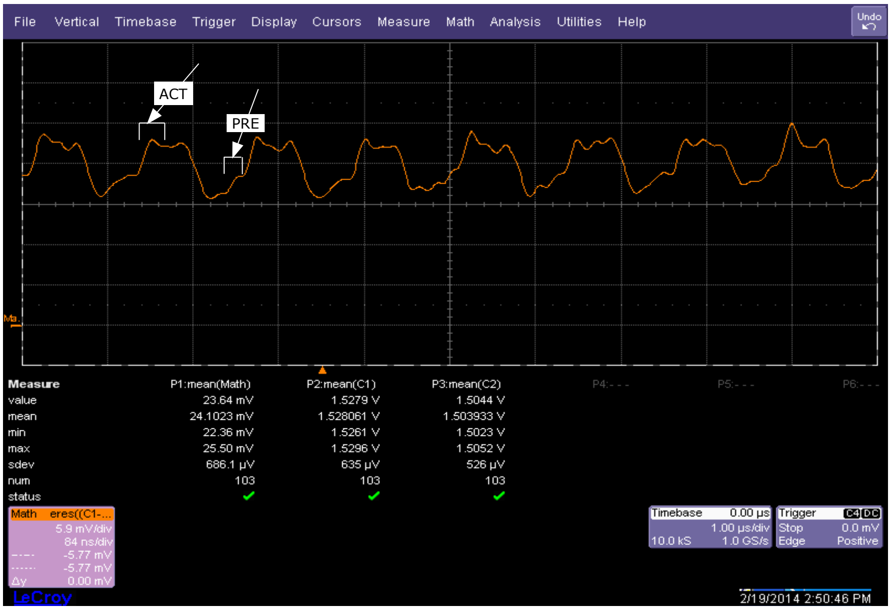
We also compare the measured $I_{DDs}$ to datasheet measures and NGSPICE based nominal and better than worst-case current measures. We classify these current measures as complete and partial currents. Complete currents are measured over complete DRAM operations and partial currents are measured for specific DRAM operations which may be triggered using additional DRAM operations. These partial currents are filtered out by repeating the concerned operation several times and then using density plots, which highlight the current measure corresponding to that operation. The current measures can be obtained using the the mean voltage drop value in (1) column P1 for the complete currents and (2) the Math ERES (Enhanced Resolution) box on the lower left hand corner for the partial currents. In this section, we show the plots for one Complete current: $I_{DD0}$ and one Partial current: $I_{DD2P}$. The measurement plots for the other $I_{DD}$ currents are shown in Appendix A. A summary of all the current measurements is shown in Table 4.4.

**[1] $I_{DD0}$ current - Complete Current**

The $I_{DD0}$ test loop measures current across one ACT and PRE command combination sent to one bank, with the other banks retained in precharged state and the test looping over all banks one after the other. The time period between an ACT and PRE command is maintained as *nRAS* cycles. *nRC* cycles after the ACT command, the test switches to the next bank and repeats itself over all the banks in the DRAM, with one bank active at a time. Since this test repeats itself without requiring any additional operations, the measurement observed is a complete current and shown in Figure 4.4.

**[2] $I_{DD2P}$ current - Partial Current - Requires Precharging**

The $I_{DD2P}$ test loop measures current in the precharged power-down state. This test starts with a PREA command (observable as a peak at the beginning of the plot in Figure 4.5) and then a power-down is initiated. The time period between the PREA and power-down commands is maintained as *nRP* cycles to allow all the banks to be precharged. The DRAM is retained in the power-down state for 2000 clock cycles (sufficiently long compared to precharging time and power-up time). Retaining the DRAM in the power-down state for a long duration helps identify the power consumption of the power-down state in density plots. After this period, a power-up is issued and the test pattern ends after further *nXPDLL* clock cycles, power-up exit latency (short peak period at the end of the

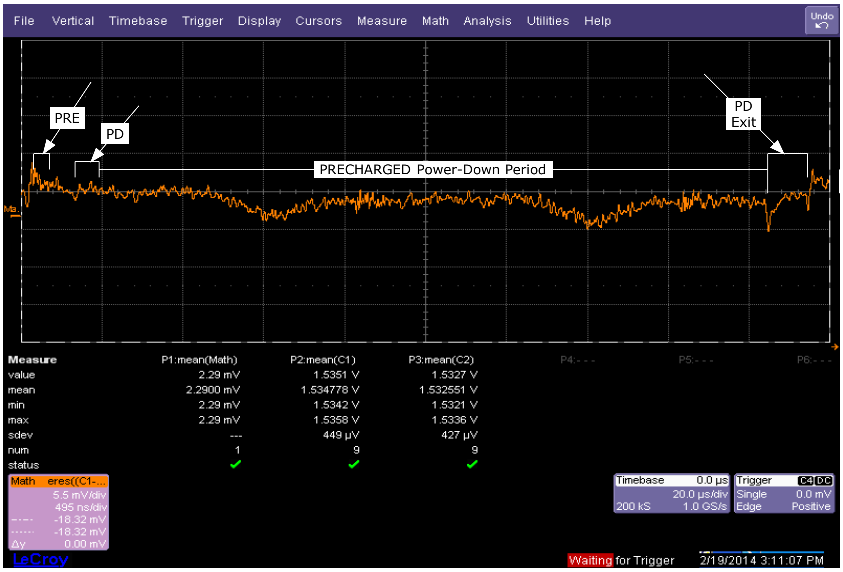


```
Measure   P1:mean(Math)   P2:mean(C1)   P3:mean(C2)   P4:---   P5:---   P6:---
value          23.64 mV      1.5279 V      1.5044 V
mean         24.1023 mV    1.528061 V    1.503933 V
min            22.36 mV      1.5261 V      1.5023 V
max            25.50 mV      1.5296 V      1.5052 V
sdev           686.1 µV        635 µV        526 µV
num                 103           103           103
status                ✓             ✓             ✓

Math  eres((C1-...        Timebase     0.00 µs   Trigger     C4 DC
       5.9 mV/div                   1.00 µs/div  Stop       0.0 mV
        84 ns/div         10.0 kS      1.0 GS/s  Edge     Positive
----   -5.77 mV
------ -5.77 mV
Δy      0.00 mV                                 2/19/2014 2:50:46 PM
```

Figure 4.4: Measuring $I_{DD0}$

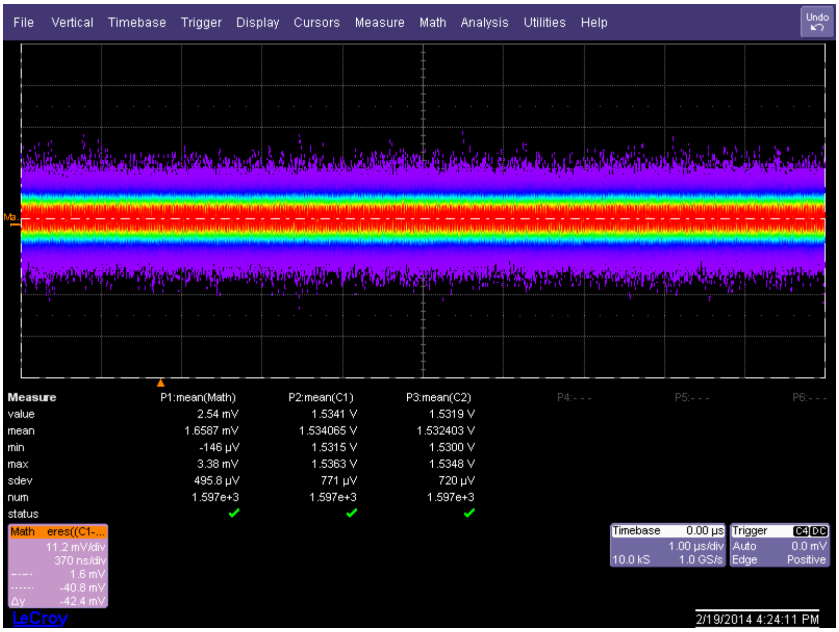
plot). The test pattern then repeats itself in a loop (iteration shown in Figure 4.5. Using the density plot shown in Figure 4.6, the voltage drop corresponding to the power-down state can be observed by the dotted line across the region in red (highest density identifier) and is reported by the 3rd measure in the Math ERES (Enhanced Resolution) results on the bottom left of Figure 4.6. The PREA current in Figure 4.5 is not very high since all banks are already precharged at the end of one test loop.

Similarly, we observed all the different $I_{DD}$ measures for the fastest and slowest 512MB DDR3-800 DIMMs identified in the previous experiment (plots for the fastest DIMM in Appendix A). In Table 4.4, we compare these measured currents to the nominal and better than worst-case measures derived using the SPICE simulations and the datasheet measures. Note that the datasheet current measures from Micron do not include I/O power consumption, however, the real measurements on the JET-5466 board do. For a fair comparison, we resolve this by employing the Micron power calculator [17] estimates for I/O power consumption and subtracting them from the real measurements for a fair comparison. The SPICE model also does not consider I/O power consumption. In this table, $I_{DD1W}$ is not a JEDEC standard measure, however, its reference measures can be calculated by substituting write current ($I_{DD4W}$) instead of read current ($I_{DD4R}$) in $I_{DD1}$ current and corresponds to activation-write-precharge current. As can be observed from the results, the current measures for both the fastest and the slowest DIMMs are significantly lower than the datasheet measures.



```
Measure   P1:mean(Math)   P2:mean(C1)   P3:mean(C2)   P4:- - -   P5:- - -   P6:- - -
value     2.29 mV         1.5351 V      1.5327 V
mean      2.2900 mV       1.534778 V    1.532551 V
min       2.29 mV         1.5342 V      1.5321 V
max       2.29 mV         1.5358 V      1.5336 V
sdev      ---             449 µV        427 µV
num       1               9             9
status    ✓               ✓             ✓

Math  eres((C1-...
      5.5 mV/div
      495 ns/div
      -18.32 mV
      -18.32 mV
Δy    0.00 mV

Timebase   0.0 µs      Trigger   C4 DC
           20.0 µs/div Single    0.0 mV
200 kS     1.0 GS/s    Edge      Positive

Waiting for Trigger      2/19/2014 3:11:07 PM
```

Figure 4.5: Measuring $I_{DD2P}$ after Precharging



```
Measure   P1:mean(Math)   P2:mean(C1)   P3:mean(C2)   P4:- - -   P5:- - -   P6:- - -
value     2.54 mV         1.5341 V      1.5319 V
mean      1.6587 mV       1.534065 V    1.532403 V
min       -146 µV         1.5315 V      1.5300 V
max       3.38 mV         1.5363 V      1.5348 V
sdev      495.8 µV        771 µV        720 µV
num       1.597e+3        1.597e+3      1.597e+3
status    ✓               ✓             ✓

Math  eres((C1-...
      11.2 mV/div
      370 ns/div
      1.6 mV
      -40.8 mV
Δy    -42.4 mV

Timebase   0.00 µs     Trigger   C4 DC
           1.00 µs/div Auto      0.0 mV
10.0 kS    1.0 GS/s    Edge      Positive

2/19/2014 4:24:11 PM
```

Figure 4.6: Measuring $I_{DD2P}$ - Density Plot

In the case of the slowest DIMM, this difference ranges from 21% in the case of $I_{DD1R}$ current to 65% in the case of $I_{DD6}$ self-refresh current. Employing the nominal (average) or better than worst-case current measures derived from the SPICE simulations on the circuit-level model would provide better values of currents than the worst-case measures. The difference between the nominal and measured values is only about 13% for $I_{DD1R}$ current and 25% for $I_{DD6}$ current for the slowest DIMM and 6% for $I_{DD1R}$ current and 25% for $I_{DD6}$ current for the fastest DIMM (from the evaluated population).

Table 4.4: DIMM Current Measures under worst-case operating conditions

| *Range* → *Current* ↓ | *Fastest (I4) (-5σ) mA* | *Slowest (I4) (-6σ) mA* | *Nominal (I1) (μ) mA* | *Better than Worst-Case (+3σ) (I2) mA* | *Datasheet (I0) (+5σ) mA* | *Slowest vs. DS %* |
|---|---|---|---|---|---|---|
| $I_{DD0}$ | 263 | 241 | 320 | 343 | 360 | -33 |
| $I_{DD1R}$ | 376 | 346 | 400 | 428 | 440 | -21 |
| $I_{DD1W}$ | 320 | 293 | 335 | 360 | 410 | -28 |
| $I_{DD2N}$ | 110 | 101 | 140 | 164 | 180 | -43 |
| $I_{DD2P}$ | 17 | 16 | 20 | 30 | 40 | -60 |
| $I_{DD3N}$ | 116 | 107 | 150 | 177 | 200 | -46 |
| $I_{DD3P}$ | 39 | 38 | 48 | 63 | 100 | -62 |
| $I_{DD4R}$ | 584 | 535 | 722 | 790 | 840 | -36 |
| $I_{DD4W}$ | 599 | 549 | 730 | 793 | 840 | -34 |
| $I_{DD5}$ | 477 | 462 | 717 | 766 | 800 | -42 |
| $I_{DD6}$ | 8.4 | 8.4 | 11.2 | 16.6 | 24 | -65 |

### Experiment V: Side-Effects of Scaling Timings

Based on the performance-variation results, if we shrink the timing measures to reflect a DRAM's actual performance, we risk increasing the DRAM power consumption. However, since we are only exploiting process-margins in the timings, the current measures should *NOT* go higher than the worst-case datasheet current values, since the current measures also include appropriate process-margins.

By overlapping the measured performance metrics (including temperature and noise margins) with the power-performance relation obtained in Figure 3.5, we observe that the performance of this particular DIMM (fastest) ranks around the $+1\sigma$ mark. As suggested earlier, we conservatively estimate the power consumption at $+2\sigma$ from the SPICE simulations.

We then verify the $+2\sigma$ SPICE estimates to the actual power consumption measured by using these optimized timings. The currents affected by our optimization of the four critical analog timings include: (1) activation-precharge current ($I_{DD0}$) [103], (2) activation-read-precharge current ($I_{DD1}$) [103], and (3) activation-write-precharge current ($I_{DD1W}$). As before, we subtrace the estimates for I/O power consumption from the real measurements for a fair comparison.

The impact on current measures for the fastest (most exploited) DIMM are presented in Table 4.5. The results show that the current measures, despite shrinking the analog delays, remain lower than the datasheet and the conservative $+2\sigma$ SPICE estimates. This is as expected, since we are not violating any timing margins for noise and temperature or any of the actual delays associated with these operations. We are merely conservatively identifying the excessive process-margins associated with these delays and eliminating them partially depending on the actual impact on a given DRAM device.

Table 4.5: Impact on Current Measures

| *Current Type* | *Measured (I4) (mA)* | *SPICE ($+2\sigma$) (I3) (mA)* | *Datasheet (I0) (mA)* |
|---|---|---|---|
| $I_{DD0}$ | 314 | 330 | 360 |
| $I_{DD1}$ | 396 | 410 | 440 |
| $I_{DD1W}$ | 320 | 340 | 410 |

### 4.3.3 Verifying and Validating DRAMPower

In this section, we present two sets of experiments to highlight the accuracy of the power and energy estimates of the DRAMPower model and tool. For each experiment, we present the power consumption estimates of DRAMPower and Micron's model when using the measured $I_{DDs}$ (from the tested DIMM) as inputs to both the models for a fair comparison and compare them to real measurements from hardware.

In the first set of experiments, we scale the number of activations and reads to analyze the differences in their modeling of the active DRAM operations and the corresponding impact on power. Since varying the number of writes and degree of bank-interleaving have a similar scaling impact, we present those experiments in Appendix A. In the second set of experiments, we observe the power impact of state transitions from active to power-down and self-refresh modes, and analyze the differences in their modeling of DRAM power-saving modes and state transitions. Transitions to idle mode and refreshes are also presented in Appendix A, since they have a similar impact as transition into power-down.

For all experiments, we employed directed micro-benchmarks that reflect all possible combinations of DRAM commands and operations. We decided to employ these micro-benchmarks since it was easier to continuously run them over in loops and measure voltage drop averages over time. The accuracy of power estimates can be expected to be the same for the different operations under realistic workloads as well.

## Experiment VI: Verifying scaling of power

DRAMPower employs the actual timings between commands obtained from a given DRAM command trace (in place of the minimal timings from DRAM datasheets) and performs cycle-accurate power estimation with high-precision modeling of different DRAM operations under any degree of bank interleaving.

To stress the importance of accurate scaling, we first show scaling of ACT-PRE operations across two different degrees of bank interleaving. We then show the impact of issuing multiple bursts of read commands on the corresponding power consumption, including transactions with multi-bank interleaving. With these experiments, we show the importance of precisely modeling scaling of DRAM timings and support for multi-bank interleaving.

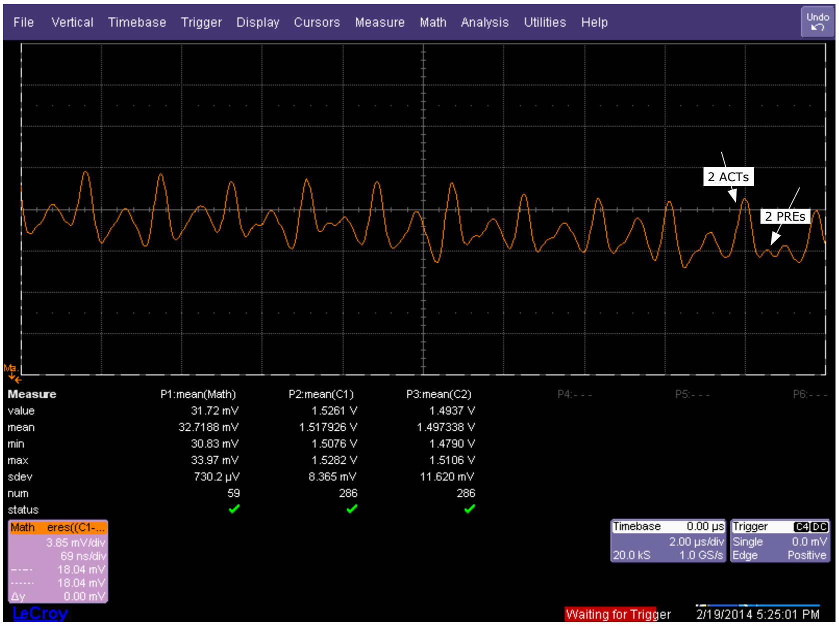
In these 4 experiments, we achieve significantly better accuracy compared to Micron's estimates, as will be shown later in Table 4.6. Similar impact of writes and more degrees of bank-interleaving are presented in Appendix A.

[1] **2 Banks - ACT-PRE**: We performed the two-bank activate-precharge operations to highlight the importance of the need for accurate scaling in Micron's power modeling, as represented by the current measures in Figure 4.7. The test is looped over several times.

In this case, the average power consumption is expected to be slightly lower than two times that of a single bank ACT-PRE. This is due to *nRRD* delay, which is imposed as a restriction (minimum timing constraint) between successive ACT commands to different banks, which increases the transaction length. On the other hand, Micron's model estimates this power measure to be exactly twice of that of a single-bank ACT-PRE, ignoring the effect of the *nRRD* delay. This is addressed by accurate scaling shown in Equations (2.2) and (2.3) in Chapter 2.

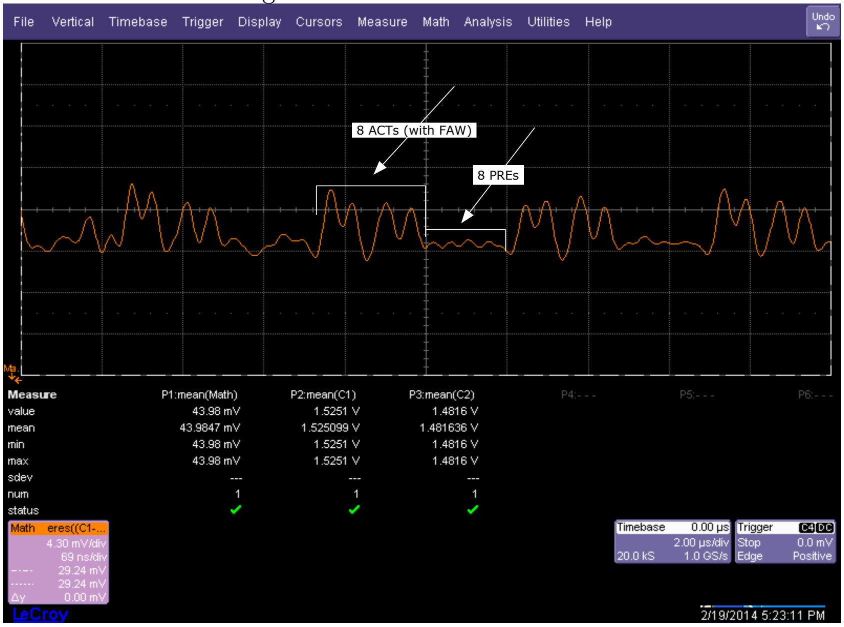
[2] **8 Banks - ACT-PRE**: We also performed the eight-bank activate-precharge operations to further emphasize importance of the need for accurate scaling, as represented by the current measures in Figure 4.8. In this case, the average power consumption is expected to be closer to that of the four-bank ACT-PRE. This is due to *nFAW* delay, which in this case for this particular DRAM, keeps first and last four ACT commands in separate groups over time without overlapping each other. This is true by design, to keep the DRAM power consumption low. As a result, the power consumption averages out to be similar to that of the four-bank ACT-PRE. On the other hand, Micron's model estimates this power measure to be eight times of that of a single-bank ACT-PRE.

[3] **4 Reads - 1Bank**: We performed the four successive Read operations to a bank including the corresponding ACT-PRE operations to stress the importance of power modeling of Read transactions, as shown by the current measures in Figure 4.9. In this case, the average power consumption is expected to be slightly lower than the sum of four Read operations and that of a single bank ACT-PRE. This is due to the total latency of the transaction, which can be calculated as the sum of *nRCD* delay after the ACT command leading up to the first Read, *nRTP* delay after the last Read before a PRE is issued, the *nRP* delay for the



```
Measure   P1:mean(Math)  P2:mean(C1)  P3:mean(C2)   P4:- - -   P5:- - -   P6:- - -
value     31.72 mV       1.5261 V     1.4937 V
mean      32.7188 mV     1.517926 V   1.497338 V
min       30.83 mV       1.5076 V     1.4790 V
max       33.97 mV       1.5282 V     1.5106 V
sdev      730.2 µV       8.365 mV     11.620 mV
num       59             286          286
status    ✓              ✓            ✓

Math  eres((C1-...     Timebase   0.00 µs    Trigger     C4 DC
      3.85 mV/div                2.00 µs/div Single     0.0 mV
      69 ns/div        20.0 kS   1.0 GS/s    Edge       Positive
----  18.04 mV
----- 18.04 mV
Δy    0.00 mV

Waiting for Trigger   2/19/2014 5:25:01 PM
```

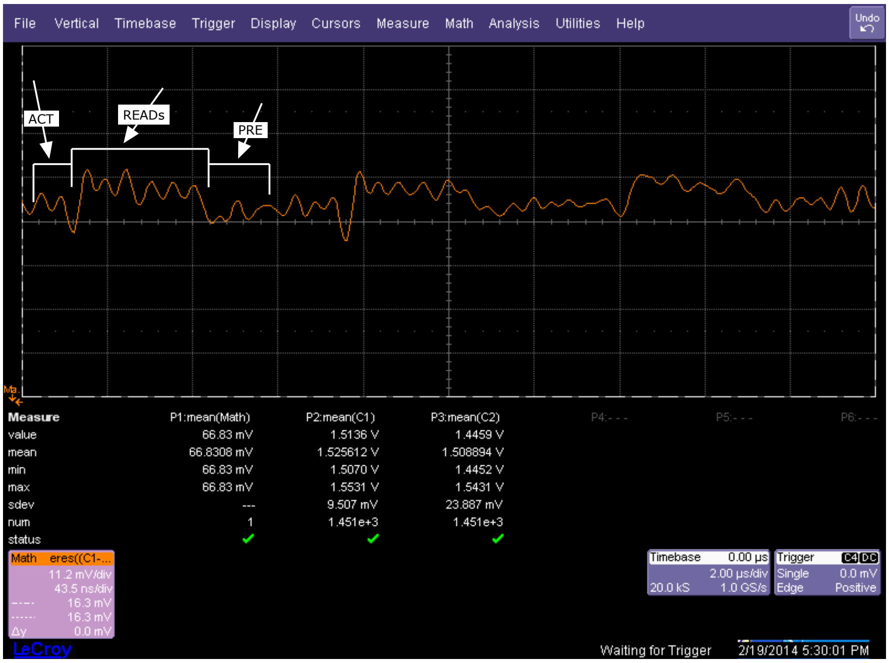
Figure 4.7: 2 Banks - ACT-PRE



```
Measure   P1:mean(Math)  P2:mean(C1)  P3:mean(C2)   P4:- - -   P5:- - -   P6:- - -
value     43.98 mV       1.5251 V     1.4816 V
mean      43.9847 mV     1.525099 V   1.481636 V
min       43.98 mV       1.5251 V     1.4816 V
max       43.98 mV       1.5251 V     1.4816 V
sdev      ---            ---          ---
num       1              1            1
status    ✓              ✓            ✓

Math  eres((C1-...     Timebase   0.00 µs    Trigger     C4 DC
      4.30 mV/div                2.00 µs/div Stop       0.0 mV
      69 ns/div        20.0 kS   1.0 GS/s    Edge       Positive
----  29.24 mV
----- 29.24 mV
Δy    0.00 mV

2/19/2014 5:23:11 PM
```

Figure 4.8: 8 Banks - ACT-PRE

PRE to complete and successive $nCCD$ delays between the Read commands. This overall latency is expected to be longer than $nRC$ cycles, which is assumed as the transaction length by Micron's model. This is addressed by Equations (2.2), (2.3) and (2.7) in Chapter 2, which scale timing appropriately.



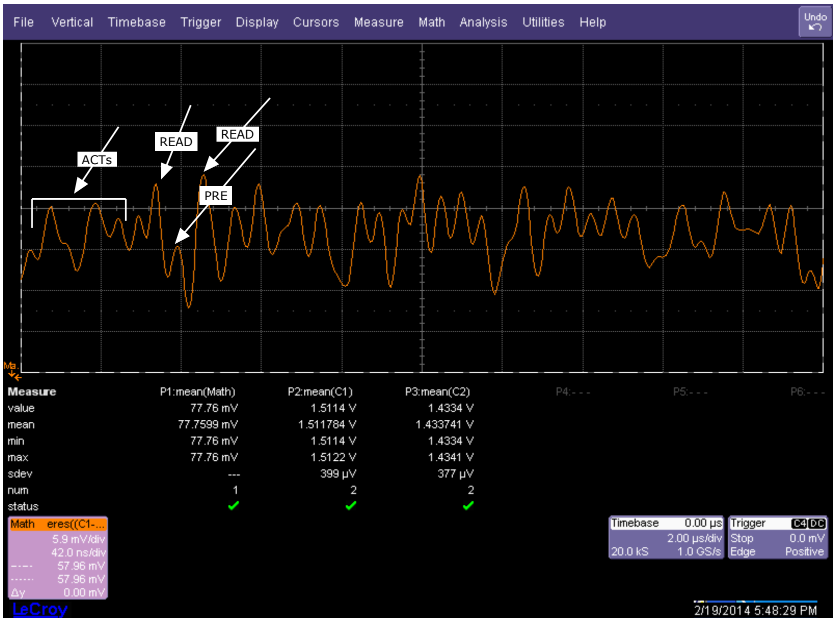
```
Measure   P1:mean(Math)  P2:mean(C1)  P3:mean(C2)  P4:---  P5:---  P6:---
value     66.83 mV       1.5136 V     1.4459 V
mean      66.8308 mV     1.525612 V   1.508894 V
min       66.83 mV       1.5070 V     1.4452 V
max       66.83 mV       1.5531 V     1.5431 V
sdev      ---            9.507 mV     23.887 mV
num       1              1.451e+3     1.451e+3
status    ✓              ✓            ✓

Math  eres((C1-...           Timebase   0.00 µs    Trigger   C4 DC
      11.2 mV/div                      2.00 µs/div Single    0.0 mV
      43.5 ns/div            20.0 kS   1.0 GS/s    Edge      Positive
      16.3 mV
      16.3 mV
Δy    0.0 mV

Waiting for Trigger   2/19/2014 5:30:01 PM
```

Figure 4.9: 4 Reads - 1 Bank

[4] **1 Read - 4 Banks**: We performed the four-bank activate-precharge operations combined with a Read operation on each bank to emphasize accurate scaling of both sets of operations, as represented by the current measures in Figure 4.10.

In this case, the average power consumption is expected to be significantly lower than the sum of four Read operations and that of four single bank ACT-PRE commands. This is due to the total latency of the transaction, which can be calculated as the sum of $nRCD$ delay after the first ACT command leading up to the first Read, $nRTP$ delay after the last Read (to the fourth bank) before a PRE is issued, the $nRP$ delay for the last PRE to complete, the $nRRD$ delay between the four successive ACT commands and the successive $nCCD$ delays between the four Read commands. This overall latency is expected to be much longer than $nRC$ cycles, which is assumed as the transaction length by Micron's model to estimate this power measure, ignoring the $nRRD$ delays.

These experiments show the importance of accurate scaling of power measures, which is achieved by DRAMPower unlike Micron's model. This will be evident from the power estimates for the two models for these experiments, as presented later in Table 4.6.



| Measure | P1:mean(Math) | P2:mean(C1) | P3:mean(C2) | P4:- - - | P5:- - - | P6:- - - |
|---|---|---|---|---|---|---|
| value | 77.76 mV | 1.5114 V | 1.4334 V | | | |
| mean | 77.7599 mV | 1.511784 V | 1.433741 V | | | |
| min | 77.76 mV | 1.5114 V | 1.4334 V | | | |
| max | 77.76 mV | 1.5122 V | 1.4341 V | | | |
| sdev | --- | 399 µV | 377 µV | | | |
| num | 1 | 2 | 2 | | | |
| status | ✓ | ✓ | ✓ | | | |

```
Math   eres((C1-...
         5.9 mV/div
        42.0 ns/div
----     57.96 mV
......   57.96 mV
Δy        0.00 mV
```

```
Timebase     0.00 µs   Trigger     C4 DC
           2.00 µs/div Stop       0.0 mV
20.0 kS      1.0 GS/s  Edge     Positive
```

2/19/2014 5:48:29 PM

Figure 4.10: 1 Read - 4 Banks

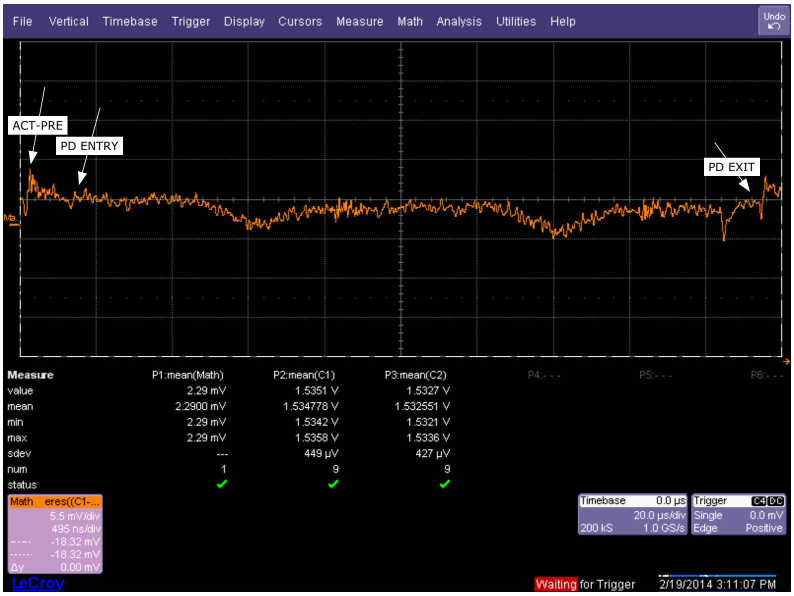
## Experiment VII: Verifying DRAM state transitions

DRAMPower performs high-precision modeling of DRAM power saving states and state transitions into the low power states. In the following set of experiments, we employ transactions with transitions to different states (power-down and self-refresh) and obtain the power consumption value from real hardware measurements, to compare against DRAMPower's estimates. Transitions into idle and refresh states are shown in Appendix A.

To underline the importance of modeling state transitions, we perform experiments that involve significant transitions from: (1) ACT-PRE commands into precharged power-down state and then powering back up, and (2) ACT-PRE commands into Self-refresh state and powering back up. With these experiments, we show the importance of precise modeling memory state transitions. In these 2 experiments as well, we achieve significantly better accuracy compared to Micron's estimates, as will be shown later in Table 4.6, at the end of this section.

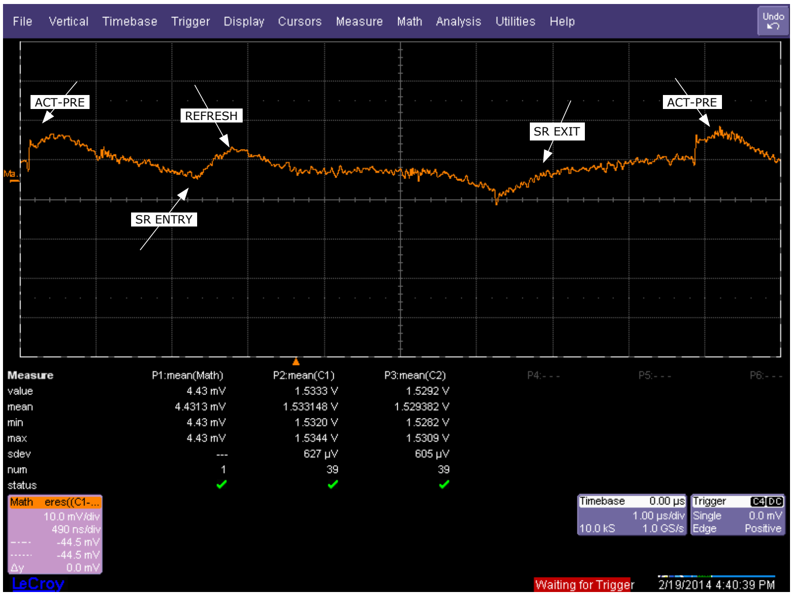
[5] **ACT - PRE to Precharged Power-Down**: We performed the experiment to transition from a single-bank activate-precharge operation into the precharged powered-down state, as represented by Figure 4.11. In this case, it is important to note that the power consumption reduces gradually as we enter or exit the precharged power-down state. This transition must be captured in the power model based on the duration for which the clock is gated and the time required to power-up the memory back into a precharged idle (standby) state.

Micron's model does not model these transitions. In our power model, we address these transitions as shown in Equations (2.9) and (2.10) in Chapter 2.


File Vertical Timebase Trigger Display Cursors Measure Math Analysis Utilities Help
Undo
ACT-PRE
PD ENTRY
PD EXIT
LeCroy

```
Measure   P1:mean(Math)  P2:mean(C1)  P3:mean(C2)  P4:---  P5:---  P6:---
value          2.29 mV     1.5351 V     1.5327 V
mean         2.2900 mV   1.534778 V   1.532551 V
min            2.29 mV     1.5342 V     1.5321 V
max            2.29 mV     1.5358 V     1.5336 V
sdev               ---     449 µV       427 µV
num                  1          9            9
status               ✓          ✓            ✓

Math  eres((C1-...     Timebase   0.0 µs   Trigger    C4 DC
      5.5 mV/div             20.0 µs/div   Single    0.0 mV
      495 ns/div       200 kS  1.0 GS/s    Edge    Positive
-·-·-  -18.32 mV
------ -18.32 mV
Δy      0.00 mV
                       Waiting for Trigger   2/19/2014 3:11:07 PM
```

Figure 4.11: ACT - PRE to Precharged Power-Down


File Vertical Timebase Trigger Display Cursors Measure Math Analysis Utilities Help
Undo
ACT-PRE
REFRESH
SR EXIT
ACT-PRE
SR ENTRY
LeCroy

```
Measure   P1:mean(Math)  P2:mean(C1)  P3:mean(C2)  P4:---  P5:---  P6:---
value          4.43 mV     1.5333 V     1.5292 V
mean         4.4313 mV   1.533148 V   1.529382 V
min            4.43 mV     1.5320 V     1.5282 V
max            4.43 mV     1.5344 V     1.5309 V
sdev               ---     627 µV       605 µV
num                  1         39           39
status               ✓          ✓            ✓

Math  eres((C1-...     Timebase   0.00 µs  Trigger    C4 DC
      10.0 mV/div             1.00 µs/div  Single    0.0 mV
      490 ns/div       10.0 kS  1.0 GS/s   Edge    Positive
-·-·-  -44.5 mV
------ -44.5 mV
Δy      0.0 mV
                       Waiting for Trigger   2/19/2014 4:40:39 PM
```

Figure 4.12: ACT - PRE to Self-Refresh

[6] **ACT - PRE to Self-Refresh**: We performed an experiment to transition from a single-bank activation-precharge operation into the self-refresh mode, as represented by the current measures in Figure 4.12.

In this case, it is important to note that an auto-refresh is instantiated when transitioning into a self-refresh. The power consumption due to this auto-refresh can be significant, depending on the duration of the self-refresh period.

This transition must be captured in the power model depending on the duration for which the clock is gated and stopped and the time required to power-up the memory back into a precharged idle (standby) state, as shown in Equation (2.16) in Chapter 2. Micron's model does not model these transitions into and out of a self-refresh operation.

## Summary

We employed the measured $I_{DDs}$ (I4) with both the high-level DRAM power models (M0 and M2) and observe their relative accuracy in the 17 DRAM operation tests (11 are shown in Appendix A) and compared the power estimates to real hardware measurements ($O_{ref}$) in Figure 4.13. As evident from Figure 4.13, DRAMPower performs much better compared to Micron's power model in all cases, with an average of 97% accuracy.

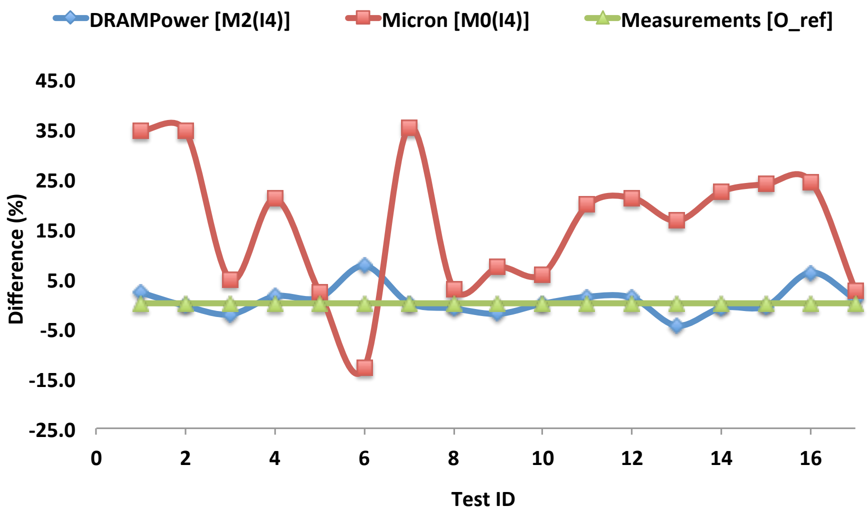


Figure 4.13: Accuracy of DRAMPower & Micron Model using Measured $I_{DDs}$

In case of self-refresh test (#6), the accuracy drops by 8%. This can be attributed to the fact that our model assumes a digital implementation of the DLL in the DIMM. However, the particular MICRON DIMM tested shows characteristics of an analog implementation of DLL [119], as evident from the gradual drop in power consumption in Figure 4.12 when the clock is gated or turned-off. The power estimation accuracy improves with longer self-refresh periods.

In Table 4.6, we present the measured power values for the 17 experiments and the corresponding estimates from DRAMPower (M2) and Micron's model (M0), when using datasheet (DS) $I_{DDs}$ (I0), measured $I_{DDs}$ (I4) and nominal $I_{DDs}$ (I1) as inputs to the two models. For each of these estimates, we also provide the error % in brackets compared to the measured reference. At the end, we present the average accuracy measure for each of these model and input currents combinations over the 17 experiments. As observed, DRAMPower (M2) when employing measured currents $I_{DDs}$ (I4) achieved an accuracy of 97% on average, compared to Micron's model (M0), which achieved an average accuracy of 82%.

Note, we do not compare the models using 'better than worst-case' current measures (I2), since they are more pessimistic than nominal currents for the DIMM being tested. Similarly, we do not compare the models using realistic current measures (I3), since the comparison is under nominal timings and not optimized timings, for which the I3 currents are relevant.

These results underline the accuracy of DRAMPower's power estimates and the significance of high-precision cycle-accurate modeling and use of accurate current measures as inputs.

## 4.4 Conclusion

In the first set of experiments, we showed significant performance gains by optimizing process-variation margins in DRAM timings. In the second set of experiments, we showed large differences in the current measured from real hardware and from DRAM datasheets. In the final set of experiments, we validated DRAMPower's power estimates. We showed that compared to real measurements in hardware, DRAMPower on average achieved over 97% accuracy in power estimation for a set of 17 micro-benchmarks. In comparison, Micron's model achieved around 82% accuracy, highlighting the importance of high-precision cycle-accurate modeling of DRAM state transitions and accurate scaling of power estimates based on actual observed timings, as performed by DRAMPower.

Table 4.6: Comparison of DRAMPower and Micron against Measurements

| | Meas. | Meas. $I_{DD}$ (I4) | | Nominal $I_{DD}$ (I1) | | Datasheet $I_{DD}$ (I0) | |
|---|---|---|---|---|---|---|---|
| | Ref. ($O_{ref}$) | DRAM-Power (M2) | Micron (M0) | DRAM-Power (M2) | Micron (M0) | DRAM-Power (M2) | Micron (M0) |
| Test # | (mW) | (mW) [%] | (mW) [%] | (mW) [%] | (mW) [%] | (mW) [%] | (mW) [%] |
| 1 | 490 | 500 [3%] | 659 [35%] | 636 [30%] | 848 [74%] | 689 [41%] | 970 [98%] |
| 2 | 658.5 | 655 [1%] | 885 [35%] | 823 [25%] | 1131 [72%] | 871 [33%] | 1278 [94%] |
| 3 | 1002 | 980 [3%] | 1050 [5%] | 1050 [5%] | 1145 [15%] | 1191 [19%] | 1316 [32%] |
| 4 | 1165.5 | 1182 [2%] | 1409 [21%] | 1328 [14%] | 1631 [40%] | 1435 [24%] | 1836 [58%] |
| 5 | 151.5 | 154 [2%] | 155 [3%] | 213 [41%] | 214 [42%] | 273 [80%] | 275 [82%] |
| 6 | 66.5 | 72 [8%] | 58 [13%] | 101 [52%] | 80 [20%] | 135 [104%] | 111 [67%] |
| 7 | 646.5 | 647 [1%] | 874 [36%] | 813 [26%] | 1116 [73%] | 859 [33%] | 1260 [95%] |
| 8 | 1119 | 1108 [2%] | 1152 [3%] | 1159 [4%] | 1220 [9%] | 1325 [19%] | 1405 [26%] |
| 9 | 562.5 | 551 [3%] | 604 [8%] | 726 [29%] | 795 [42%] | 830 [48%] | 923 [64%] |
| 10 | 631.5 | 631 [1%] | 669 [6%] | 835 [33%] | 884 [40%] | 958 [52%] | 1023 [62%] |
| 11 | 835.5 | 847 [2%] | 1000 [20%] | 965 [16%] | 1169 [40%] | 1060 [27%] | 1330 [60%] |
| 12 | 1446 | 1462 [2%] | 1751 [22%] | 1630 [13%] | 2017 [40%] | 1749 [21%] | 2260 [57%] |
| 13 | 703 | 672 [5%] | 819 [17%] | 863 [23%] | 1060 [51%] | 945 [35%] | 1205 [72%] |
| 14 | 877.5 | 870 [1%] | 1074 [23%] | 1110 [27%] | 1382 [58%] | 1197 [37%] | 1557 [78%] |
| 15 | 1104.75 | 1099 [1%] | 1368 [24%] | 1397 [27%] | 1757 [60%] | 1489 [35%] | 1965 [78%] |
| 16 | 73 | 78 [7%] | 91 [25%] | 100 [37%] | 117 [60%] | 177 [142%] | 197 [170%] |
| 17 | 687 | 693 [1%] | 705 [3%] | 1065 [55%] | 1080 [58%] | 1184 [73%] | 1206 [76%] |
| Avg. Accuracy (%) | | 97% | 82% | 74% | 54% | 52% | 26% |

# Chapter 5

# Real-Time DRAM Power Optimization

This chapter is based on our work on real-time power-optimization policies for DRAMs, published at DAC 2012 [86].

## 5.1 Introduction

Increasing performance and functionality demands of modern embedded systems often reflect poorly in overall system energy consumption. With the end of Dennard Scaling [130, 131], the power-density of chips is no longer constant, hence, power management has become extremely important. This becomes critical when the power supply is limited, as in the case of battery-powered devices. With the market pushing for both high-performance and green-computing solutions [6, 7], the demand for higher performance has been constantly met by tighter power and energy budgets. Hence, to satisfy the application performance requirements within the limited energy budgets, it is extremely important to employ run-time power optimization solutions that do not sacrifice the overall system performance, for all system components, including DRAMs.

DRAMs have been shown to contribute considerably to the system energy consumption [14], irrespective of whether they are in use or not. Several power optimization strategies have been proposed in the recent past targeting DRAM power consumption. While some target active-power management by reducing the number of DRAM operations, such as Activation and Precharging by exploiting row-buffer locality [129] or reducing refreshes, others target idle-power optimization by powering-down the DRAMs or scaling the frequency down [23] to reduce standby power consumption.

Active-power optimization policies reduce the overall DRAM usage by minimizing DRAM operations and improve average performance. As a consequence, they may also increase idleness in DRAMs. On the other hand, standby-power optimization powers-down the DRAM but may impact DRAM's performance guarantees, due to its power-up latencies. Hence, it is important for idle-power optimization strategies to efficiently avoid or hide the performance loss, so that they can retain the overall average-case and worst-case system performance. Note: Frequency scaling to reduce idle power consumption would always impact the latency of individual DRAM operations, due to the slowing down of the memory, hence, power-down mechanisms are more suitable when performance loss is not acceptable. Such *performance neutrality* in powering-down the DRAM could be essential for applications with real-time requirements, which demand worst-case performance guarantees from every component in the system, including the DRAMs and cannot tolerate any impact on guaranteed worst-case performance.

Real-time DRAM controllers provide such guarantees to a memory client, in terms of a minimum guaranteed bandwidth and/or a maximum latency bound for memory accesses. Real-time DRAM controllers, such as [62–67], employ predictable *memory arbiters*, such as Round-Robin or Time Division Multiplexing, to schedule memory accesses from different requesters and to provide performance guarantees. If they speculatively employ power-down mechanisms at run-time when the memory is idle, it can affect both the latency and the bandwidth guarantees provided, due to the power-up latencies [103]. As a result, the existing real-time memory controllers do not support run-time power-down. Hence, there is a need for performance-neutral power-down strategies that would retain the original DRAM performance guarantees, while reducing DRAM power consumption when it is idle.

Towards this, in this chapter, we propose: **(1)** a conservative and an aggressive DRAM power-down strategy and **(2)** a run-time power management policy that employs these two strategies efficiently, while preserving the original DRAM performance guarantees. Both these strategies exploit the idle memory service cycles by initiating DRAM power-down and can be employed with any real-time arbiter like TDM, Round-Robin, etc. They can be used with either of the two power-down modes (fast-exit or slow-exit), defined in Chapter 2. The difference between the two strategies is defined by the DRAM power-up scheme employed by them. The conservative strategy powers-up the DRAM by the end of every powered-down service cycle and avoids any impact on the original performance guarantees. The aggressive strategy on the other hand, powers-up only when there is a new request to serve and yet manages to avoid impacting the performance, by successfully hiding the power-up penalty within the original latency bounds.

To further assure that both these strategies are used in a performance-neutral way, we add a run-time power-management policy. This policy first evaluates both these power-down strategies for their applicability based on DRAM access granularity and memory service cycle durations. It then selects the most energy-efficient combination of power-down strategy and power-down mode (fast/slow exit) that does not violate the original DRAM performance guarantees.

Both the power-down strategies and the power management policy together can be adapted and employed with any of the real-time memory controllers presented in [62–67]. By employing the proposed performance-neutral power-down strategies with the run-time power management policy, these memory controllers can effectively and efficiently power-down the DRAM memory when it is idle, without impacting the original DRAM performance guarantees.

In this chapter, we experimentally evaluate the combination of the power management policy and the two proposed power-down strategies for three different DRAM generations on four concurrently executing media applications on an MPSoC platform using a real-time DRAM memory controller.

## 5.2 Related Work

Real-time DRAM memory controllers such as [62–67] employ predictable arbiters, like Round-Robin or TDM, and provide latency and/or bandwidth (rate) guarantees by bounding the temporal interference between requesters.

[63] employs Round-Robin arbitration and provides upper bounds on delays for different memory accesses. Similarly, [65] employs Round-Robin arbitration and uses worst-case response time as memory access latency bound. [62] adopts a budget-based static-priority arbitration and provides bounds on latency and guarantees a minimum bandwidth for every memory requester. It also supports Round-Robin or TDM arbiters.

[66] uses TDM arbitration and provides bandwidth guarantees and a worst-case execution time for memory accesses. In [64], weighted Round-Robin arbitration is used to provide both bandwidth guarantees and latency bounds. [67] uses static scheduling of commands and round-robin arbitration and provides predictable memory access latencies. However, none of these real-time memory controllers support power-down, due to the impact of power-up latencies on performance guarantees.

When it comes to work on idle power minimization in DRAMs, there exists no generic performance-neutral power-down solution for use with real-time memory controllers. For instance, [55] proposed to reduce idle power consumption by using a compiler-directed selective power-down and a hardware-assisted run-time power-down. However, the former is not applicable to memory controllers and the latter can incur large performance penalties due to mis-predictions of future idleness.

[24] proposed history-based scheduling and an adaptive memory throttling mechanism to allow memory to remain in the idle mode for longer periods of time to increase duration of power-down. [57] proposed a fuzzy-logic based prediction mechanism to dynamically employ power-down or self-refresh modes based on predicted idle period lengths. However, this mechanism is speculative by nature and cannot be used with real-time memory controllers. Unfortunately, these methods incur performance penalties and cannot be used for real-time applications with high performance requirements.

Several articles have targeted active power minimization. For instance, [23,61] employ DVFS techniques, [29,32,36,37,122,133] target improving row-buffer hits and [31, 124, 125, 127] target reducing of number of refreshes. However, these techniques may impact worst-case latencies of individual transactions, which can be critical in real-time systems. Besides, most of the existing real-time power-saving solutions provide application-specific optimizations.

In short, real-time memory controllers do not currently support generic power-down mechanisms, and existing power-saving solutions cannot be used with real-time memory controllers. This work bridges this gap and provides run-time power-down strategies for real-time DRAM memory controllers that are performance-neutral and optimize idle power consumption.

## 5.3 Background

In this section, we give details on real-time DRAM memory controller architectures and arbiters and their real-time guarantees that form the basis for deriving the power-optimization strategies.

### 5.3.1 Memory Controller

A generic DRAM memory controller architecture can be visualized as comprising two sections, a front-end and a back-end, as depicted in Figure 5.1.

The front-end includes a bus and buffers per input port of the bus to accept incoming requests from different requesters (clients) and an arbiter that selects one of them at a time to be served by the memory. The back-end includes the logic required to translate the arbiter selected request into a set of scheduled commands, address and data components of a memory transaction and forwards them to the DRAM memory, based on the memory's timing requirements.

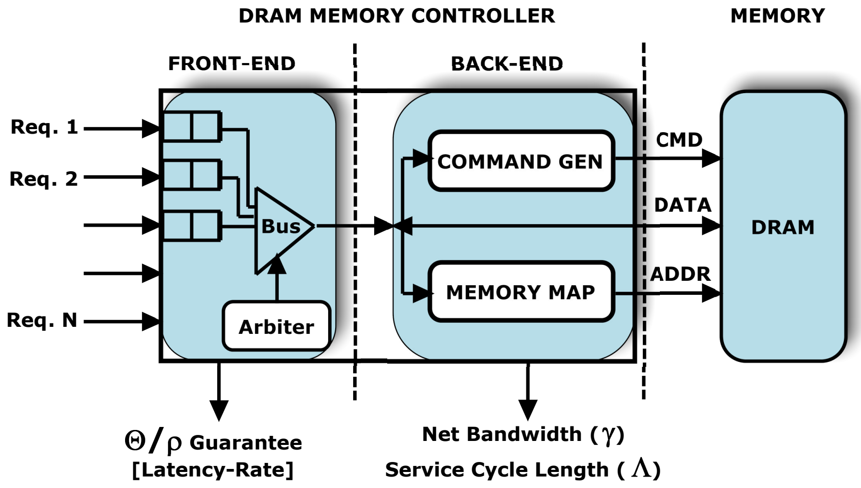


Figure 5.1: Memory Controller Overview

In the case of real-time DRAM controllers, both of these sections together provide performance guarantees for each incoming request in terms of a maximum bound on the latency and a minimum bound on the bandwidth [117], such that each incoming request is served in a finite known time.

As depicted in the figure, the back-end defines a *net memory bandwidth* ($\gamma$) based on the *maximum service cycle length* ($\Lambda$) for a given *memory access granularity (G)*. The net memory bandwidth gives the actual memory bandwidth attainable for requests of a given access granularity taking into account the performance loss due to DRAM timing requirements on related operations, as opposed to the theoretical peak memory bandwidth where 2 words of data are assumed to be transferred every clock cycle in or out of the memory. The maximum service cycle length gives the worst-case latency of scheduling DRAM commands corresponding to a read or a write request of a defined *access granularity* (supported request size in bytes). Hence, the net memory bandwidth is defined by access granularity of a request.

In the front-end, the arbiter defines a *maximum latency bound* ($\Theta$) and a *minimum guaranteed rate of service* ($\rho$), in terms of service cycles for every requester. The maximum latency bound is defined in terms of number of service cycles and depends on (1) the number of requesters accessing the DRAM (also indicated by the number of input ports on the bus) and (2) the service cycles allocated by the arbiter to each of those requesters. The guaranteed rate of service gives the proportion of the service provided to all requesters, that is allocated to the particular requester.

By combining the latency-rate guarantee (in terms of service cycles) of the front-end and the net memory bandwidth and the maximum service cycle length defined by the back-end, we can define a requester's bandwidth guarantee ($\beta$) as the fraction $\rho$ of the net memory bandwidth ($\gamma$) and an access time guarantee ($\Phi$) as the equivalent of $\Theta$ in absolute time.

### 5.3.2 Arbiters and Latency-Rate Servers

Real-time DRAM memory controllers employ predictable arbiters to provide latency and rate (bandwidth) guarantees to requesters accessing the memory. These arbiters use real-time scheduling algorithms like Round-Robin and TDM, and can be analyzed using the Latency-Rate ($\mathcal{LR}$) server model [95], to characterize their performance guarantees.

These guarantees are provided to a requester in terms of a *minimum allocated rate of service* ($\rho$) and a *maximum service latency* ($\Theta$), whenever it is *busy* (requesting a higher rate of service on average than allocated to it). To formally define a *busy* requester, we first define standard Latency-Rate notations.

Let $\rho_i$ denote the allocated rate of service for requester $i$, $A_i(\tau_1, \tau_2]$ denote the rate of arrival of requests of requester $i$ in time interval $(\tau_1, \tau_2]$, and $W_i(\tau_1, \tau_2]$ denote the rate of service provided to requester $i$ in the same time interval. Then, we can define the busyness of a requester as per Definition 5.1.

**Definition 5.1** *Busy Requester*: A requester $i$ is *busy* in a time interval $(\tau_1, \tau_2]$, if its arrival rate is greater than its allocated rate of service, defined by Equation (5.1).

$$A_i(\tau_1, \tau_2] \geq \rho_i \cdot (\tau_2 - \tau_1) \tag{5.1}$$

Hence, a requester is considered *busy*, if its arrival rate during the time interval $(\tau_1, \tau_2]$ is greater than or equal to its minimum allocated rate of service. Note that it is possible that the actual *provided service* is higher than the allocated rate of service, if the system has the unallocated or unused capacity to support it. Hence, a requester's *busyness* is defined independently of the rate of service provided to it and is only based on the rate of service allocated to it. Whereas, a requester's *backlog* is defined in terms of its actual provided service ($W_i$) instead of its minimum allocated rate of service, as described in Definition 5.2.

**Definition 5.2** *Backlogged Requester*: A requester $i$ is *backlogged* at time $t$, if its arrival rate is greater than its provided service, defined by Equation (5.2).

$$A_i(0, t) > W_i(0, t) \tag{5.2}$$

These service rates are depicted in the form of service curves for a requester in Figure 5.2.

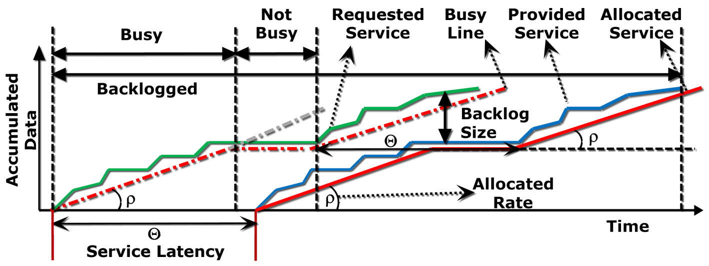


Figure 5.2: Latency-Rate Server

In the figure, we identify a *busy line* as the red dash-dotted line representing the slope of the allocated rate of service $\rho$. A *busy period* for a requester corresponds to the maximum time interval when its *requested service rate* (indicated by the green curve) is above the *busy line* (allocated rate of service), else, it is considered to be *not busy* (lower requested service rate). The requested service curve depicts the arrival of the requests from the requester over time and the allocated service line depicts the guaranteed service rate by which the requests of a *busy* requester is served over time.

The actual *provided service* (indicated by the blue curve) may be higher than the allocated rate. This provided service curve shows how the requests actually get served over time by the memory, which can be greater than or equal to the allocated service rate. As a result, in such cases, the requester may be busy but not backlogged, since it got better service rate than its minimum allocated rate.

As stated earlier, a *busy* requester is also guaranteed a maximum service latency ($\Theta$). This maximum latency bound gives the maximum initial duration after which the busy requester starts getting served at its allocated rate ($\rho$) by the memory. During this waiting period, the requester gets backlogged (since it is busy and not getting service) and when it is not longer busy, its backlogged requests accumulated during its busy period will get served at the guaranteed rate. We formally define a Latency-Rate server characterized by $\Theta$ and $\rho$ using Definition 5.3.

**Definition 5.3** *Latency-Rate* ($\mathcal{LR}$) *Server* (Maximum Service Latency ($\Theta_i$) and Minimum Service Rate ($\rho_i$)): If a requester $i$ starts a new busy period $j$ for a time interval $(\tau_1, \tau_2]$, a server $S$ is an $\mathcal{LR}$ server, if and only if for any $t \epsilon (\tau_1, \tau_2]$ it provides the requester $i$ a guaranteed rate of service $\rho_i$ delayed only by an initial non-negative time period $\Theta_i$, as shown in Equation (5.3):

$$W^S_{i,j}(\tau_1, t) \geq \rho_i \cdot (t - \tau_1 - \Theta_i) \tag{5.3}$$

In this case, $\Theta$ depends on the number of interfering requests from competing requesters and is defined in terms of service cycles. $\rho$ is defined by the fraction of service cycles allocated to the requester. Note that the actual observed service latency, may be less than the maximum service latency bound ($\Theta$), if there is less interference from competing requesters. However, if the requester is provided only its allocated rate of service, as long as it is busy, it will always be backlogged (have pending requests), since it will have more incoming requests than served requests. In other words, if a requester $i$ is busy in a time interval $(\tau_1, \tau_2]$, then Equation (5.4) holds for any $t \epsilon (\tau_1, \tau_2]$.

$$A_i(\tau_1, t) \geq \rho_i \cdot (t - \tau_1) > \rho_i \cdot (t - \tau_1 - \Theta_i) \tag{5.4}$$

As shown in this equation, for any $\Theta_i > 0$, for a busy requester, the arrival rate will be greater than the allocated service rate, hence the requests will always be backlogged. In short, a Latency-Rate ($\mathcal{LR}$) arbiter provides a busy requester, a guaranteed rate of service $\rho$ after a maximum service latency $\Theta$ (measured in service units). These translate to absolute time ($\beta$ (Minimum Bandwidth Guarantee) and $\Phi$ (Maximum Access Time Guarantee)) when combined with a real-time memory controller back-end.

### 5.3.3 Deriving Memory Controller Guarantees

This section describes how bandwidth (rate) and access time (latency) guarantees are derived for a real-time DRAM controller.

To define the minimum bandwidth guarantee, we begin with deriving the net memory bandwidth. The net memory bandwidth ($\gamma$) is predominantly defined by the *access granularity* parameter (G), which is given by the request (atom) size supported by the memory controller for all requests and the *maximum service cycle length* ($\Lambda$) for the particular access granularity.

The maximum service cycle length ($\Lambda$) for a given access granularity of the memory controller can be computed as the maximum of: (1) the service time of a read transaction (*nRD*) and the time to switch to a read after a write (*nWTR*), and (2) the service time of a write transaction (*nWR*) and the time to switch to a write after a read (*nRTW*) (given by Equation (5.5)).

$$\Lambda = max(nWTR + nRD, nRTW + nWR) \tag{5.5}$$

The read and write transaction service times *nRD* and *nWR* are derived as the maximum time required to serve a read and a write transaction, respectively [117]. The time to switch between read and write transactions (or vice versa) are also derived by the analysis of the memory controller based on memory-specific timing constraints [117, 132].

Using the definition of maximum service cycle length (Equation (5.5)), we can derive the net memory bandwidth. We compute the net memory bandwidth ($\gamma$) over a period of the length of a refresh interval (*nREFI*). In doing so, we consider every service cycle during *nREFI* (in clock cycles) to be as long as the maximum service cycle length ($\Lambda$). Hence, the total number of service cycles ($num\_\Lambda$) in *nREFI* is given by Equation (5.6). Note that this is a conservative estimate used for simplification of the analysis. Here, *nRFC* is the length of a single refresh request in clock cycles.

$$num\_\Lambda = \lfloor (nREFI - nRFC)/\Lambda \rfloor \tag{5.6}$$

The total data transfer during this period, assuming an access granularity of *G* bytes, is given by $num\_\Lambda \times G$. Hence, the net memory bandwidth in one refresh interval *tREFI* is given by Equation (5.7).

$$\gamma = num\_\Lambda \times G/tREFI \tag{5.7}$$

Having derived the net memory bandwidth, next, we express the guaranteed rate of service ($\rho_i$) in absolute time. Assuming TDM or Round-Robin arbitration, say 'x' is the number of service cycles allocated to the requester ($i$) from the total number of service cycles in a frame of size F (where, F > x), $\rho_i$ corresponds to the fraction of service cycles allocated to requester $i$, as shown in Equation (5.8).

$$\rho_i = x_i/F \tag{5.8}$$

Using the net memory bandwidth definition from Equation (5.7) and value of $\rho_i$ from Equation (5.8), we define the minimum bandwidth guarantee in Definition 5.4, as follows:

**Definition 5.4** (Minimum Bandwidth Guarantee): A minimum bandwidth guarantee $\beta_i$ is provided to a requester $i$ based on its allocated guaranteed rate of service ($\rho_i$) and the net memory bandwidth ($\gamma$), as shown in Equation (5.9).

$$\beta_i = \rho_i \times \gamma \tag{5.9}$$

Having defined the minimum bandwidth guarantee $\beta_i$, next, we define the *maximum access time guarantee* ($\Phi_i$) provided to a requester $i$ as the maximum initial time period for which the requester in a new busy period may have to wait, before it is provided sustained service at its allocated bandwidth ($\beta_i$). $\Phi_i$ represents the latency guarantee ($\Theta_i$) in absolute time. Equation (5.10) informally depicts this relation, however, it is expanded later to consider all interferences in a real system in Equation (5.16).

$$\Phi_i = \Theta_i \times \Lambda \tag{5.10}$$

Using this definition of maximum access time guarantee ($\Phi_i$), the maximum finishing time of any request $j$ of Requester $i$ in a busy period can be defined as the maximum of *(1) arrival time of request $j$ + $\Phi_i$ or (2) completion time of the last request $j-1$ in the same busy period* and $\rho$ fraction of service cycles in a service time frame ($F$), as given by Equation (5.11).

$$t_{fin_i^j} = max(t_{arr_i^j} + \Phi_i, t_{fin_i^{j-1}}) + F/x_i \times \Lambda \tag{5.11}$$

The exact value of $\Phi$ depends on the type of Latency-Rate arbiter being used, such as Round-Robin, TDM, Credit-Controlled Static Priority, etc. [117,118]. In this work, we restrict our analysis to the most commonly used ($\mathcal{LR}$) arbiters, Round-Robin and TDM, which use slot/frame-based arbitration (allotting a defined number of service slots per requester per arbitration time-frame).

To derive a conservative common bound for $\Phi$ applicable to Round-Robin and TDM, we begin by defining the timing parameters that contribute towards the same. We derive the worst-case value of $\Phi$ assuming all requesters are busy and competing for the memory.

1. The maximum service cycle length parameter ($\Lambda$) is defined as the maximum time period to service a given request of access granularity $G$. It can be computed as discussed earlier in Equation (5.5).

2. The minimum service cycle length parameter ($\lambda$) gives the minimum time period for the memory to serve a request of access granularity $G$, after which the next request may have to be scheduled. This parameter $\lambda$ can be employed by the arbiter as its *scheduling interval*: defined as the time instant during a service cycle, at which it decides which requester to serve next. $\lambda$ is the minimum of the service latencies of a read or write transaction, as shown in Equation (5.12). Note that, in general, arbiters may employ different time instances per service cycle for scheduling the next request. However, a scheduling interval as long as $\lambda$ waits long enough to give subsequent requesters sufficient time to arrive before their slots and high-priority requests to be detected as late as possible, which is better for lower worst-case access time guarantee for high-priority requesters by avoiding getting blocked by lower priority requests (see next point).

Hence, to be deterministic and derive lower worst-case guarantees, we employ the scheduling interval parameter for all service cycles. Note: If no request is waiting to be served, the slot goes idle and is at most as long as the scheduling interval, since anything shorter will affect the worst-case guarantee, as will be shown next.

$$\lambda = min(nRD, nWR) \tag{5.12}$$

The read and write transaction service times ($nRD$ and $nWR$) are derived by the memory controller analysis as stated before.

3. Combining Equations (5.5) and (5.12), the difference between the two, gives the time for which the next requester selected for service at the scheduling interval ($\lambda$) may have to wait before the end of the current service cycle.

   The shorter this difference, the lower is the effective value of $\Phi$ (worst-case access time guarantee). Hence, any value of scheduling interval shorter than $\lambda$ will only worsen the worst-case access time guarantee $\Phi$. We define this difference between the two as the *Blocking Period*, denoted by $\Delta$, as shown in Equation (5.13).

$$\Delta = \Lambda - \lambda \tag{5.13}$$

4. Besides the *Blocking Period* ($\Delta$), a request ($r_i^k$) may have to wait longer depending on the time required to serve the interfering memory requests. As stated before, assuming TDM or Round-Robin arbitration, with $x_i$ being the number of service cycles allocated to the requester $i$ from the total number of service cycles in a frame of size F, a request ($r_i^k$) in the worst-case, may have to wait for $F$ - $x_i$ requests to be served first, before it is forwarded for service. This latency is defined as the *Interference Period*, denoted by $I$, given by Equation (5.14).

$$I_i = (F - x_i) \times \Lambda \tag{5.14}$$

5. Also, if the request ($r_i^k$) arrives just after the scheduling interval during a service cycle, it may be delayed further by one full service cycle in addition to the $F$ - $x$ service cycles, if another request was scheduled in its place.

   This period is defined as the *Penalty Period*, denoted by $P$ and in the worst-case is equal to one $\Lambda$.

6. Finally, since the *Interference Period* may stretch longer than a refresh interval (*nREFI*), further interference may be caused by automated Refresh requests. This additional delay is defined as *Refresh Delay*, denoted by *nRef*, derived in Equation (5.15).

In this equation, we conservatively figure out the number of refresh intervals during the *Interference Period* and add the delays due to the corresponding number of refresh requests. From the refresh interval, we subtract $\Lambda$ to account for any delay in scheduling the refresh due to a currently execution read or write request of request size $G$.

Finally, we add the latency of one full refresh operation (+1 in the equation) to account for the fact that the *Interference Period* and refresh interval may not be aligned (a refresh may be issued at the beginning of the Interference Period) and this can result in one additional refresh operation.

$$nRef = \big(\lceil((F - x_i) \times \Lambda + P_i + \Delta)/(nREFI - \Lambda)\rceil + 1\big) \times nRFC \tag{5.15}$$

Together, these delays add up to define a conservative value of $\Phi_i$ for a given requester $i$, for Latency-Rate arbiters like TDM and Round-Robin in Definition 5.5, as follows:

**Definition 5.5** (Maximum Access Time Guarantee): A maximum access time guarantee $\Phi_i$ is provided to a requester $i$ as the maximum time period the requester may have to wait in a new busy period to get its allocated service. The value of $\Phi_i$ is given by the sum of the service cycles of all interfering requests, including refreshes, as shown in Equation (5.16).

$$\Phi_i = \Delta + nRef + P_i + I_i \tag{5.16}$$

Figure 5.3 sums up these worst-case interfering service cycles to derive a conservative bound for $\Phi_i$.

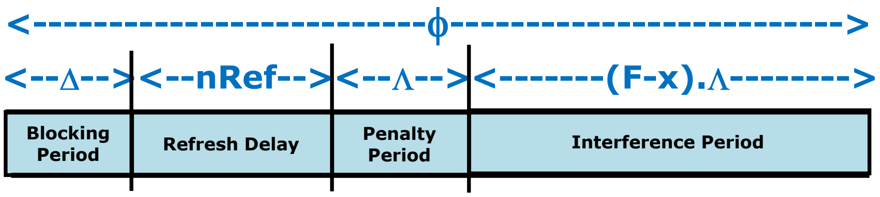


Figure 5.3: Deriving Worst-Case Maximum Access Time Guarantee ($\Phi$)

This access time guarantee ($\Phi$) is a conservative estimate, and may be optimized specific to the actual arbitration being used (as shown in [116, 117]).

Having derived the original bandwidth and access time guarantees provided by a real-time memory controller, next, we define our proposed real-time DRAM power-down strategies and show that these original performance guarantees are preserved even when employing them.

## 5.4 Real-Time Power-Down Strategies

Employing DRAM power-down mechanisms may affect the access time (latency) and bandwidth guarantees provided by a real-time memory controller, due to the expected performance impact of the *powering-up latencies*. Hence, existing real-time memory controllers do not support DRAM power-down. In this section, we propose two performance-neutral run-time power-down strategies for real-time memory controllers that employ predictable $\mathcal{LR}$ arbiters; one a conservative strategy and the other an aggressive strategy. These strategies can be employed whenever the memory is idle (not backlogged) to reduce memory power consumption, while preserving the original guaranteed performance bounds.

As stated before in Chapter 2, it is possible to employ either the fast-exit or slow-exit modes, when employing DRAM power-down. We can hence derive *Power-Up Latency*, the period after which the DRAM can be accessed again (sent an ACT command), as per Equation (5.17) for fast-exit and as per Equation (5.18) for slow-exit.

$$nPUP(F) = nXP \tag{5.17}$$

$$nPUP(S) = nXPDLL - nRCD \tag{5.18}$$

As can be noticed in Equation (5.18), *nPUP* contains the minimum timing constraint between an ACT and a RD/WR command (*nRCD*) [103], besides the slow-exit (*nXPDLL*) power-up timing constraint. The rationale behind it is as follows: When using the slow-exit power-down mode in the precharged state, every read/write transaction ends with a precharge and begins with an ACT command. The power-up timing constraint before issuing a RD/WR command after a slow-exit power-down is given by *nXPDLL* in Equation (5.18). For efficient memory access, the first RD/WR command is scheduled immediately after *nRCD* is satisfied, after an ACT command is issued. However, since a RD/WR can be issued only after a duration of *nXPDLL* cycles, after the memory begins to power-up [103], the corresponding ACT in the transaction can be issued immediately after *nXPDLL* - *nRCD* is satisfied, to avoid any additional delays in issuing the RD/WR command.

### 5.4.1 Conservative Strategy

The first strategy involves triggering a special *power-off* request whenever an arbiter service cycle is idle. This power-off request is designed to power-down the memory and power it back up within the minimum service cycle length, thus hiding the power-up transition latencies within the idle service cycle. This ensures that the scheduling of memory access requests is not disturbed and the power-down mechanism is effectively hidden from the requesters. This latency and bandwidth neutral strategy provides significant energy savings and preserves both the guaranteed access time bound and bandwidth. The idle service cycle length is equal to the minimum service cycle length ($\lambda$), as defined earlier in Equation (5.6).

Since both the power-down period ($nPDN$) and power-up period ($nPUP$) should fit within $\lambda$, we derive the length of the power-down period in Equation (5.19).

$$nPDN = \lambda - nPUP \tag{5.19}$$

### 5.4.2 Aggressive Strategy

The second strategy is more aggressive, since it checks for new requests before powering up the memory. It involves issuing a *power-down* request when there are no pending requests at the arbiter, and snooping the arbiter inputs for new requests before the end of the current idle service cycle. If there are any pending requests, a *power-up* request is issued to the memory to power it up by the end of the idle service cycle (thus maintaining the scheduling interval).

To implement this strategy, we introduce a *Snooping Point* at a pre-defined time instance from the start of the idle service cycle, before the end of the scheduling interval, as shown in Figure 5.4. This snooping point ($n_{Snoop}$) can be derived by subtracting the power-up latency ($nPUP$) (F/S) from the minimum (idle) service cycle length ($\lambda$), as given by Equation (5.20). This strategy assures that the memory powers-up in time and the following request is scheduled on-time, if it arrives before this snooping point.

$$n_{Snoop} = \lambda - nPUP \tag{5.20}$$

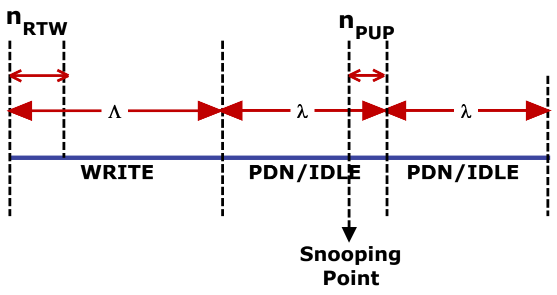


Figure 5.4: Snooping Point in Aggressive Power-Down

However, if the next request arrives after the snooping point but before the end of the scheduling interval, no power-up will be issued and the memory continues to be in the power-down mode for one more idle slot. This results in the next request being delayed by a service cycle of length $\lambda$.

Such a situation may impact the original performance guarantees, since they were derived using the scheduling interval as the time to schedule the next transaction and not the snooping point, which is introduced to support the power-down mechanism. To ascertain any performance impact of using the snooping point as the scheduling time when the power-down mode is employed, we derive the performance guarantees based on the snooping point, and compare them to the original guarantees. It is important to note though, that the original scheduling interval is retained whenever there are pending (backlogged) requests in the system and the memory is not idle. The snooping point is employed only when the memory is powered-down. Hence, the worst-case performance guarantees when the system is busy, are not affected by power-down, since it is not employed when the system is not idle. However, it must be assured that the performance guarantees when the system is indeed powered-down, are not any worse than the original performance guarantees.

Towards this, we first define a value of access time guarantee for when the system is idle (powered-down) as $\Phi'$ in Equation (5.21). $\Phi'$ should be lesser than or equal to $\Phi$. When the system is in the power-down mode and the snooping point is employed in place of the original scheduling interval, instead of the blocking period of $\Delta = (\Lambda - \lambda)$ in Equation (5.16), a minimum delay equal to the power-up period of $nPUP$ is observed, before the request can be served. During this power-up period, competing requests may arrive, and as a result, the worst-case interference period ($I$) due to competing requests and penalty due to additional refreshes is applicable as before. Furthermore, if the request arrives just after the snooping point and the system continues to be in the power-down state, a penalty period ($P$) is used up by an idle slot of length $\lambda$ (minimum service cycle) and not $\Lambda$ (maximum service cycle due to another request), as in the original case.

Combining these conditions, we derive a value of $\Phi'$, when the system is idle, as shown in Equation (5.21).

$$\Phi' = nPUP + ((F - x) \times \Lambda) + \lambda + nRef \tag{5.21}$$

Since $\Phi'$ must be less than or equal to $\Phi$, the difference between the definitions of $\Phi$ and $\Phi$' must at most be zero. Combining the two, we derive the following:

$$(\Lambda - \lambda + \Lambda) - (nPUP' + \lambda) \geq 0 \tag{5.22}$$

where, $nPUP'$ is the maximum acceptable value of power-up penalty ($nPUP$) such that $\Phi'$ is lesser than or equal to $\Phi$.

This reduces to:

$$2 \times \Delta \geq nPUP' \tag{5.23}$$

This defines the maximum value of $nPUP'$, the maximum value of power-up period that does not affect the original latency guarantee ($\Phi$). Hence, the aggressive power-down strategy is employed when the power-up penalty is shorter than $2 \times \Delta$. As a result, it can successfully hide any performance impact within the original $\Phi$. We define Equation (5.23) as the *power-up neutrality condition.*

Since the aggressive strategy employs the snooping point to power-up and always powers-up within the idle period length $\lambda$, it does not affect the value of $\lambda$ or $\Lambda$. As a result, it has no impact on the original bandwidth ($\beta$) and latency ($\Phi$) guarantees. In essence, by using the aggressive strategy only when the neutrality condition is met, ensures it to be latency (access-time) and bandwidth neutral, thus guaranteeing real-time memory performance. The advantage of this strategy is that all contiguous idle periods are combined into one large idle period, thereby avoiding frequent powering-up of memory as in the case of the conservative strategy, to save more energy.

## 5.5 Impact of Speculative Strategies

To highlight the importance of the proposed conservative and aggressive power-down strategies in the real-time systems context, we analyse the impact of powering-up the DRAM when a new request arrives at the arbiter. We define such a power-down strategy as a *Speculative power-down strategy* that powers-down the memory whenever it is idle and *allows it to power-up even at any point in the idle service cycle* (even after the snooping point).

In the worst-case, a request may arrive at the last clock cycle of the idle service cycle (the original scheduling interval, $\lambda$) and hence, the power-up transition time (*nPUP*) is added to the maximum service cycle length ($\Lambda$) of the following request. This impact on $\Lambda$ increases its value to $\Lambda''$ as a result of a speculative power-up and is shown in Equation (5.24). Hence, this reduces the number of service cycles in a refresh interval and thus, the net memory bandwidth from $\gamma$ to $\gamma''$ and the bandwidth guarantee from $\beta$ to $\beta''$, as shown in Equations (5.25) and (5.26), respectively. It also increases $\Phi$ to $\Phi''$ due to the impact of the power-up latency (*nPUP*) on $\Lambda$, based on Equation (5.21), as shown in Equation (5.27).

$$\Lambda'' = \Lambda + nPUP \tag{5.24}$$

$$\gamma'' = num\_\Lambda'' \times G/nREFI \tag{5.25}$$

$$\beta'' = \gamma'' \times \rho \tag{5.26}$$

$$\Phi'' = nPUP + ((F - x) \times \Lambda'') + \lambda + nRef \tag{5.27}$$

Such a drop in bandwidth guarantee and increase in access-time guarantee are not acceptable in high-performance real-time systems, as we show in the next section.

## 5.6 Power Management Policy

In this section, we present a power-down management policy (Algorithm 4) that actively determines the combination of the best power-down strategy (conservative or aggressive) and power-down mode (fast-exit or slow-exit) [103] based on their best achievable power-savings and the use-case (access granularity) defined at design-time.

**Algorithm 4:** Power-Down Management Algorithm

```
Require: $n_{Off}$ $(= \lambda)$
1: if $n_{Off} > nCKE + nXPDLL - nRCD$ then
2:    {Comment: Minimum PRE Slow-Exit Duration}
3:    if $nXPDLL - nRCD \leq 2 \times (max\_SCL - min\_SCL)$ then
4:       {Comment: Performance Neutrality Condition - Aggressive}
5:       $Mode \leftarrow Slow\text{-}exit$
6:       $Strategy \leftarrow Aggressive$
7:    else
8:       {Comment: Possible Mode-Strategy - Slow-exit-Conservative}
9:       if $nXP \leq 2 \times (max\_SCL - min\_SCL)$ then
10:         {Comment: Performance Neutrality Condition - Aggressive}
11:         Evaluate Slow-exit/Conservative and Fast-exit/Aggressive for $n_{Off} \times 2$
12:         $Mode \leftarrow Best\ Mode$
13:         $Strategy \leftarrow Best\ Strategy$
14:      else
15:         $Mode \leftarrow Slow\text{-}exit$
16:         $Strategy \leftarrow Conservative$
17:      end if
18:   end if
19: else if $n_{Off} > nCKE + nXP$ then
20:   {Comment: Minimum PRE Fast-Exit Duration}
21:   if $nXP \leq 2 \times (max\_SCL - min\_SCL)$ then
22:      {Comment: Performance Neutrality Condition - Aggressive}
23:      $Mode \leftarrow Fast\text{-}exit$
24:      $Strategy \leftarrow Aggressive$
25:   else
26:      $Mode \leftarrow Fast\text{-}exit$
27:      $Strategy \leftarrow Conservative$
28:   end if
29: else
30:   $Mode \leftarrow No\_PD$
31:   $Strategy \leftarrow No\_PD$
32: end if
33: return $Mode\ and\ Strategy$
```

To employ Algorithm 4, we derive a 'power-off' ($nOff$) period for the entire power-down request (including the power-up latency $nPUP$ and minimum power-down time $nCKE$), equal to the $\lambda$ (idle service cycle length), which is derived based on the access granularity of the requests, as shown in Equation (5.7). Based on the 'power-off' period, the algorithm may select either the aggressive strategy or the conservative strategy with the slow-exit or fast-exit power-down mode, while ensuring the neutrality condition in Equation (5.23) is not violated. Also, if there can be no energy savings with any of the power-down modes, the algorithm opts against the use of power-down (No_PD).

As shown in the algorithm (Line 3), if the neutrality condition is satisfied, the aggressive power-down strategy with the slow-exit mode is selected. If not, other combinations are evaluated (Lines 25 to 33). For instance, if the neutrality condition is met with the aggressive strategy using the fast-exit mode, we evaluate the power-savings offered by slow-exit conservative strategy combination against th at of the fast-exit aggressive strategy for a conservative minimal power-down period of $n_{Off} \times 2$ (Line 28). We employ this minimal power-down period for the evaluation, since the aggressive strategy is useful only if there are at least two consecutive idle slots.

However, if the aggressive strategy cannot ensure bandwidth neutrality, then the slow-exit power-down mode is employed in combintation with the conservative strategy (Lines 31/32). If the 'power-off' period is not long enough to employ slow-exit power-down mode, we check for the possibility of using fast-exit power-down with aggressive strategy (Line 13). If this is not feasible, the fast-exit mode is selected with the conservative strategy (Lines 18/19). If the idle service cycle cannot accommodate any power-down mode, then the algorithm opts against the use of power-down.

In short, this power management policy conservatively selects the best possible mode-strategy combination (if feasible) for the given use-case, that assures no impact on the original performance guarantees.

## 5.7 Experiments and Results

We perform experiments to show: (1) the effectiveness of the power-management policy that selects a combination of power-down mode and strategy for different memories and (2) the power-saving efficiency of the proposed policies for the different memories when compared to using no power-down, speculative power-down and theoretical best power-savings.

### 5.7.1 Power-Management Policy

We evaluated four memories each (of different frequencies) from 7 DRAM generations (DDR2/3/4, LPDDR/2/3 and Wide IO DRAM) to observe the effectiveness of the power-management policy. From these evaluations, we selected four memories across different DRAM generations, that caused the power-management policy to select a unique combination of power-down strategy (conservative or aggressive) and power-down mode (slow-exit or fast-exit) to enable performance neutral power-management.

These memories and their selection of power-down strategy and mode include:

1. MICRON 1Gb DDR2-1066 x16 - Aggressive / Slow-exit
2. MICRON 1Gb LPDDR-333 x32 - Conservative / Fast-exit
3. MICRON 2Gb DDR3-1600 x8 - Conservative / Slow-exit
4. MICRON 2Gb LPDDR2-1066-S4 x32 - Aggressive / Fast-exit

Employing these four memories as targets for the power-management policy, we evaluate the two proposed real-time power-down strategies and power-management algorithm against a speculative power-down and a theoretical best power-down policy. We derive two sets of results: (1) The efficiency of the power-management policy (PD) in terms of its impact on performance guarantees $\Phi$ and $\beta$ compared to using speculative power-down. (2) The effectiveness of the power-management policy (PD) with respect to its impact on energy savings and execution times of applications and compare the same against speculative power-down.

### 5.7.2 Experimental Setup

We employ four commonly used Mediabench [83] applications: (1) H.263 encoder, (2) EPIC Encoder, (3) JPEG Encoder and (4) MPEG2 Decoder. We executed these four test applications independently on the Simplescalar simulator [69] with a 16KB L1 D-cache, 16KB L1 I-cache, 128KB shared L2 cache and 64-byte and 128-byte cache line configurations. We filtered out the L2 cache misses, and obtained a trace of the transactions meant for the DRAM memory. The traces were obtained assuming zero DRAM latency and the actual latencies are included by the memory controller during the simulations.

To simulate four requesters, we employ the traces from these four applications on four blocking trace players in a SystemC model of our real-time MPSoC platform [70] connected to our real-time DRAM memory controller [62] that uses a Round-Robin arbiter.

To derive the theoretical best-case energy savings when using power-down, we generate memory traces without enabling power-down and post-process the same by manually inserting power-down and power-up commands (based on the selected mode and strategy) during idle periods such that there is no impact on the trace execution times and performance guarantees. We use the observed energy savings as the theoretical best-case value.

The memory controller implements the power-management policy with the two strategies/modes and powers down the memory whenever it is idle. The memory controller also implements a speculative power-down strategy to compare its impact against the two proposed strategies.

### 5.7.3 Impact on Performance Guarantees

We show the effectiveness of the power-management policy (PD) in terms of its impact on performance guarantees $\Phi$ (access-time guarantee) and $\beta$ (bandwidth guarantee) compared to using speculative power-down in Table 5.1. The table depicts the combinations of power-down mode and strategy used by the power management policy for the four memories and their impact on the worst-case latency and bandwidth guarantees compared to not using any power-down. The table also shows the possible impact of speculative power-down policy on these factors for these memories.

Table 5.1: Difference in Worst-Case Bandwidth and Latency

| *Mem Type* | *PD Policy* | *PD $\beta'$ (%)* | *PD $\Phi'$ (%)* | *Spec $\beta''$ (%)* | *Spec $\Phi''$ (%)* |
|---|---|---|---|---|---|
| DDR2-1066 | AGG / SE | 0% | 0% | -10.9% | 3.1% |
| LPDDR-333 | CON / FE | 0% | 0% | -13.2% | 2.6% |
| LPDDR2-1066 | AGG / FE | 0% | 0% | -16.8% | 5.5% |
| DDR3-1600 | CON / SE | 0% | 0% | -15.0% | 7.8% |

As can be observed from the table, the power management policy (PD) avoids any impact on both the latency and bandwidth guarantees, while the speculative policy significantly impacts both the performance guarantees (up to 7.8% increase in $\Phi$ and 16.8% decrease in $\beta$), compared to no power-down.

### 5.7.4 Power-Saving Efficiency

In these experiments, we evaluate the efficiency of the two proposed real-time power-down strategies and power-management policy against a speculative power-down and a theoretical best power-down policy, with respect to their energy savings and execution times.

Figure 5.5 shows the average memory power savings achieved when running the four application traces and using the selected power-down policy for the 4 memories. It also compares the same against the savings offered by the speculative policy and a theoretical best power-down policy.

Table 5.2 presents the energy consumption measures before and after employing these power-down policies. As can be noticed from the results, the proposed power-down strategy obtains power-savings close to the theoretical best savings for all memories and all combinations of power-down mode and strategy. The speculative policy also achieves significant energy savings (always $\leq 1\%$ from theoretical best), however, with a much larger impact on the performance guarantees. Note: The differences in the energy savings for the different memories depend not just on the combination of power-down mode and strategy employed, but also on the potential power savings offered by the power-down modes for that memory.

Table 5.2: Energy Consumption using different policies

| *Mem Type* | *PD Policy* | *No PD Energy (mJ)* | *PD Energy (mJ)* | *Speculative Energy (mJ)* | *Theoretical Energy (mJ)* |
|---|---|---|---|---|---|
| DDR2-1066 | AGG / SE | 192.78 | 59.62 | 60.32 | 59.55 |
| LPDDR-333 | CON / FE | 250.83 | 32.12 | 32.5 | 32.08 |
| LPDDR2-1066 | AGG / FE | 100.03 | 33.4 | 33.87 | 33.36 |
| DDR3-1600 | CON / SE | 126.06 | 42.01 | 42.55 | 41.89 |

Figure 5.6 shows the impact on the total execution time of the four applica-

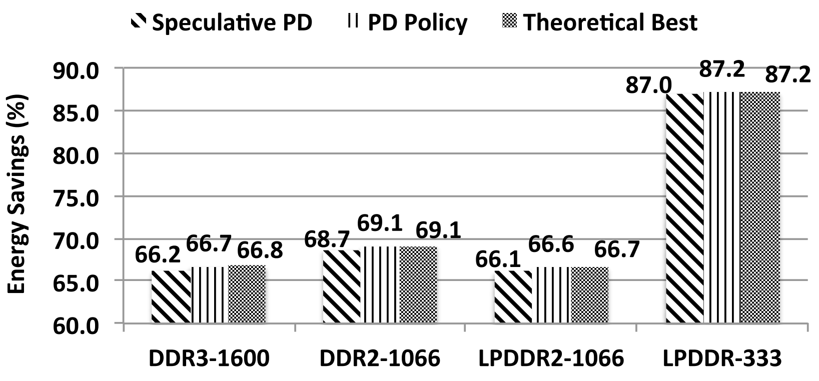


Figure 5.5: Energy Savings vs. No Power-Down

tions, when using either the selected power-management policy or the speculative policy for the 4 memories. This result highlights the marginal impact on the proposed policy on average-case execution time.

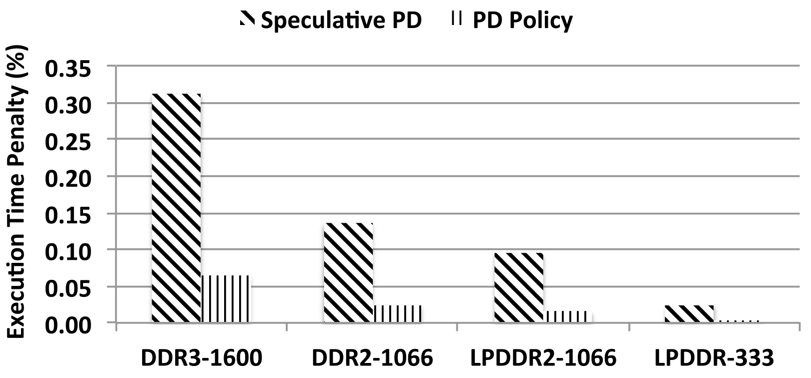


Figure 5.6: Power-Down vs. Speculative Execution Time Penalty

As depicted in the graph, the proposed power-management policy has smaller impact on the execution times of the applications compared to using the speculative power-down policy for all memories and all combinations of power-down mode and strategy. This impact on the average case performance for both the proposed policy and the speculative policy is marginal. However, as shown earlier, there is a significant difference when it comes to worst-case performance guarantees. The impact on the overall execution time depends on the number of times power-up penalty is observed. Hence, more power-ups result in higher penalty for the speculative strategy.

## 5.8 Conclusion

In this chapter, we proposed two real-time DRAM power-down strategies termed (i) conservative and (ii) aggressive. To verify their applicability to real-time systems, we first derived the original performance guarantees provided by a Latency-Rate arbiter in a real-time system. We then analyzed the impact of the proposed power-down strategies on these guarantees. From the analysis, we derived a neutrality condition, which would ensure that these strategies when used, never worsen the original performance guarantees. We finally proposed a run-time power management policy that employs either of these two strategies efficiently in combination with a DRAM power-down mode (fast-exit or slow-exit), such that the original DRAM performance guarantees are preserved. The proposed power-management policy (PD) and the two power-down strategies together achieve significant energy savings close to the theoretical best possible savings, at very low average-case performance penalty, without impacting any of the original latency and bandwidth guarantees. On the other hand, although speculative power-down has similar energy savings, its worst-case behavior is worse, as is its average case execution time.

# Chapter 6

# Conclusions and future work

In this chapter, we summarize the conclusions at the end of this thesis work and provide an insight into possible uses and extensions to this work.

## 6.1 Conclusions

This thesis targeted the two key factors defining energy-efficient use of DRAMs in embedded systems: (1) accurate power/energy estimation of DRAMs and (2) efficient power/energy optimization of DRAMs. Towards this, we proposed a high-precision power model of DRAMs referred to as 'DRAMPower' and a set of performance-neutral DRAM power-down strategies.

DRAMPower is a high-level DRAM power model that employs JEDEC-specified current metrics and performs high-precision modeling of the power consumption of different DRAM operations, state transitions and power-saving modes at the cycle-accurate level. To further improve the accuracy of DRAMPower's power/energy estimates, we derived nominal measures for the JEDEC current metrics instead of vendor-provided worst-case measures from device datasheets that pessimistically account for worst-case design-time and run-time variations. Towards this, we modified a SPICE-based circuit-level DRAM architecture and power model to accommodate the effects of these variations and derived nominal current measures under nominal operating conditions applicable to all DRAM devices with any given configuration (capacity, data-width and frequency). Besides these nominal current measures, we also proposed a generic post-manufacturing power and performance characterization methodology for DRAMs that identified the actual current estimates and optimized set of timing measures for a given DRAM device, thereby, further improving the accuracy of the power and energy estimates for that particular DRAM device.

To optimize DRAM power consumption, we proposed a set of performance-neutral DRAM power-down strategies coupled with a power management policy that for any given use-case (access granularity, page policy and memory type) achieved significant power savings without impacting its worst-case performance (bandwidth and latency) guarantees.

We verified the impact of variations on DRAM currents and four critical DRAM timing parameters (nWR, nRP, nRCD and nRTP) on 48 DDR3 devices of the same configuration against the expected impact from SPICE simulations. We derived optimal set of timings (using the performance characterization algorithm) for the fastest device at which it continued to operate successfully under worst-case run-time conditions, without increasing its energy consumption. We observed up to of 33.3% and 25.9% reduction in DRAM read and write latencies and 17.7% and 15.4% improvement in energy-efficiency.

We validated DRAMPower against a circuit-level DRAM power model and verified it against real power measurements from hardware for different DRAM operations and observed between 2-8% difference in power estimates, with an average of around 97% accuracy. We also evaluated the power-management policy and power-down strategies and observed significant energy savings (close to theoretical optimal) at very low average-case performance penalty without impacting any of the original latency and bandwidth guarantees.

## 6.2 Future work

The power modeling and optimization work proposed in this thesis open-up several new research opportunities and can be further extended to improve their applicability.

- The high-precision DRAM power model (DRAMPower) can be used to evaluate memory controller policies and strategies on row-buffer management, transaction scheduling and use of power-saving policies and modes.
- Track DRAM power and energy consumption per application for each clock cycle it accesses the memory.
- Predict DRAM power/energy consumption for an application based on its expected memory usage and optimize the same.
- Evaluate multi-channel or multi-rank DRAM management policies in terms of power/energy consumption.

The variation study and performance characterization algorithm can lead to:

1. Improved performance extraction from DRAMs that can perform better than worst-case.

2. Use of faster DRAM timings at run-time without frequency scaling to avoid the associated overhead.

Additionally, the power modeling and optimization work can be extended to:

1. Adapt DRAMPower to include memories like GDDR5, Hybrid Memory Cube (HMC), RLDRAM and FB (fully buffered) DIMMs.
2. Identify individual power-performance behavior of all the DRAMs in a given system (heterogeneity) and exploit the same to map applications.
3. Perform performance-neutral active power management. For instance, frequency scaling for LPDDRs or improve average performance and power consumption using memory access information from the processor or data locality information from the memory controller.

# Bibliography

[1] ITRS, “International technology roadmap for semiconductors.” http://www.itrs.net.

[2] ITRS, 2011. “System Drivers” - Report and Tables.

[3] Y. Huh, “Future direction of power management in mobile devices,” In Proc. ASSCC 2011.

[4] N. Goulding, J. Sampson, G. Venkatesh, S. Garcia, J. Auricchio, J. Babb, M. Taylor, and S. Swanson, “GreenDroid: A Mobile Application Processor for a Future of Dark Silicon,” In Proc. Hotchips 2010.

[5] F. Shearer, *Power Management in Mobile Devices.* Elsevier, 2007.

[6] DARPA, 2010. “Ubiquitous High Performance Computing (BAA-10-37)”.

[7] S. Borkar, “The exascale challenge,” In Proc. VLSI-DAT 2010.

[8] Nvidia, 2011. “Visual Computing Module”.

[9] Samsung, 2012. “Galaxy S4 User Manual for GT-I9500”.

[10] JEDEC, “JEDEC solid state technology association.” http://www.jedec.org.

[11] Micron, 2013. “DDR4 - Advantages of Migrating from DDR3”.

[12] JEDEC, 2012. “Migrating to LPDDR3”.

[13] JEDEC, 2011. “Wide IO SDR Standard - JESD229”.

[14] O. Vargas, “Achieve minimum power consumption in mobile memory subsystems,” March 2006. EE Times Asia.

[15] P. Kollig, C. Osborne, and T. Henriksson, “Heterogeneous multi-core platform for consumer multimedia applications,” In Proc. DATE 2009.

[16] B. Jeff, “Advances in big.little technology for power and energy savings,” 2012. White Paper, ARM.

[17] Micron, 2007. “TN-41-01: Calculating Memory System Power for DDR3”.

[18] Samsung, 2010. “LPDDR2 Power Calculation”.

[19] C. H. van Berkel, “Multi-core for mobile phones,” In Proc. DATE 2009.

[20] A. Carroll and G. Heiser, “An analysis of power consumption in a smartphone,” In Proc. USENIXATC 2010.

[21] J. Fiinn, K. I. Farkas, and J. M. Anderson, “Power and energy characterization of the ITSY pocket computer (version 1.5),” Tech. Report, 2000.

[22] T. Vogelsang, “Understanding the energy consumption of dynamic random access memories,” In Proc. MICRO 2010.

[23] H. David, C. Fallin, E. Gorbatov, U. R. Hanebutte, and O. Mutlu, “Memory power management via dynamic voltage/frequency scaling,” In Proc. ICAC 2011.

[24] I. Hur and C. Lin, “A comprehensive approach to DRAM power management,” In Proc. HPCA 2008.

[25] E. Cooper-Balis and B. Jacob, “Fine-grained activation for power reduction in DRAM,” *IEEE Micro*, vol. 30, May 2010.

[26] S. Min, H. Javaid, and S. Parameswaran, “XDRA: Exploration and optimization of last-level cache for energy reduction in DDR DRAMs,” In Proc. DAC 2013.

[27] H. Huang, K. G. Shin, C. Lefurgy, and T. Keller, “Improving energy efficiency by making DRAM less randomly accessed,” In Proc. ISLPED 2005.

[28] K. T. Malladi, I. Shaeffer, L. Gopalakrishnan, D. Lo, B. C. Lee, and M. Horowitz, “Rethinking DRAM power modes for energy proportionality,” In Proc. MICRO 2012.

[29] K. Fang and Z. Zhu, “Conservative row activation to improve memory power efficiency,” In Proc. ICS 2013.

[30] N. Aggarwal, J. Cantin, M. Lipasti, and J. Smith, “Power-efficient DRAM speculation,” In Proc. HPCA 2008.

[31] K. K.-W. Chang, D. Lee, Z. Chishti, C. Wilkerson, A. Alameldeen, Y. Kim, and O. Mutlu, "Improving DRAM performance by parallelizing refreshes with accesses," In Proc. HPCA 2014.

[32] M. Xie, D. Tong, Y. Feng, K. Huang, and X. Cheng, "Page policy control with memory partitioning for DRAM performance and power efficiency," In Proc. ISLPED 2013.

[33] X. Li, G. Jia, C. Wang, X. Zhou, and Z. Zhu1, "A scheduling of periodically active rank of DRAM to optimize power efficiency," In Proc. Highly-Reliable Power-Efficient Embedded Designs 2013.

[34] S. Mittal, "A survey of architectural techniques for DRAM power management," *Int. J. High Perform. Syst. Archit.*, vol. 4, December 2012.

[35] S. Kim, S. Kim, and Y. Lee, "DRAM power-aware rank scheduling," In Proc. ISLPED 2012.

[36] C. Ma and S. Chen, "A DRAM precharge policy based on address analysis," In Proc. DSD 2007.

[37] M. Awasthi, D. W. Nellans, R. Balasubramonian, and A. Davis, "Prediction based DRAM row-buffer management in the many-core era," In Proc. PACT 2011.

[38] D. Schmidt and N. Wehn, "DRAM power management and energy consumption: A critical assessment," In Proc. SBCCI 2009.

[39] D. Schmidt and N. Wehn, "A review of common belief on power management and power consumption," Tech. Report, 2009.

[40] K. Chandrasekar, B. Åkesson, and K. Goossens, "Improved power modeling of DDR SDRAMs," In Proc. DSD 2011.

[41] K. Chandrasekar, C. Weis, B. Åkesson, N. Wehn, and K. Goossens, "System and circuit level power modeling of energy-efficient 3d-stacked wide I/O DRAMs," In Proc. DATE 2013.

[42] F. Rawson, "MEMPOWER: A simple memory power analysis tool set," Tech. Report, 2004.

[43] A. Joshi, S. Sambamurthy, S. Kumar, , and L. John, "Power modeling of DRAMs," Tech. Report, 2004.

[44] J. Ji, C. Wang, and X. Zhou, "System-level early power estimation for memory subsystem in embedded systems," In Proc. SEC 2008.

[45] M. Gottscho, A. Kagalwalla, and P. Gupta, "Power variability in contemporary DRAMs," *Embedded Systems Letters, IEEE*, vol. 4, no. 2, 2012.

[46] C. Weis, N. Wehn, L. Igor, and L. Benini, "Design space exploration for 3D-stacked DRAMs," In Proc. DATE 2011.

[47] C. Weis, I. Loi, L. Benini, and N. Wehn, "Exploration and optimization of 3-D integrated DRAM subsystems," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 32, no. 4, 2013.

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

[49] P. Zheng and L. Ni, *Smart Phone and Next Generation Mobile Computing.* Morgan Kaufmann Publishers Inc., 2005.

[50] B. Jacob, S. Ng, and D. Wang, *Memory Systems: Cache,* DRAM*, Disk.* Morgan Kaufmann Publishers Inc., 2007.

[51] S. Rixner, W. J. Dally, U. J. Kapasi, P. Mattson, and J. D. Owens, "Memory access scheduling," *SIGARCH Comput. Archit. News*, vol. 28, May 2000.

[52] D. Wang, B. Ganesh, N. Tuaycharoen, K. Baynes, A. Jaleel, and B. Jacob, "DRAMsim: A memory system simulator," *SIGARCH Comput. Archit. News*, vol. 33, November 2005.

[53] K. Chandrasekar, C. Weis, B. Åkesson, N. Wehn, and K. Goossens, "Towards variation-aware system-level power estimation of DRAMs: An empirical approach," In Proc. DAC 2013.

[54] K. Chandrasekar, S. Goossens, K. Martijn, C. Weis, B. Åkesson, N. Wehn, and K. Goossens, "Exploiting expendable process-margins in DRAMs for run-time performance optimization," In Proc. DATE 2014.

[55] V. Delaluz, M. Kandemir, N. Vijaykrishnan, A. Sivasubramaniam, and M. J. Irwin, "Hardware and software techniques for controlling DRAM power modes," *IEEE Trans. Comput.*, vol. 50, November 2001.

[56] V. Delaluz, A. Sivasubramaniam, M. Kandemir, N. Vijaykrishnan, and M. J. Irwin, "Scheduler-based DRAM energy management," In Proc. DAC 2002.

[57] G. Thomas, K. Chandrasekar, B. Åkesson, B. Juurlink, and K. Goossens, "A predictor-based power-saving policy for DRAM memories," In Proc. DSD 2012.

[58] Y. Joo, Y. Choi, and H. Shim, "Energy exploration and reduction of SDRAM memory systems," In Proc. DAC 2002.

[59] H. David, E. Gorbatov, U. R. Hanebutte, R. Khanna, and C. Le, "RAPL: Memory power estimation and capping," In Proc. ISLPED 2010.

[60] Q. Deng, D. Meisner, L. Ramos, T. F. Wenisch, and R. Bianchini, "Memscale: Active low-power modes for main memory," *SIGPLAN Not.*, vol. 47, April 2012.

[61] Q. Deng, D. Meisner, A. Bhattacharjee, T. F. Wenisch, and R. Bianchini, "Multiscale: Memory system DVFS with multiple memory controllers," In Proc. ISLPED 2012.

[62] B. Akesson and K. Goossens, "Architectures and modeling of predictable memory controllers for improved system integration," In Proc. DATE 2011.

[63] M. Paolieri, E. Quiñones, and F. J. Cazorla, "Timing effects of ddr memory systems in hard real-time multicore architectures: Issues and solutions," *ACM Trans. Embedded Comput. Syst.*, vol. 12, no. 1, 2013.

[64] A. Burchardt, E. Hekstra-Nowacka, and A. Chauhan, "A real-time streaming memory controller," In Proc. DATE 2005.

[65] S. Edwards, S. Kim, E. Lee, I. Liu, H. Patel, and M. Schoeberl, "A disruptive computer design idea: Architectures with repeatable timing," In Proc. ICCD 2009.

[66] C. Pitter, "Time-predictable memory arbitration for a java chip-multiprocessor," In Proc. JTRES 2008.

[67] J. Reineke, I. Liu, H. D. Patel, S. Kim, and E. A. Lee, "PRET DRAM controller: Bank privatization for predictability and temporal isolation," In Proc. CODES+ISSS 2011.

[68] K. Chandrasekar, C. Weis, Y. Li, B. Akesson, N. Wehn, and K. Goossens, "DRAMPower: Open-source DRAM power & energy estimation tool." www.es.ele.tue.nl/drampower.

[69] D. Burger and T. M. Austin, "The SimpleScalar tool set, version 2.0," *SIGARCH Comput. Archit. News*, vol. 25, June 1997.

[70] A. Hansson, K. Goossens, M. Bekooij, and J. Huisken, "CoMPSoC: A template for composable and predictable multi-processor system on chips," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 14, January 2009.

[71] Y. Cao and L. T. Clark, "Mapping statistical process variations toward circuit performance variability: An analytical modeling approach," In Proc. DAC 2005.

[72] S. Bhardwaj, S. Vrudhula, P. Ghanta, and Y. Cao, "Modeling of intra-die process variations for accurate analysis and optimization of nano-scale circuits," In Proc. DAC 2006.

[73] S. Borkar, T. Karnik, S. Narendra, J. Tschanz, A. Keshavarzi, and V. De, “Parameter variations and impact on circuits and microarchitecture,” In Proc. DAC 2003.

[74] K. Bowman, A. Alameldeen, S. Srinivasan, and C. Wilkerson, “Impact of die-to-die and within-die parameter variations on the throughput distribution of multi-core processors,” In Proc. ISLPED 2007.

[75] X. Liang and D. Brooks, “Mitigating the impact of process variations on processor register files and execution units,” In Proc. MICRO 2006.

[76] Intel. “Memory 3-sigma Power Analysis Methodology”.

[77] S. Desai, S. Roy, and K. Chakraborty, “Process variation aware DRAM design using block based adaptive body biasing algorithm,” In Proc. ISQED 2012.

[78] L. A. D. Bathen, N. D. Dutt, A. Nicolau, and P. Gupta, “Vamv: Variability-aware memory virtualization,” In Proc. DATE 2012.

[79] L. A. D. Bathen, M. Gottscho, N. Dutt, A. Nicolau, and P. Gupta, “ViPZonE: OS-level memory variability-driven physical address zoning for energy savings,” In Proc. CODES+ISSS ’12.

[80] T. Schloesser, F. Jakubowski, J. v.Kluge, A. Graham, S. Slesazeck, M. Popp, P. Baars, K. Muemmler, P. Moll, K. Wilson, A. Buerke, D. Koehler, J. Radecker, E. Erben, U. Zimmermann, T. Vorrath, B. Fischer, G. Aichmayr, R. Agaiby, W. Pamler, T. Schuster, W. Bergner, and W. Mueller, “$6f^2$ buried wordline DRAM cell for 40nm and beyond,” In Proc. IEDM 2008.

[81] Y. Ye, T. Liu, M. Chen, S. Nassif, and Y. Cao, “Statistical modeling and simulation of threshold variation under random dopant fluctuations and line-edge roughness,” *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 19, June 2011.

[82] C.-H. Lin, M. Dunga, D. Lu, A. Niknejad, and C. Hu, “Statistical compact modeling of variations in nano MOSFETs,” In Proc. VLSI-TSA 2008.

[83] C. Lee, M. Potkonjak, and W. H. Mangione-Smith, “Mediabench: A tool for evaluating and synthesizing multimedia and communicatons systems,” In Proc. MICRO 1997.

[84] A. Keshavarzi, J. Tschanz, S. Narendra, V. De, W. Daasch, K. Roy, M. Sachdev, and C. Hawkins, “Leakage and process variation effects in current testing on future CMOS circuits,” *Design Test of Computers, IEEE*, vol. 19, Sep 2002.

[85] S. Herbert and D. Marculescu, “Variation-aware dynamic voltage/frequency scaling,” In Proc. HPCA 2009.

[86] K. Chandrasekar, B. Akesson, and K. Goossens, “Run-time power-down strategies for real-time SDRAM memory controllers,” In Proc. DAC 2012.

[87] J. Vollrath, “Signal margin analysis for DRAM sense amplifiers,” In Proc. DELTA 2002.

[88] Micron, 2010. “TN-00-18: Temperature Uprating on Semiconductors”.

[89] S. Nassif, “Delay variability: sources, impacts and trends,” In Proc. ISSCC 2000.

[90] I.-G. Kim, S.-K. Choi, J.-H. Choi, and J.-S. Park, “DRAM reliability characterization by using dynamic operation stress in wafer burn-in mode,” In Proc. Reliability Physics Symposium 2003.

[91] T. Sato, T. Hayashida, and K. Yano, “Dynamically reducing overestimated design margin of multicores,” In Proc. HPCS 2012.

[92] Micron, 2009. “512MB (x64, SR) 204-Pin DDR3 SDRAM SODIMM - MT4JSF6464H”.

[93] Z. Al-Ars, S. Hamdioui, and A. J. Van de Goor, “Space of DRAM fault models and corresponding testing,” In Proc. DATE 2006.

[94] A. Van De Goer and J. De Neef, “Industrial evaluation of DRAM tests,” In Proc. DATE 1999.

[95] D. Stiliadis and A. Varma, “Latency-rate servers: A general model for analysis of traffic scheduling algorithms,” *IEEE/ACM Trans. Netw.*, vol. 6, October 1998.

[96] S. Goossens, B. Akesson, and K. Goossens, “Conservative open-page policy for mixed time-criticality memory controllers,” In Proc. DATE 2013.

[97] S. Goossens, T. Kouters, B. Akesson, and K. Goossens, “Memory-map selection for firm real-time SDRAM controllers,” In Proc. DATE 2012.

[98] Micron, 2004. “DDR2 DRAM 1Gb Data Sheet”.

[99] Micron, 2010. “1Gb: X4, X8, X16 DDR3 DRAM”.

[100] Micron, 2012. “2Gb: X16, X32 Mobile LPDDR2 SDRAM S4”.

[101] S. Thoziyoor, N. Muralimanohar, J. H. Ahn, and N. P. Jouppi, 2008. “CACTI 5.1”, Technical Report HP Laboratories.

[102] JEDEC, 2009. “DDR2 SDRAM Standard - JESD79-2F”.

[103] JEDEC, 2010. “DDR3 SDRAM Standard - JESD79-3E”.

[104] “NGSPICE.” http://ngspice.sourceforge.net/.

[105] “BSIM4.6.” http://www-device.eecs.berkeley.edu/bsim/.

[106] “Predictive technology model.” http://ptm.asu.edu/.

[107] ITRS, 2011. “Design” - Report and Tables.

[108] ITRS, 2011. “Modeling & Simulation” - Report and Tables.

[109] “Memtest86.” http://www.memtest86.com/.

[110] Xilinx, 2010. “Memory Interface Generator and Physical Layer (PHY)” - UG086.

[111] Multicomp. “Peltier Cooler - 4W” - MCPF-031-10-25.

[112] Micron, 2008. “TN-00-08: Thermal Applications”.

[113] “Arduino platform.” http://www.arduino.cc/.

[114] JEDEC, 2010. “204-pin DDR3 SDRAM Unbuffered SODIMM Design Specification” - MO-268.

[115] JUJET, “JET-5466 extender.” http://www.eztest.com.tw/.

[116] H. Shah, A. Knoll, and B. Akesson, “Bounding SDRAM interference: Detailed analysis vs. latency-rate analysis,” In Proc. DATE 2013.

[117] B. Akesson and K. Goossens, *Memory Controllers for Real-Time Embedded Systems*. Springer, 2011.

[118] B. Akesson, L. Steffens, E. Strooisma, and K. Goossens, “Real-Time Scheduling Using Credit-Controlled Static-Priority Arbitration,” In Proc. RTCSA 2008.

[119] K. Kim and D. Choi, “Delay stage-interweaved analog DLL/PLL,” 2008. US Patent 7,447,106.

[120] A. Hansson, N. Agarwal, A. Kolli, T. Wenisch, and A. N. Udipi, “Simulating DRAM controllers for future system architecture exploration,” In Proc. ISPASS 2014.

[121] T. Bandyopadhyay, R. Chatterjee, D. Chung, M. Swaminathan, and R. Tummala, “Electrical modeling of through silicon and package vias,” In Proc. 3DIC 2009.

[122] S.-I. Park and I.-C. Park, “History-based memory mode prediction for improving memory performance,” In Proc. ISCAS 2003.

[123] 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 Proc. ISCA 2013.

[124] P. Nair, C.-C. Chou, and M. K. Qureshi, “A case for refresh pausing in DRAM memory systems,” In Proc. HPCA 2013.

[125] S. Liu, K. Pattabiraman, T. Moscibroda, and B. G. Zorn, “Flikker: Saving DRAM refresh-power through critical data partitioning,” In Proc. ASPLOS 2011.

[126] C. Weis, I. Loi, L. Benini, and N. Wehn, “An energy efficient DRAM subsystem for 3d integrated socs,” In Proc. DATE 2012.

[127] M. Ghosh and H.-H. S. Lee, “Smart refresh: An enhanced memory controller design for reducing energy in conventional and 3d die-stacked DRAMs,” In Proc. MICRO 2007.

[128] N. P. Jouppi, A. B. Kahng, N. Muralimanohar, and V. Srinivas, “CACTI-IO: CACTI with off-chip power-area-timing models,” In Proc. ICCAD 2012.

[129] X. Li, G. Jia, C. Wang, X. Zhou, and Z. Zhu, “A scheduling of periodically active rank of DRAM to optimize power efficiency,” In Proc. HARSH 2013.

[130] R. Dennard, F. Gaensslen, V. Rideout, E. Bassous, and A. LeBlanc, “Design of ion-implanted mosfet’s with very small physical dimensions,” *Solid-State Circuits, IEEE Journal of*, vol. 9, October 1974.

[131] H. Esmaeilzadeh, E. Blem, R. St. Amant, K. Sankaralingam, and D. Burger, “Dark silicon and the end of multicore scaling,” In Proc. ISCA 2011.

[132] Y. Li, B. Akesson, and K. Goossens, “Dynamic command scheduling for real-time memory controllers,” In Proc. ECRTS 2014.

[133] Z. P. Wu, Y. Krish, and R. Pellizzoni, “Worst case analysis of dram latency in multi-requestor systems,” In Proc. RTSS 2013.

# Chapter A

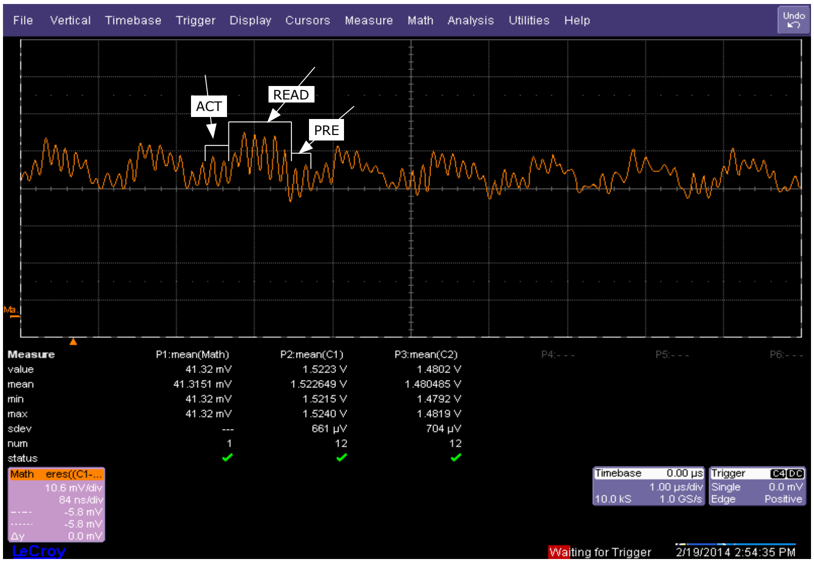
# Power Measurements

## A.1 Current Measures - Datasheet vs. Hardware

Having presented the current measurement plots for $I_{DD0}$ and $I_{DD2P}$ currents in Chapter 4, in this section in Appendix A, we provide the measurement plots for the remaining DRAM currents (defined in Chapter 2, Section 2.2). As before, we classify the currents as partial or complete depending on the operations involved and the current measures can be obtained using the the mean voltage drop value in the measurement plots from: (1) column P1 for the complete currents and (2) the Math ERES (Enhanced Resolution) box for the partial currents.

**[1] $I_{DD1R}$ current - Complete Current**

The $I_{DD1R}$ test loop measures current across ACT, RD and PRE commands sent to one bank, with the other banks retained in precharged state. The time period between the ACT and RD commands is maintained as $nRCD$ cycles and between RD and PRE commands as $nRAS - nRCD$ cycles. $nRC$ cycles after the ACT command, the test is repeated on the same bank but on a different row with different data to introduce flipping of bits on the data bus. The data set is defined by JEDEC. Hence, before the test is initiated the data is written to the specific two rows in all banks. After reading from the two rows in a bank, the test switches to the next bank and repeats itself over all the banks in the DRAM, with one of the banks and the same two rows in that bank accessed in a loop iteration. Since this test also repeats itself without requiring any additional operations (after the initial writing operations), the measurement observed is a complete current. Note that when reading data from the DIMM, the power measurements using the JET-5466 extender board include the power consumption due to the I/O pins and associated circuitry. This is given by Micron's estimates on I/O power consumption [17], and must be subtracted from the measured current (based on mean voltage drop), since the datasheet measures do not include these. I/O power during a read operation is computed in terms of a current of 176mA per 64-bits of data read out from the DRAM [17] at the supply voltage (1.5V). This test loop is depicted in Figure A.1. As shown in the figure, the test loop repeats itself over different banks and the average current measure is identified as $I_{DD1R}$ current.



| Measure | P1:mean(Math) | P2:mean(C1) | P3:mean(C2) | P4:- - - | P5:- - - | P6:- - - |
|---|---|---|---|---|---|---|
| value | 41.32 mV | 1.5223 V | 1.4802 V | | | |
| mean | 41.3151 mV | 1.522649 V | 1.480485 V | | | |
| min | 41.32 mV | 1.5215 V | 1.4792 V | | | |
| max | 41.32 mV | 1.5240 V | 1.4819 V | | | |
| sdev | --- | 661 μV | 704 μV | | | |
| num | 1 | 12 | 12 | | | |
| status | ✓ | ✓ | ✓ | | | |

```
Math   eres((C1-...
        10.6 mV/div
         84 ns/div
----     -5.8 mV
------   -5.8 mV
Δy        0.0 mV
```

```
Timebase     0.00 μs   Trigger      C4 DC
           1.00 μs/div Single      0.0 mV
10.0 kS     1.0 GS/s   Edge      Positive
```

```
Waiting for Trigger     2/19/2014 2:54:35 PM
```

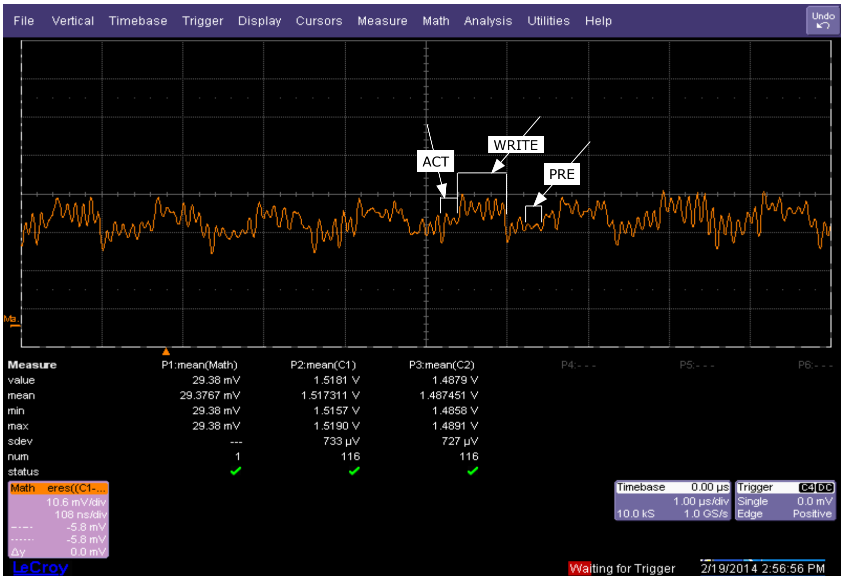
Figure A.1: Measuring $I_{DD1R}$

**[2] $I_{DD1W}$ current - Complete Current**

The $I_{DD1W}$ test measures current across ACT, WR and PRE commands sent to one bank, with the other banks retained in precharged state. The time period between the ACT and WR commands is maintained as $nRCD$ cycles and between WR and PRE commands as $nWL + nWR + BL/2$ cycles [103]. $nRP$ cycles after the PRE command, the test is repeated on the same bank but on a different row, with different data being written (specified by JEDEC), as in the case of $I_{DD1R}$ current. After writing into the two rows in a bank, the test switches to the next bank and repeats itself over all the banks in the DRAM, with one bank active at a time. Since this test also repeats itself without requiring any additional operations, the measurement observed is a complete current. Note that, the $I_{DD1W}$ current is not a JEDEC standard measure, however, its reference measures can be calculated by substituting write current ($I_{DD4W}$) instead of read current ($I_{DD4R}$) in $I_{DD1}$ current and the timings scaled as per writing to precharge timing requirements [103].This test loop is depicted in Figure A.2.

**[3] $I_{DD2N}$ current - Partial Current - Requires Precharging**

The $I_{DD2N}$ test loop measures current in the precharged state, after a set of ACT and PRE commands are sent to one bank, with the other banks retained in precharged state. The time period between the ACT and PRE commands is maintained as $nRAS$ cycles. The time period between PRE and the end of the test pattern is kept at 2000 clock cycles (sufficiently long compared to $nRAS$), which retains the DRAM in the precharged state. The test pattern then switches to the next bank and repeats itself over all the banks in the DRAM, with one bank accessed at a time.



```
Measure   P1:mean(Math)  P2:mean(C1)  P3:mean(C2)   P4:- - -   P5:- - -   P6:- - -
value         29.38 mV     1.5181 V     1.4879 V
mean        29.3767 mV   1.517311 V   1.487451 V
min           29.38 mV     1.5157 V     1.4858 V
max           29.38 mV     1.5190 V     1.4891 V
sdev               ---       733 µV       727 µV
num                  1          116          116
status               ✓            ✓            ✓

Math  eres((C1-...        Timebase    0.00 µs   Trigger      C4 DC
        10.6 mV/div                1.00 µs/div   Single      0.0 mV
        108 ns/div        10.0 kS     1.0 GS/s   Edge      Positive
- - -      -5.8 mV
......     -5.8 mV
Δy          0.0 mV

Waiting for Trigger   2/19/2014 2:56:56 PM
```

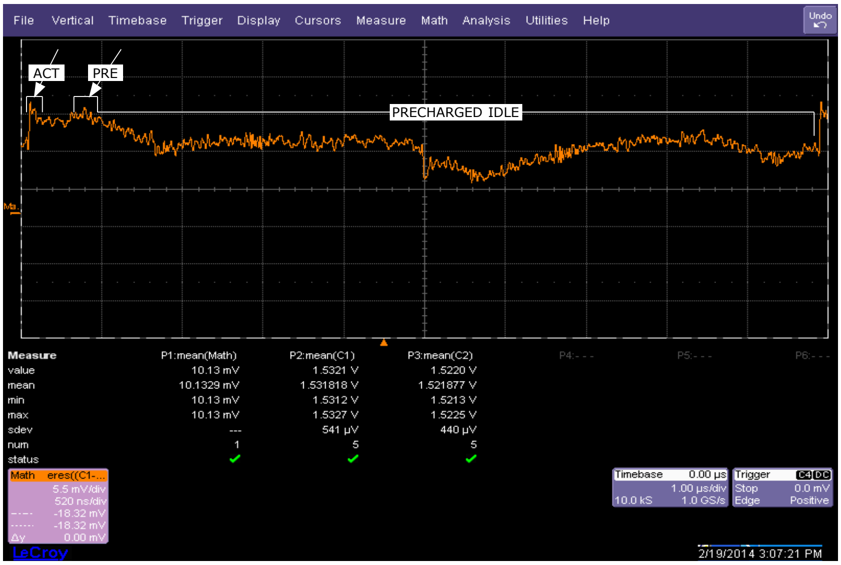
Figure A.2: Measuring $I_{DD1W}$

Since this test requires activation and precharging operations to retain the DRAM in the precharged state (as shown in Figure A.3), we use density plots to identify the voltage measure for the longest retained state (precharged state) as depicted in Figure A.4. This voltage measure can be observed by the dotted line across the region in red (density identifier).

[4] $\boldsymbol{I_{DD3N}}$ current - Partial Current - Requires Activation

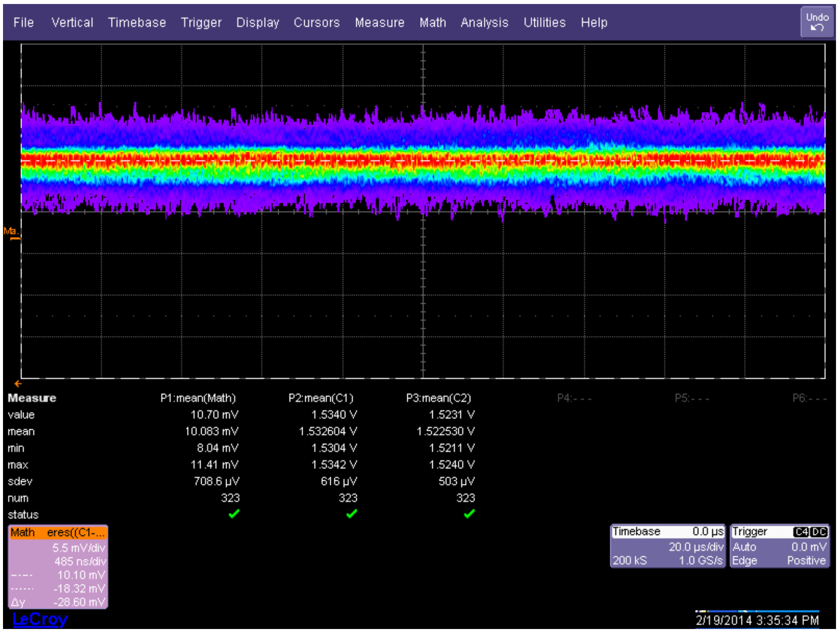
The $I_{DD3N}$ test measures current after an ACT command is sent to one bank, with the other banks retained in precharged state. Subsequently, a PRE command is issued to the same bank after 2000 clock cycles, hence retaining the bank in the active standby state for a long period. Then, the bank is allowed to precharged over $nRP$ clock cycles after which, the test pattern switches to the next bank and repeats itself over all the banks in the DRAM, with one bank accessed at a time. Since this test requires activation and precharging operations to retain the DRAM in the active standby state (as shown in Figure A.5), we use density plots to identify the voltage measure for the longest retained state (active standby state), as depicted in Figure A.6.

According to JEDEC, the $I_{DD3N}$ current should the same irrespective of the number of active banks (one or all). To verify this claim, we modified the $I_{DD3N}$ test loop to first activate all 8 banks in the DRAM (instead of just one) and then retained the memory in the active standby state for a long period. The corresponding current measure for this modified test loop is depicted in Figure A.7. From our experiment, we observed a marginal difference in the $I_{DD3N}$ current measure when all 8 banks are active, as shown in Figure A.8.



```
Measure   P1:mean(Math)   P2:mean(C1)   P3:mean(C2)   P4:- - -   P5:- - -   P6:- - -
value         10.13 mV      1.5321 V      1.5220 V
mean        10.1329 mV    1.531818 V    1.521877 V
min           10.13 mV      1.5312 V      1.5213 V
max           10.13 mV      1.5327 V      1.5225 V
sdev               ---        541 µV        440 µV
num                  1             5             5
status               ✓             ✓             ✓

Math  eres((C1-...        Timebase    0.00 µs   Trigger        C4 DC
         5.5 mV/div                1.00 µs/div  Stop          0.0 mV
         520 ns/div       10.0 kS    1.0 GS/s   Edge        Positive
        -18.32 mV
        -18.32 mV
Δy        0.00 mV                               2/19/2014 3:07:21 PM
```

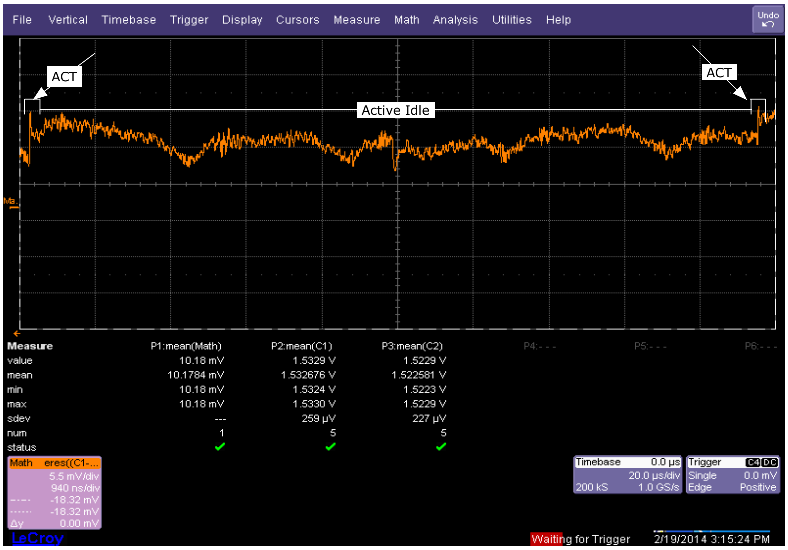
Figure A.3: Measuring $I_{DD2N}$ after Precharging



```
Measure   P1:mean(Math)   P2:mean(C1)   P3:mean(C2)   P4:- - -   P5:- - -   P6:- - -
value         10.70 mV      1.5340 V      1.5231 V
mean        10.083 mV     1.532604 V    1.522530 V
min           8.04 mV       1.5304 V      1.5211 V
max          11.41 mV       1.5342 V      1.5240 V
sdev         708.6 µV         616 µV        503 µV
num               323            323           323
status              ✓              ✓             ✓

Math  eres((C1-...        Timebase     0.0 µs   Trigger        C4 DC
         5.5 mV/div                20.0 µs/div  Auto          0.0 mV
         485 ns/div       200 kS     1.0 GS/s   Edge        Positive
         10.10 mV
        -18.32 mV
Δy      -28.60 mV                               2/19/2014 3:35:34 PM
```

Figure A.4: Measuring $I_{DD2N}$ - Density Plot

To be conservative, we chose to employ this measure (with 8 banks open) for the reference $I_{DD3N}$ current.


File Vertical Timebase Trigger Display Cursors Measure Math Analysis Utilities Help
Undo
ACT
ACT
Active Idle
Ma
LeCroy

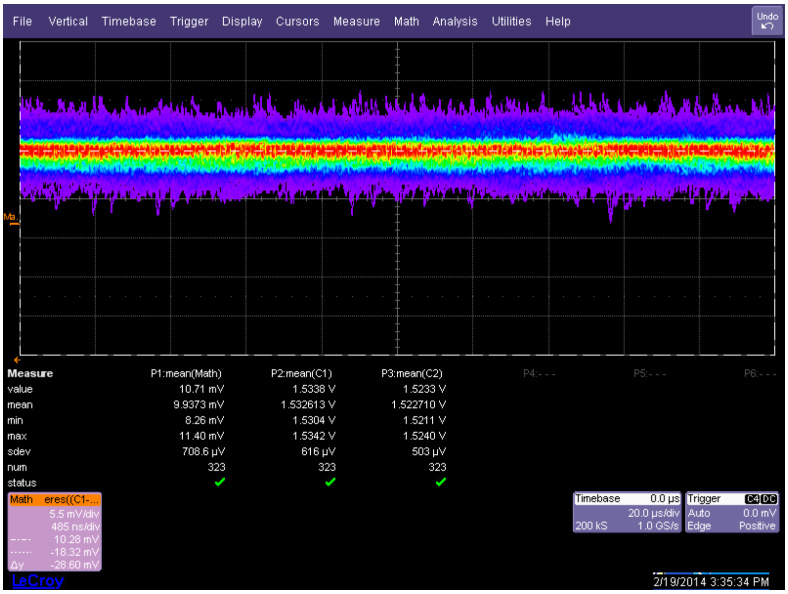
```
Measure      P1:mean(Math)   P2:mean(C1)   P3:mean(C2)   P4:---   P5:---   P6:---
value           10.18 mV       1.5329 V      1.5229 V
mean          10.1784 mV     1.532676 V    1.522581 V
min             10.18 mV       1.5324 V      1.5223 V
max             10.18 mV       1.5330 V      1.5229 V
sdev                 ---         259 μV        227 μV
num                    1              5             5
status                 ✓              ✓             ✓
```

```
Math   eres((C1-...
          5.5 mV/div
          940 ns/div
----     -18.32 mV
......   -18.32 mV
Δy         0.00 mV
```

```
Timebase      0.0 μs    Trigger    C4 DC
        20.0 μs/div     Single    0.0 mV
200 kS     1.0 GS/s     Edge    Positive
```

Waiting for Trigger 2/19/2014 3:15:24 PM

Figure A.5: Measuring $I_{DD3N}$ with 1 Active Bank


File Vertical Timebase Trigger Display Cursors Measure Math Analysis Utilities Help
Undo
Ma
LeCroy

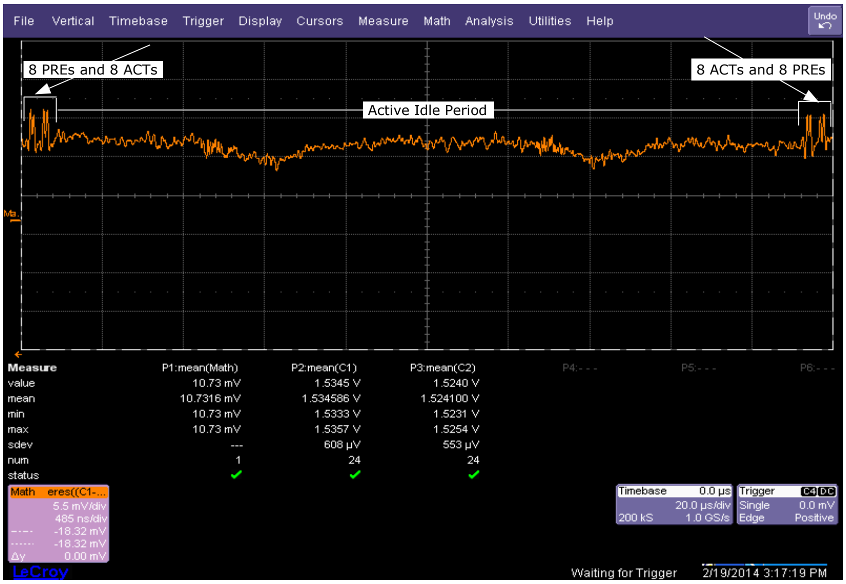
```
Measure      P1:mean(Math)   P2:mean(C1)   P3:mean(C2)   P4:---   P5:---   P6:---
value           10.71 mV       1.5338 V      1.5233 V
mean           9.9373 mV     1.532613 V    1.522710 V
min              8.26 mV       1.5304 V      1.5211 V
max             11.40 mV       1.5342 V      1.5240 V
sdev            708.6 μV         616 μV        503 μV
num                  323            323           323
status                 ✓              ✓             ✓
```

```
Math   eres((C1-...
          5.5 mV/div
          485 ns/div
----      10.28 mV
......   -18.32 mV
Δy       -28.60 mV
```

```
Timebase      0.0 μs    Trigger    C4 DC
        20.0 μs/div     Auto      0.0 mV
200 kS     1.0 GS/s     Edge    Positive
```

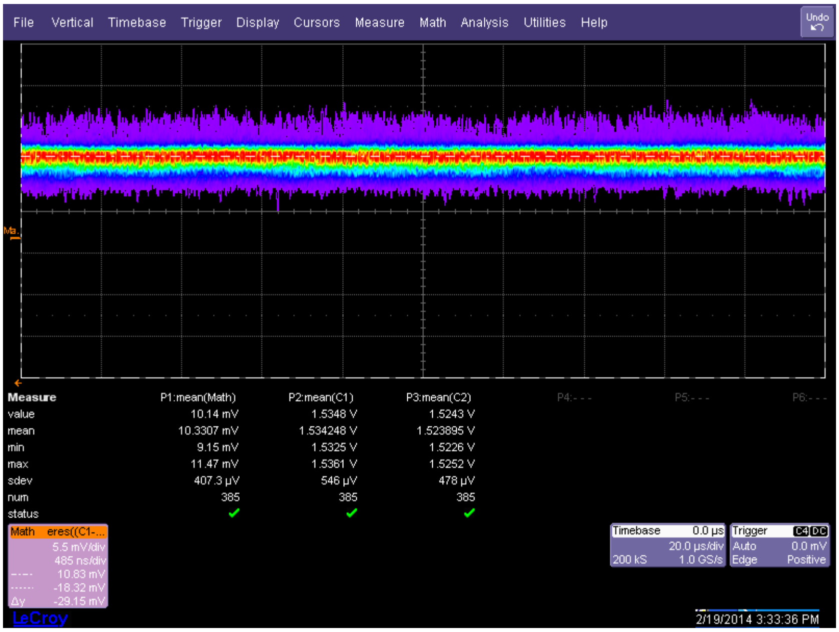
2/19/2014 3:35:34 PM

Figure A.6: Measuring $I_{DD3N}$ with 1 Active Bank - Density Plot



```
Measure    P1:mean(Math)   P2:mean(C1)   P3:mean(C2)   P4:- - -   P5:- - -   P6:- - -
value          10.73 mV      1.5345 V      1.5240 V
mean         10.7316 mV    1.534586 V    1.524100 V
min            10.73 mV      1.5333 V      1.5231 V
max            10.73 mV      1.5357 V      1.5254 V
sdev                ---        608 µV        553 µV
num                   1            24            24
status                ✓             ✓             ✓

Math  eres((C1-...        Timebase      0.0 µs   Trigger     C4 DC
      5.5 mV/div                   20.0 µs/div   Single     0.0 mV
      485 ns/div          200 kS     1.0 GS/s    Edge     Positive
      -18.32 mV
      -18.32 mV
Δy     0.00 mV
                          Waiting for Trigger   2/19/2014 3:17:19 PM
```

Figure A.7: Measuring $I_{DD3N}$ with 8 Active Banks



```
Measure    P1:mean(Math)   P2:mean(C1)   P3:mean(C2)   P4:- - -   P5:- - -   P6:- - -
value          10.14 mV      1.5348 V      1.5243 V
mean         10.3307 mV    1.534248 V    1.523895 V
min             9.15 mV      1.5325 V      1.5226 V
max            11.47 mV      1.5361 V      1.5252 V
sdev           407.3 µV        546 µV        478 µV
num                 385           385           385
status                ✓             ✓             ✓

Math  eres((C1-...        Timebase      0.0 µs   Trigger     C4 DC
      5.5 mV/div                   20.0 µs/div   Auto       0.0 mV
      485 ns/div          200 kS     1.0 GS/s    Edge     Positive
       10.83 mV
      -18.32 mV
Δy    -29.15 mV
                                                2/19/2014 3:33:36 PM
```

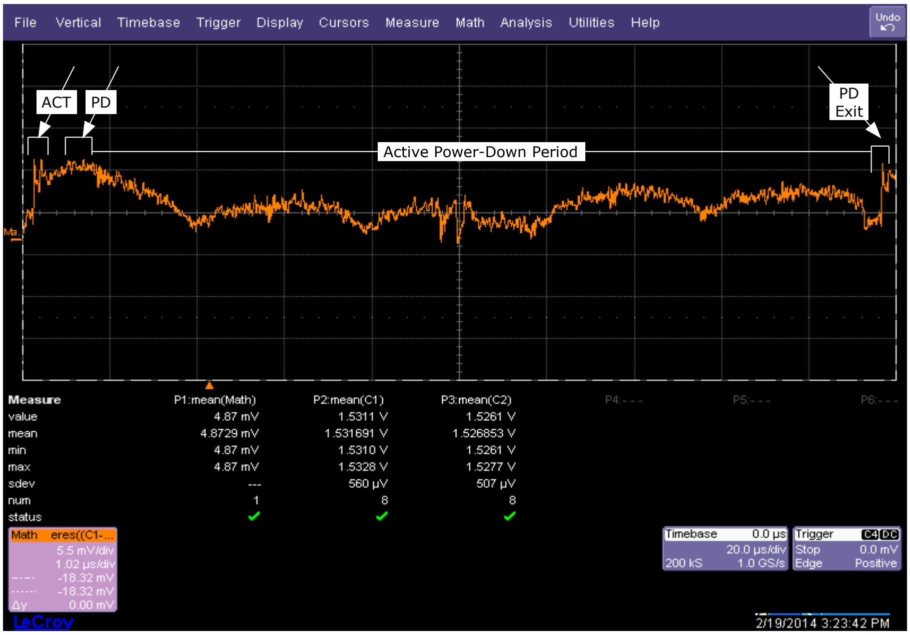
Figure A.8: Measuring $I_{DD3N}$ with 8 Active Banks - Density Plot

[5] $\boldsymbol{I_{DD3P}}$ current - Partial Current - Requires Activation

The $I_{DD3P}$ test loop measures current in the active power-down state. This test starts with an ACT command sent to a bank and then a power-down is initiated. The time period between the ACT and power-down commands is maintained as $nRAS$ cycles to allow the bank to be activated, while the other banks are retained in the precharged state, since only one bank has to be active when entering the power-down mode. The DRAM is retained in the power-down state for 2000 clock cycles (sufficiently long compared to activating time and power-up time). After this period, a power-up is issued, followed by a PRE command. The time period between the power-up and PRE command is maintained as $nXP$ clock cycles and the test pattern ends after further $nRP$ cycles to allow the precharge to complete. The test pattern then switches to the next bank starting again with an ACT command and repeats itself over all the banks in the DRAM.

Since this test requires activation, precharging and power-up operations besides retaining the DRAM in the active power-down state (as shown in Figure A.9), we use density plots to identify the voltage measure for the longest retained state (active power-down state, depicted in Figure A.10.

As can be observed from the density plot in Figure A.10, the active power-down state is a low power consuming state compared to the active idle/standby state in Figure A.8.



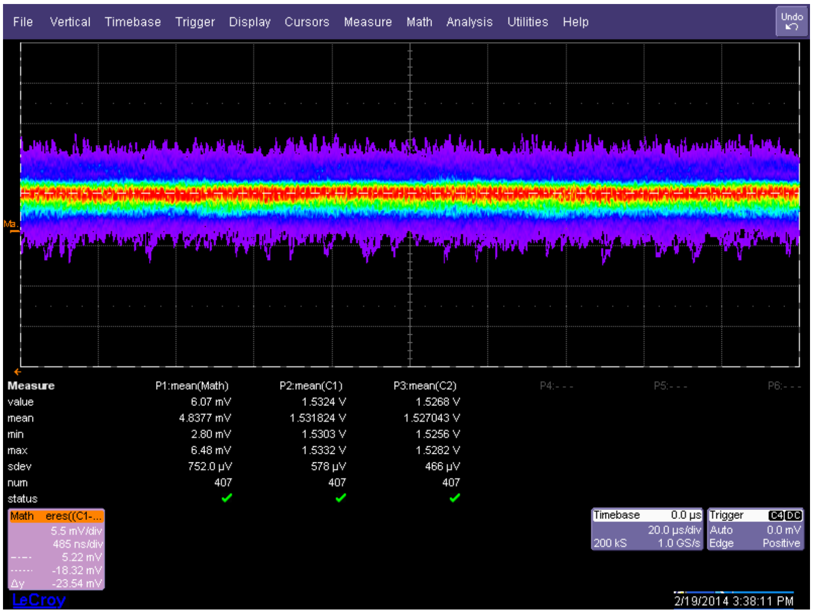
| Measure | P1:mean(Math) | P2:mean(C1) | P3:mean(C2) | P4:- - - | P5:- - - | P6:- - - |
|---|---|---|---|---|---|---|
| value | 4.87 mV | 1.5311 V | 1.5261 V | | | |
| mean | 4.8729 mV | 1.531691 V | 1.526853 V | | | |
| min | 4.87 mV | 1.5310 V | 1.5261 V | | | |
| max | 4.87 mV | 1.5328 V | 1.5277 V | | | |
| sdev | --- | 560 µV | 507 µV | | | |
| num | 1 | 8 | 8 | | | |
| status | ✓ | ✓ | ✓ | | | |

```
Math    eres((C1-...
          5.5 mV/div
         1.02 µs/div
-·-·-    -18.32 mV
......   -18.32 mV
Δy         0.00 mV
```

```
Timebase      0.0 µs   Trigger     C4 DC
         20.0 µs/div   Stop       0.0 mV
200 kS     1.0 GS/s    Edge     Positive
```

2/19/2014 3:23:42 PM

Figure A.9: Measuring $I_{DD3P}$



```
Measure   P1:mean(Math)   P2:mean(C1)   P3:mean(C2)   P4:- - -   P5:- - -   P6:- - -
value          6.07 mV      1.5324 V      1.5268 V
mean         4.8377 mV    1.531824 V    1.527043 V
min            2.80 mV      1.5303 V      1.5256 V
max            6.48 mV      1.5332 V      1.5282 V
sdev          752.0 µV        578 µV        466 µV
num                407           407           407
status               ✓             ✓             ✓

Math   eres((C1-...          Timebase      0.0 µs   Trigger     C4 DC
          5.5 mV/div                  20.0 µs/div   Auto       0.0 mV
          485 ns/div         200 kS      1.0 GS/s   Edge     Positive
----       5.22 mV
......   -18.32 mV
Δy       -23.54 mV

2/19/2014 3:38:11 PM
```

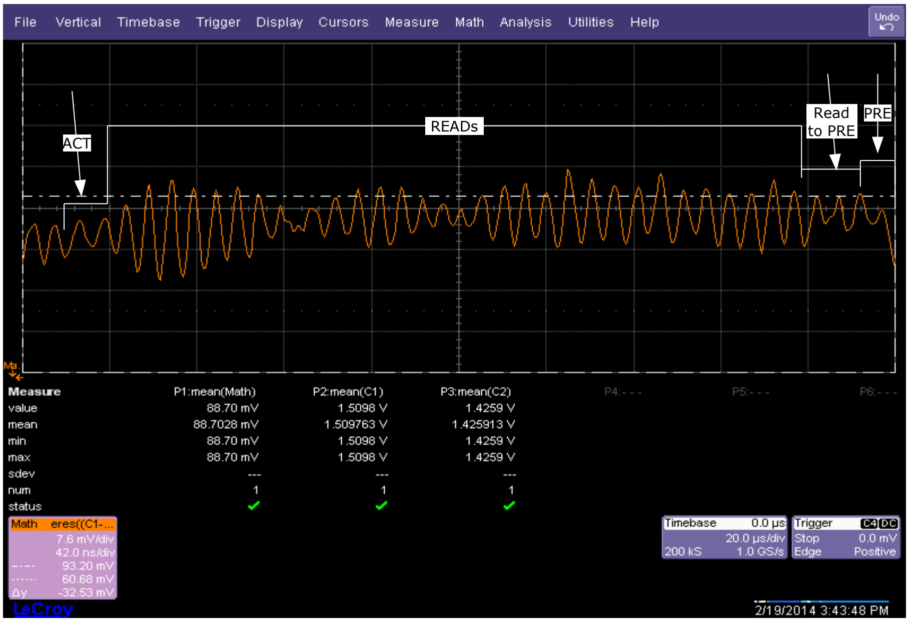
Figure A.10: Measuring $I_{DD3P}$ - Density Plot

[6] $\boldsymbol{I_{DD4R}}$ current - Complete Current - Read dominant test

The $I_{DD4R}$ test loop measures current across several RD commands sent to one bank, with the other banks retained in precharged state. Before issuing several read commands, the row needs to be activated, hence an ACT command is issued first. The time period between the ACT and first RD command is maintained as $nRCD$ cycles. Thereafter a burst of 64 RD commands are sent to the same open row to consecutive columns. After these RD commands, a PRE command is issued to close the bank after $nRTP$ clock cycles. After the precharging operation finishes (i.e. $nRP$ cycles later), the test pattern ends. The test then switches to the next bank and repeats itself over all the banks in the DRAM, with one bank active at a time. Before the test is instantiated the data is written to the JEDEC-specified rows in all banks. Since this test is dominated by Reads, and only one activate and precharge, we subtract the activation-precharge energy calculated in the $I_{DD0}$ test in Figure 4.4 and the I/O power consumption (only for Reads) from the total observed energy in Figure A.11, to obtain the $I_{DD4R}$ current measure.

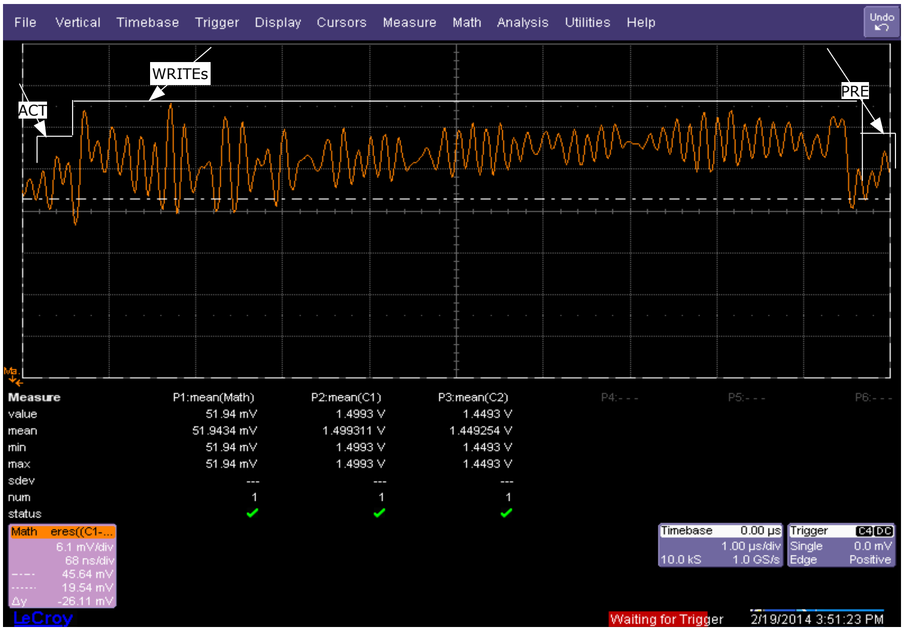
[7] $\boldsymbol{I_{DD4W}}$ current - Complete Current - Write dominant test

This is the same as $I_{DD4R}$ test loop except with issuing Writes instead of Reads. Since this test is dominated by Writes, and only one activate and precharge, as in the case of $I_{DD4R}$, we subtract the activation-precharge energy calculated from the $I_{DD0}$ test from the total observed energy in Figure A.12, to obtain the $I_{DD4W}$ current measure.


File Vertical Timebase Trigger Display Cursors Measure Math Analysis Utilities Help
Undo
ACT
READs
Read to PRE
PRE
LeCroy

```
Measure   P1:mean(Math)  P2:mean(C1)  P3:mean(C2)  P4:- - -  P5:- - -  P6:- - -
value     88.70 mV       1.5098 V     1.4259 V
mean      88.7028 mV     1.509763 V   1.425913 V
min       88.70 mV       1.5098 V     1.4259 V
max       88.70 mV       1.5098 V     1.4259 V
sdev      ---            ---          ---
num       1              1            1
status    ✓              ✓            ✓

Math  eres((C1-...     Timebase    0.0 µs   Trigger   C4 DC
      7.6 mV/div                20.0 µs/div  Stop      0.0 mV
      42.0 ns/div      200 kS    1.0 GS/s    Edge    Positive
----  93.20 mV
..... 60.68 mV
Δy    -32.53 mV                     2/19/2014 3:43:48 PM
```

Figure A.11: Measuring $I_{DD4R}$

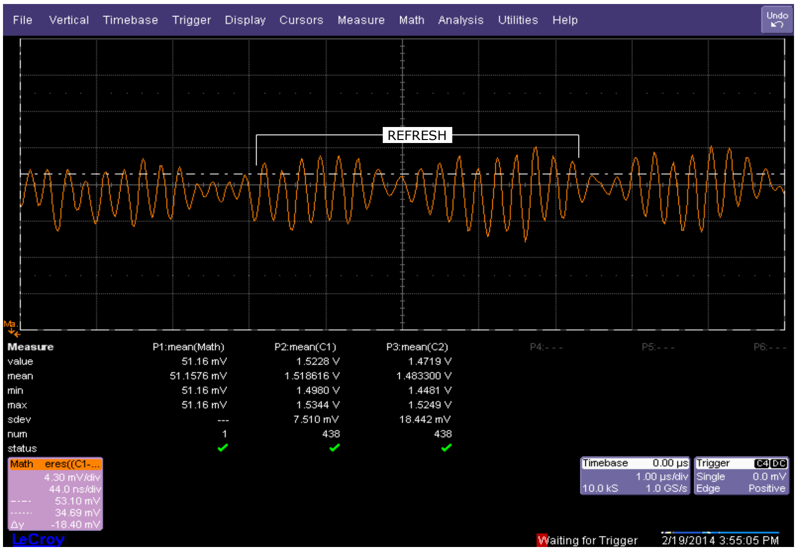

File Vertical Timebase Trigger Display Cursors Measure Math Analysis Utilities Help
Undo
WRITEs
ACT
PRE
LeCroy

```
Measure   P1:mean(Math)  P2:mean(C1)  P3:mean(C2)  P4:- - -  P5:- - -  P6:- - -
value     51.94 mV       1.4993 V     1.4493 V
mean      51.9434 mV     1.499311 V   1.449254 V
min       51.94 mV       1.4993 V     1.4493 V
max       51.94 mV       1.4993 V     1.4493 V
sdev      ---            ---          ---
num       1              1            1
status    ✓              ✓            ✓

Math  eres((C1-...     Timebase    0.00 µs   Trigger   C4 DC
      6.1 mV/div                1.00 µs/div  Single    0.0 mV
      68 ns/div        10.0 kS    1.0 GS/s   Edge    Positive
----  45.64 mV
..... 19.54 mV
Δy    -26.11 mV        Waiting for Trigger   2/19/2014 3:51:23 PM
```

Figure A.12: Measuring $I_{DD4W}$

[8] $\boldsymbol{I_{DD5}}$ current - Complete Current

The $I_{DD5}$ test loop measures current across several REF commands, with all banks retained in the precharged state. Each REF command is followed by another REF command after $nRFC$ clock cycles, the duration to allow a REF command to finish. A burst of 64 REF commands are sent. A few are shown in Figure A.13. Since this test only involves refreshes, which repeat without requiring any additional operations, the measurement observed is a complete current.



| Measure | P1:mean(Math) | P2:mean(C1) | P3:mean(C2) | P4:- - - | P5:- - - | P6:- - - |
|---|---|---|---|---|---|---|
| value | 51.16 mV | 1.5228 V | 1.4719 V | | | |
| mean | 51.1576 mV | 1.518616 V | 1.483300 V | | | |
| min | 51.16 mV | 1.4980 V | 1.4481 V | | | |
| max | 51.16 mV | 1.5344 V | 1.5249 V | | | |
| sdev | --- | 7.510 mV | 18.442 mV | | | |
| num | 1 | 438 | 438 | | | |
| status | ✓ | ✓ | ✓ | | | |

```
Math   eres((C1-...
          4.30 mV/div
          44.0 ns/div
-·-·-      53.10 mV
------     34.69 mV
Δy        -18.40 mV
```

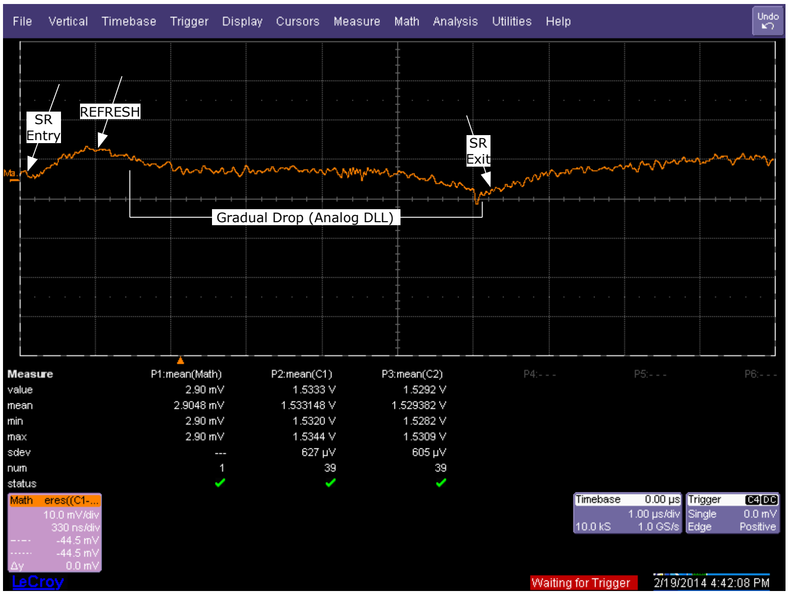
```
Timebase     0.00 µs   Trigger    C4 DC
           1.00 µs/div Single    0.0 mV
10.0 kS      1.0 GS/s  Edge    Positive
```

Waiting for Trigger 2/19/2014 3:55:05 PM

Figure A.13: Measuring $I_{DD5}$

[9] $\boldsymbol{I_{DD6}}$ current measurement - Partial Current - Required Precharging

The $I_{DD6}$ test loop measures current in the self-refresh low power state. This test starts with a self-refresh (SREN) command, when all the banks are in the precharged state and ends after a power-up is instantiated and completed. The time period between the SREN and power-up (SREX) command is 2000 clock cycles (sufficiently long compared to the power-up time). After this period, a power-up is issued and the test pattern ends after further $nXSDLL$ clock cycles. This test pattern repeats itself over time. Since this test requires power-up operations besides retaining the DRAM in the self-refresh state (as shown in Figure A.14), we use density plots to identify the voltage measure for the longest retained state (self-refresh state), depicted in Figure A.15. As can be observed from the plot in Figure A.14, there is a peak observed after entering the self-refresh mode, before the power consumption drops down. This can be attributed to the internal refresh that is instantiated when the self-refresh mode is entered. Another interesting aspect is the gradual drop in power consumption, when entering self-refresh instead of the immediate drop. This can be attributed to the fact the Micron DIMM employs an analog implementation of the DLL [119] instead of a digital implementation. The density plot in Figure A.15 further shows that the self-refresh state is lowest power consuming state in DRAMs.


File Vertical Timebase Trigger Display Cursors Measure Math Analysis Utilities Help
Undo
SR Entry
REFRESH
SR Exit
Gradual Drop (Analog DLL)
Ma
LeCroy

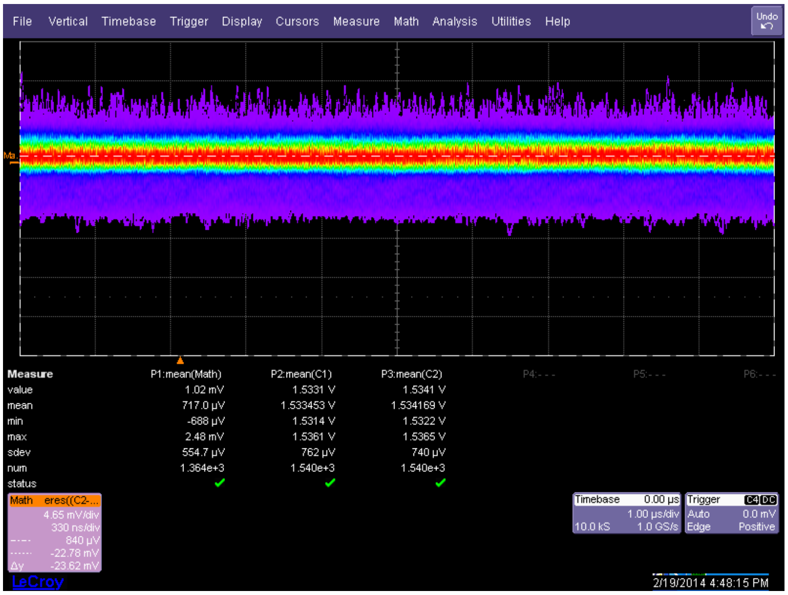
| Measure | P1:mean(Math) | P2:mean(C1) | P3:mean(C2) | P4:- - - | P5:- - - | P6:- - - |
|---|---|---|---|---|---|---|
| value | 2.90 mV | 1.5333 V | 1.5292 V | | | |
| mean | 2.9048 mV | 1.533148 V | 1.529382 V | | | |
| min | 2.90 mV | 1.5320 V | 1.5282 V | | | |
| max | 2.90 mV | 1.5344 V | 1.5309 V | | | |
| sdev | --- | 627 µV | 605 µV | | | |
| num | 1 | 39 | 39 | | | |
| status | ✓ | ✓ | ✓ | | | |

```
Math  eres((C1-...
      10.0 mV/div
      330 ns/div
---   -44.5 mV
....  -44.5 mV
Δy    0.0 mV
```

```
Timebase     0.00 µs   Trigger    C4 DC
         1.00 µs/div   Single    0.0 mV
10.0 kS    1.0 GS/s    Edge    Positive
```

Waiting for Trigger 2/19/2014 4:42:08 PM

Figure A.14: Measuring $I_{DD6}$ with Precharging


File Vertical Timebase Trigger Display Cursors Measure Math Analysis Utilities Help
Undo
Ma
LeCroy

| Measure | P1:mean(Math) | P2:mean(C1) | P3:mean(C2) | P4:- - - | P5:- - - | P6:- - - |
|---|---|---|---|---|---|---|
| value | 1.02 mV | 1.5331 V | 1.5341 V | | | |
| mean | 717.0 µV | 1.533453 V | 1.534169 V | | | |
| min | -688 µV | 1.5314 V | 1.5322 V | | | |
| max | 2.48 mV | 1.5361 V | 1.5365 V | | | |
| sdev | 554.7 µV | 762 µV | 740 µV | | | |
| num | 1.364e+3 | 1.540e+3 | 1.540e+3 | | | |
| status | ✓ | ✓ | ✓ | | | |

```
Math  eres((C2-...
      4.65 mV/div
      330 ns/div
---   840 µV
....  -22.78 mV
Δy    -23.62 mV
```

```
Timebase     0.00 µs   Trigger    C4 DC
         1.00 µs/div   Auto      0.0 mV
10.0 kS    1.0 GS/s    Edge    Positive
```

2/19/2014 4:48:15 PM

Figure A.15: Measuring $I_{DD6}$ - Density Plot

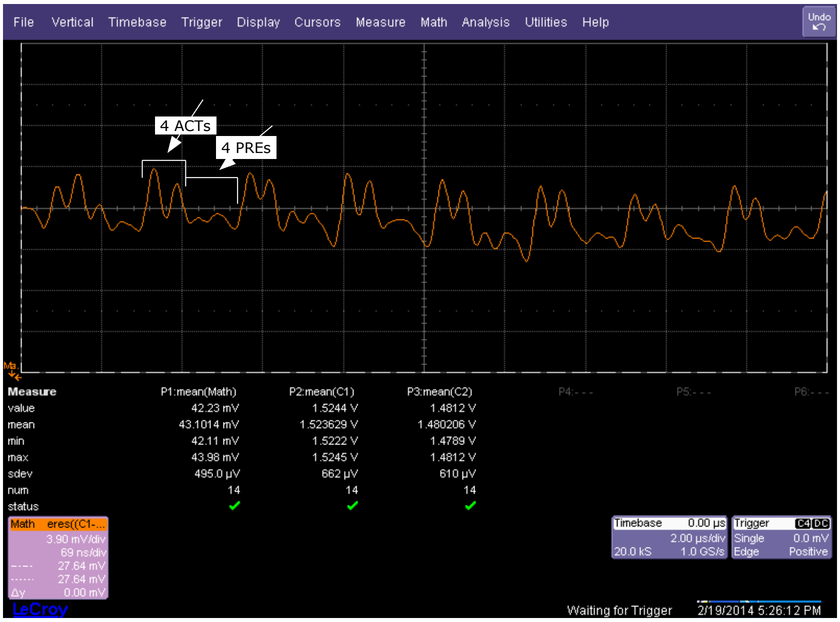
## A.2 DRAMPower Evaluation

The DRAMPower validation in Chapter 4 included two sets of experiments. The first set was defined with the aim to analyze the accuracy of the modeling of the active DRAM operations while scaling the timings and number of reads/writes and degree of bank-interleaving. The second set of experiments analyzed the same for state transitions from active to power-down and self-refresh modes, the idle standby mode and refreshes. In this section, we present more tests [#7 to #17] and their corresponding measurement plots highlighting the same modeling issues with more experiments.

### A.2.1 Activation - Precharge Scaling

[7] **4 Banks - ACT-PRE**: We performed the four-bank activate-precharge operations, as represented by the current measures in Figure A.16.

In this case, the average power consumption is expected to be significantly lower than four times of that of a single bank ACT-PRE (as also observed in the experiment). This is due to successive *nRRD* delays, which are imposed as a restriction (minimum timing constraint) between the four ACT commands to different banks. On the other hand, Micron's model estimates this power measure to be exactly four times of that of a single-bank ACT-PRE, ignoring the effect of all the *nRRD* delays.



| Measure | P1:mean(Math) | P2:mean(C1) | P3:mean(C2) | P4:- - - | P5:- - - | P6:- - - |
|---|---|---|---|---|---|---|
| value | 42.23 mV | 1.5244 V | 1.4812 V | | | |
| mean | 43.1014 mV | 1.523629 V | 1.480206 V | | | |
| min | 42.11 mV | 1.5222 V | 1.4789 V | | | |
| max | 43.98 mV | 1.5245 V | 1.4812 V | | | |
| sdev | 495.0 µV | 662 µV | 610 µV | | | |
| num | 14 | 14 | 14 | | | |
| status | ✓ | ✓ | ✓ | | | |

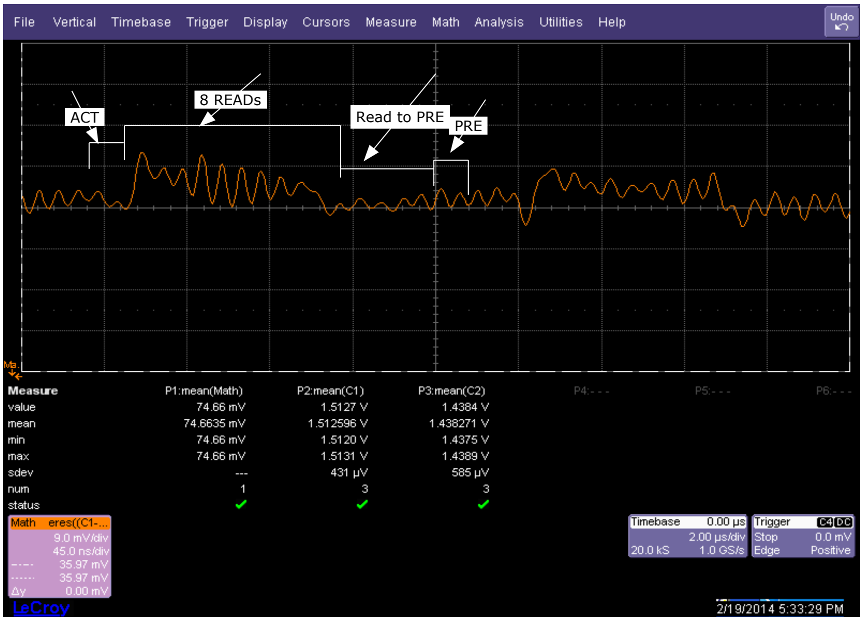
```
Math    eres((C1-...
        3.90 mV/div
        69 ns/div
        27.64 mV
        27.64 mV
Δy      0.00 mV

Timebase      0.00 µs    Trigger    C4 DC
           2.00 µs/div   Single    0.0 mV
20.0 kS     1.0 GS/s     Edge    Positive

Waiting for Trigger    2/19/2014 5:26:12 PM
```

Figure A.16: 4 Banks - ACT-PRE

## A.2.2 Read / Write Scaling

[8] **8 Reads - 1 Bank**: We performed the eight successive Read operations to a bank including the corresponding ACT-PRE operations, as represented by the current measures in Figure A.17.

In this case, the average power consumption is expected to be significantly lower than the sum of eight Read operations and that of a single bank ACT-PRE. This is due to the total latency of the transaction, which can be calculated as the sum of $nRCD$ delay after the ACT command leading up to the first Read, $nRTP$ delay after the last Read before a PRE is issued, the $nRP$ delay for the PRE to complete and successive $nCCD$ delays between the eight Read commands, which is much longer than $nRC$ cycles.



| Measure | P1:mean(Math) | P2:mean(C1) | P3:mean(C2) | P4:- - - | P5:- - - | P6:- - - |
|---|---|---|---|---|---|---|
| value | 74.66 mV | 1.5127 V | 1.4384 V | | | |
| mean | 74.6635 mV | 1.512596 V | 1.438271 V | | | |
| min | 74.66 mV | 1.5120 V | 1.4375 V | | | |
| max | 74.66 mV | 1.5131 V | 1.4389 V | | | |
| sdev | --- | 431 µV | 585 µV | | | |
| num | 1 | 3 | 3 | | | |
| status | ✓ | ✓ | ✓ | | | |

```
Math   eres((C1-...
         9.0 mV/div
        45.0 ns/div
-·-·-   35.97 mV
······  35.97 mV
Δy       0.00 mV
```

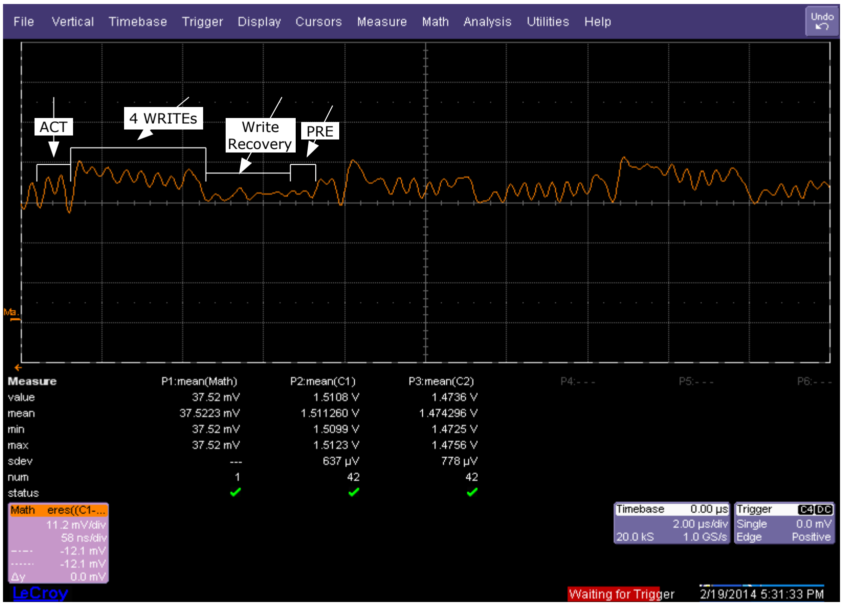
```
Timebase     0.00 µs   Trigger      C4 DC
           2.00 µs/div Stop        0.0 mV
20.0 kS      1.0 GS/s  Edge      Positive
```

2/19/2014 5:33:29 PM

Figure A.17: 8 Reads - 1 Bank

[9] **4 Writes - 1 Bank**: We performed the four successive Write operations to a bank including the corresponding ACT-PRE operations, as represented by the current measures in Figure A.18.

In this case, the average power consumption is expected to be much lower than the sum of four Write operations and that of a single bank ACT-PRE. This is also due to the total latency of the transaction, which can be calculated as the sum of $nRCD$ delay after the ACT command leading up to the first Write, $nWL$ + $nWR$ + $n(BL/2)$ delay after the last Write before a PRE is issued, the $nRP$ delay for the PRE to complete and successive $nCCD$ delays between the Write commands.



```
Measure   P1:mean(Math)  P2:mean(C1)  P3:mean(C2)   P4:---   P5:---   P6:---
value     37.52 mV       1.5108 V     1.4736 V
mean      37.5223 mV     1.511260 V   1.474296 V
min       37.52 mV       1.5099 V     1.4725 V
max       37.52 mV       1.5123 V     1.4756 V
sdev      ---            637 µV       778 µV
num       1              42           42
status    ✓              ✓            ✓

Math  eres((C1-...        Timebase    0.00 µs   Trigger   C4 DC
        11.2 mV/div                2.00 µs/div  Single    0.0 mV
         58 ns/div        20.0 kS     1.0 GS/s  Edge      Positive
----    -12.1 mV
------  -12.1 mV
Δy        0.0 mV

Waiting for Trigger    2/19/2014 5:31:33 PM
```

Figure A.18: 4 Writes - 1 Bank

[10] **8 Writes - 1 Bank**: We also performed the eight successive Write operations to a bank, as represented by the current measures in Figure A.19.

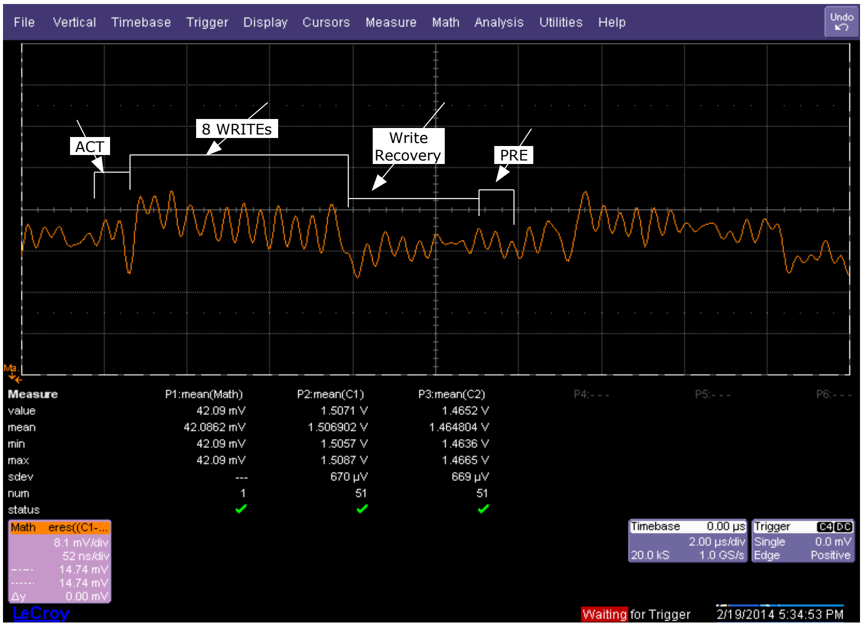
In this case as well, the average power consumption is expected to be much lower than the sum of eight Write operations and that of a single bank ACT-PRE, due to the total latency of the transaction, which is a lot longer than $nRC$ cycles.

### A.2.3 Bank Interleaving - Reads / Writes

[11] **1 Read - 2 Banks**: We performed the two-bank activate-precharge operations combined with a Read operation on each bank, as represented by the current measures in Figure A.20.

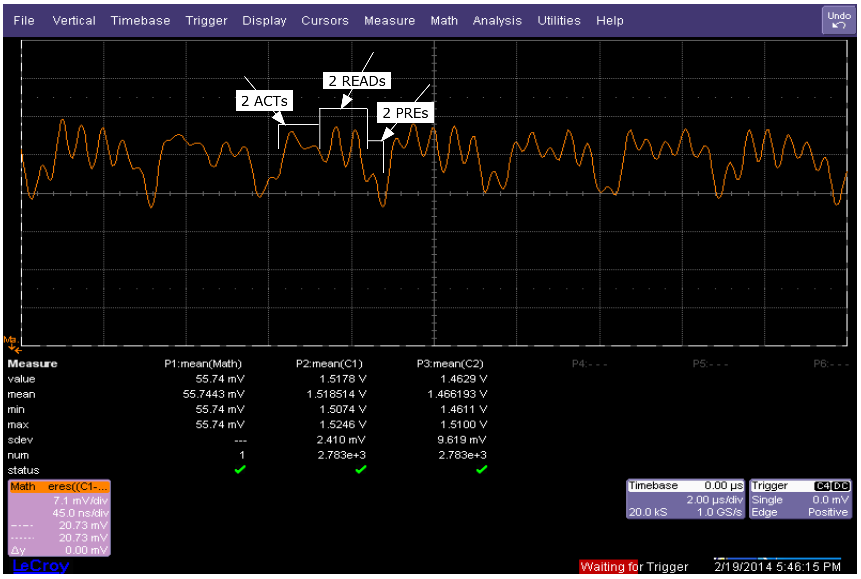
In this case, the average power consumption is expected to be lower than the sum of two Read operations and that of two single bank ACT-PRE commands. This is due to the total latency as well, which can be calculated as the sum of $nRCD$ delay after the first ACT command leading up to the first Read, $nRTP$ delay after the last Read (to the second bank) before a PRE is issued, the $nRP$ delay for the last PRE to complete, the $nRRD$ delay between the two ACT commands and the successive $nCCD$ delays between the two Read commands.

[12] **1 Read - 8 Banks**: We extended the last experiment and performed eight-bank activate-precharge operations combined with a Read operation on each bank to further highlight the importance of modeling scaling, as represented by the current measures in Figure A.21.



```
Measure   P1:mean(Math)   P2:mean(C1)   P3:mean(C2)   P4:---   P5:---   P6:---
value     42.09 mV        1.5071 V      1.4652 V
mean      42.0862 mV      1.506902 V    1.464804 V
min       42.09 mV        1.5057 V      1.4636 V
max       42.09 mV        1.5087 V      1.4665 V
sdev      ---             670 µV        669 µV
num       1               51            51
status    ✓               ✓             ✓

Math  eres((C1-...
       8.1 mV/div
       52 ns/div
       14.74 mV
       14.74 mV
Δy     0.00 mV

Timebase   0.00 µs    Trigger    C4 DC
       2.00 µs/div    Single     0.0 mV
20.0 kS   1.0 GS/s    Edge       Positive

Waiting for Trigger   2/19/2014 5:34:53 PM
```

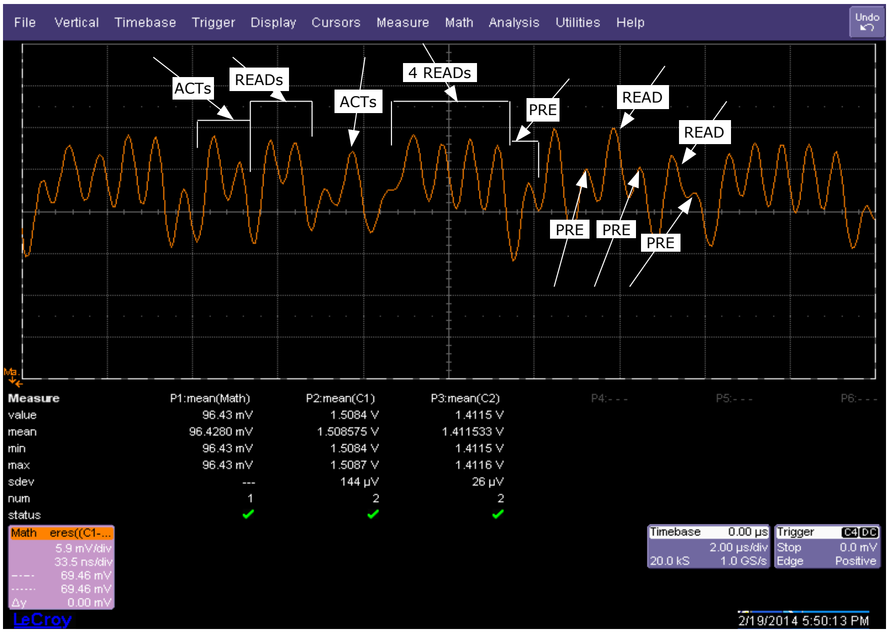
Figure A.19: 8 Writes - 1 Bank



```
Measure   P1:mean(Math)   P2:mean(C1)   P3:mean(C2)   P4:---   P5:---   P6:---
value     55.74 mV        1.5178 V      1.4629 V
mean      55.7443 mV      1.518514 V    1.466193 V
min       55.74 mV        1.5074 V      1.4611 V
max       55.74 mV        1.5246 V      1.5100 V
sdev      ---             2.410 mV      9.619 mV
num       1               2.783e+3      2.783e+3
status    ✓               ✓             ✓

Math  eres((C1-...
       7.1 mV/div
       45.0 ns/div
       20.73 mV
       20.73 mV
Δy     0.00 mV

Timebase   0.00 µs    Trigger    C4 DC
       2.00 µs/div    Single     0.0 mV
20.0 kS   1.0 GS/s    Edge       Positive

Waiting for Trigger   2/19/2014 5:46:15 PM
```

Figure A.20: 1 Read - 2 Banks

In this case, the average power consumption is expected to be closer to that of four-bank activate-precharge operation combined with a Read operation on each bank (Figure 4.10) due the $nFAW$ delay imposed after activating four banks and significantly lower than the sum of eight Read operations and that of eight single bank ACT-PRE commands.



| Measure | P1:mean(Math) | P2:mean(C1) | P3:mean(C2) | P4:- - - | P5:- - - | P6:- - - |
|---|---|---|---|---|---|---|
| value | 96.43 mV | 1.5084 V | 1.4115 V | | | |
| mean | 96.4280 mV | 1.508575 V | 1.411533 V | | | |
| min | 96.43 mV | 1.5084 V | 1.4115 V | | | |
| max | 96.43 mV | 1.5087 V | 1.4116 V | | | |
| sdev | --- | 144 µV | 26 µV | | | |
| num | 1 | 2 | 2 | | | |
| status | ✓ | ✓ | ✓ | | | |

Math eres((C1-...
5.9 mV/div
33.5 ns/div
69.46 mV
69.46 mV
Δy 0.00 mV

Timebase 0.00 µs
2.00 µs/div
20.0 kS 1.0 GS/s

Trigger C4 DC
Stop 0.0 mV
Edge Positive

2/19/2014 5:50:13 PM

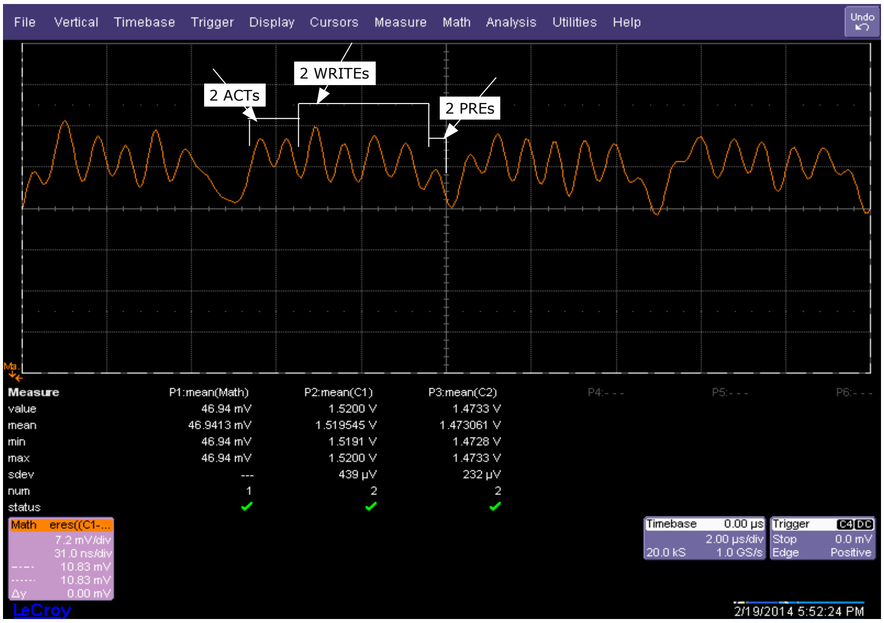
Figure A.21: 1 Read - 8 Banks

[13] **1 Write - 2 Banks**: We performed the two-bank activate-precharge operations combined with a Write operation on each bank (similar to reads), as represented by the current measures in Figure A.22.

In this case as well, the average power consumption is expected to be lower than the sum of two Write operations and that of two single bank ACT-PRE commands.

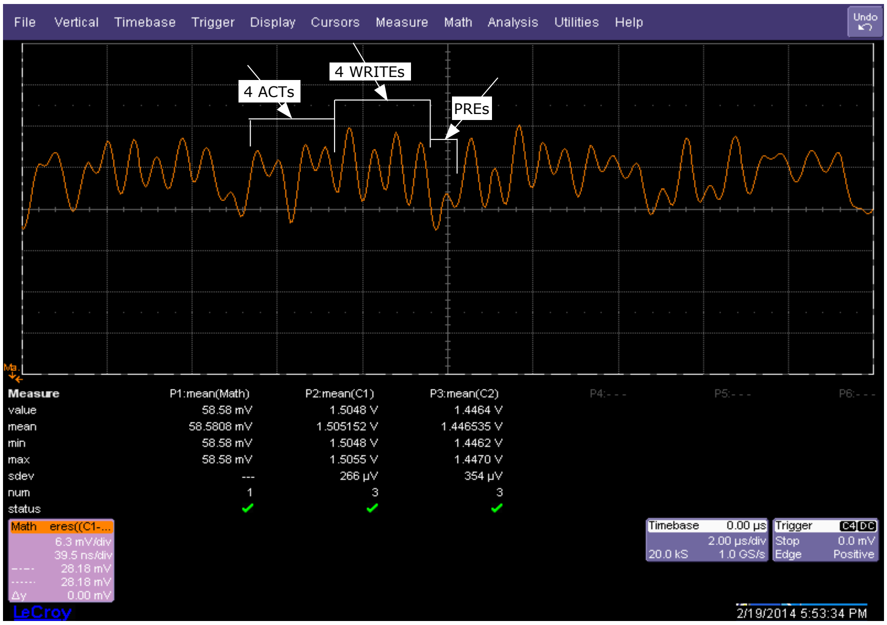
[14] **1 Write - 4 Banks**: We also performed the four-bank activate-precharge operations combined with a Write operation on each bank and observed similar scaling effects as in the case of reads, as represented by the current measures in Figure A.23.

[15] **1 Write - 8 Banks**: We further performed the eight-bank activate-precharge operations combined with a Write operation on each bank and observed current measures closer to the 4 bank active case, as observed for reads. This is depicted in Figure A.24.


File Vertical Timebase Trigger Display Cursors Measure Math Analysis Utilities Help
Undo
2 ACTs
2 WRITEs
2 PREs
LeCroy

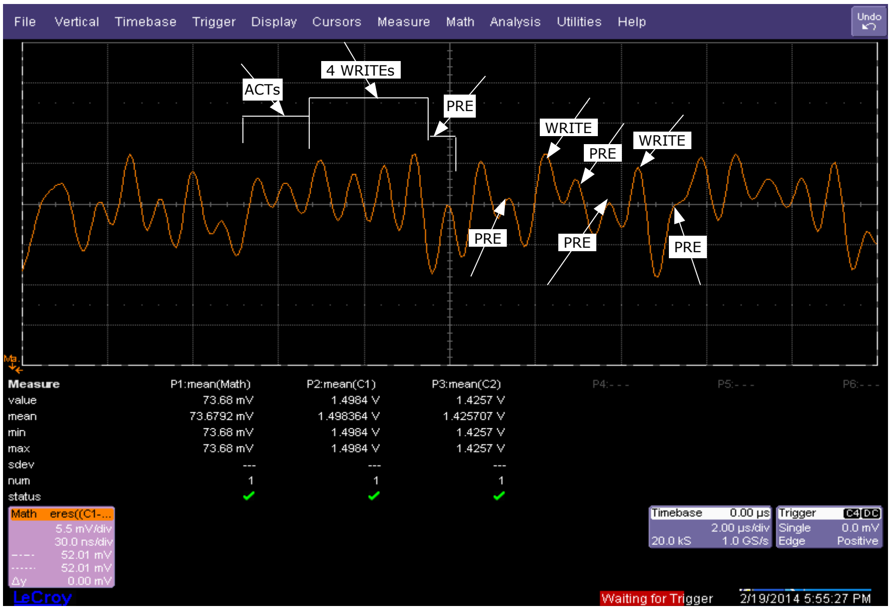
```
Measure   P1:mean(Math)   P2:mean(C1)   P3:mean(C2)   P4:- - -   P5:- - -   P6:- - -
value     46.94 mV        1.5200 V      1.4733 V
mean      46.9413 mV      1.519545 V    1.473061 V
min       46.94 mV        1.5191 V      1.4728 V
max       46.94 mV        1.5200 V      1.4733 V
sdev      ---             439 µV        232 µV
num       1               2             2
status    ✓               ✓             ✓

Math  eres((C1-...        Timebase    0.00 µs   Trigger   C4 DC
      7.2 mV/div                  2.00 µs/div   Stop      0.0 mV
      31.0 ns/div         20.0 kS   1.0 GS/s    Edge      Positive
----  10.83 mV
..... 10.83 mV
Δy    0.00 mV                                  2/19/2014 5:52:24 PM
```

Figure A.22: 1 Write - 2 Banks


File Vertical Timebase Trigger Display Cursors Measure Math Analysis Utilities Help
Undo
4 ACTs
4 WRITEs
PREs
LeCroy

```
Measure   P1:mean(Math)   P2:mean(C1)   P3:mean(C2)   P4:- - -   P5:- - -   P6:- - -
value     58.58 mV        1.5048 V      1.4464 V
mean      58.5808 mV      1.505152 V    1.446535 V
min       58.58 mV        1.5048 V      1.4462 V
max       58.58 mV        1.5055 V      1.4470 V
sdev      ---             266 µV        354 µV
num       1               3             3
status    ✓               ✓             ✓

Math  eres((C1-...        Timebase    0.00 µs   Trigger   C4 DC
      6.3 mV/div                  2.00 µs/div   Stop      0.0 mV
      39.5 ns/div         20.0 kS   1.0 GS/s    Edge      Positive
----  28.18 mV
..... 28.18 mV
Δy    0.00 mV                                  2/19/2014 5:53:34 PM
```

Figure A.23: 1 Write - 4 Banks



| Measure | P1:mean(Math) | P2:mean(C1) | P3:mean(C2) | P4:- - - | P5:- - - | P6:- - - |
|---|---|---|---|---|---|---|
| value | 73.68 mV | 1.4984 V | 1.4257 V | | | |
| mean | 73.6792 mV | 1.498364 V | 1.425707 V | | | |
| min | 73.68 mV | 1.4984 V | 1.4257 V | | | |
| max | 73.68 mV | 1.4984 V | 1.4257 V | | | |
| sdev | --- | --- | --- | | | |
| num | 1 | 1 | 1 | | | |
| status | ✓ | ✓ | ✓ | | | |

```
Math  eres((C1-...
        5.5 mV/div
       30.0 ns/div
----     52.01 mV
------   52.01 mV
Δy        0.00 mV

Timebase      0.00 µs   Trigger     C4 DC
          2.00 µs/div   Single     0.0 mV
20.0 kS     1.0 GS/s    Edge     Positive
```

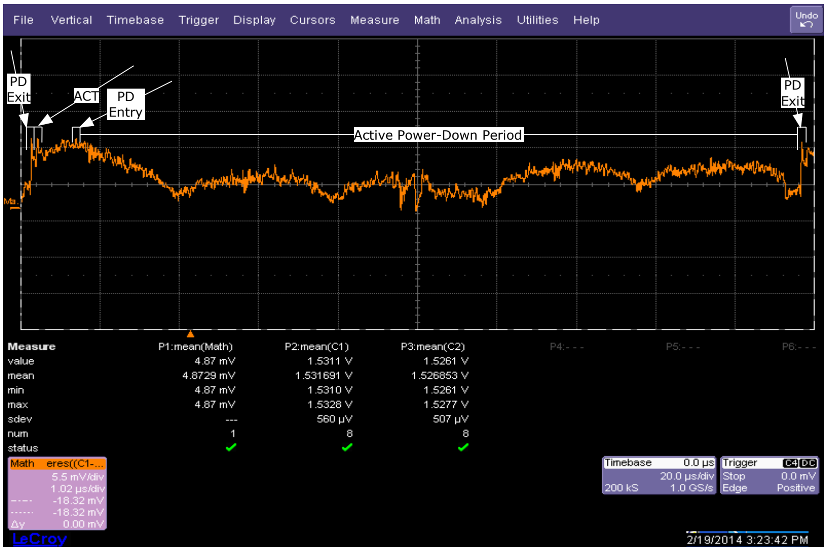
Figure A.24: 1 Write - 8 Banks

## A.2.4 Impact of State Transitions

[16] **ACT to Active Power-Down**: We performed an experiment to transition from a single-bank activation operation into the active powered-down state to highlight the importance of modeling this state transition, as represented by the current measures in Figure A.25.

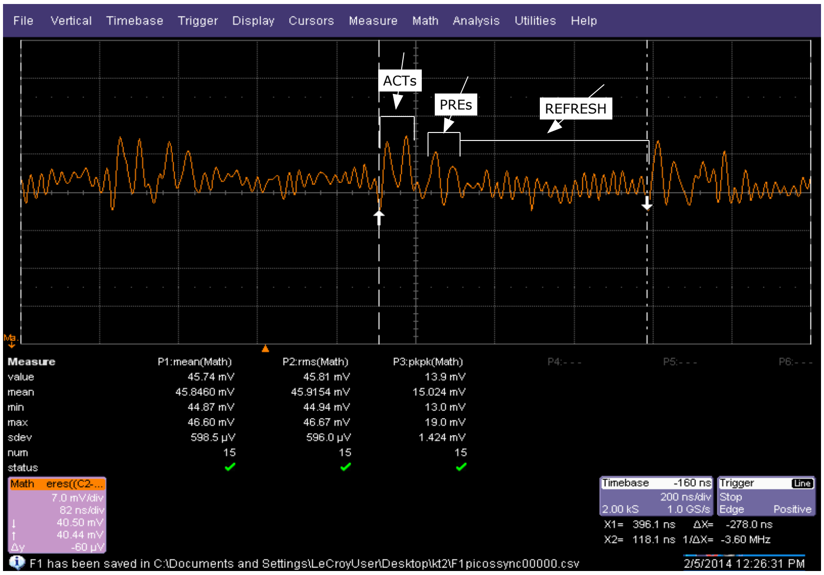
In this case, it is important to note that the power consumption reduces gradually as we enter or exit the active power-down state. This transition must be captured in the power model depending on the duration for which the clock is gated and the time required to power-up the memory back into an active idle (standby) state. Micron's model does not model these transitions.

[17] **ACT - PRE to Refresh**: We performed an experiment to transition from multi-bank activation-precharge operations into the refresh mode to highlight the importance of modeling this state transition, as represented by the current measures in Figure A.26. In this case, it is important to note that the power consumption due to the enforced precharging (multi-bank), which may be required before starting a refresh operation, is significant and must be accounted as a part of the refresh power consumption. Furthermore, the power-consumption during the last $nRP$ cycles during a refresh operation, is slightly lower than the rest of the clock cycles. This is because internally the DRAM transitions into the precharged state during the last $nRP$ cycles of the refresh operation, hence the background current is lower. Micron's model does not model these transitions into and out of a refresh operation.


File Vertical Timebase Trigger Display Cursors Measure Math Analysis Utilities Help
Undo
PD Exit
ACT
PD Entry
PD Exit
Active Power-Down Period
LeCroy

```
Measure   P1:mean(Math)  P2:mean(C1)  P3:mean(C2)  P4:---  P5:---  P6:---
value     4.87 mV        1.5311 V     1.5261 V
mean      4.8729 mV      1.531691 V   1.526853 V
min       4.87 mV        1.5310 V     1.5261 V
max       4.87 mV        1.5328 V     1.5277 V
sdev      ---            560 µV       507 µV
num       1              8            8
status    ✓              ✓            ✓

Math  eres((C1-...      Timebase    0.0 µs   Trigger   C4 DC
      5.5 mV/div                  20.0 µs/div  Stop     0.0 mV
      1.02 µs/div       200 kS     1.0 GS/s   Edge     Positive
      -18.32 mV
      -18.32 mV
Δy    0.00 mV                                  2/19/2014 3:23:42 PM
```

Figure A.25: ACT to Active Power-Down


File Vertical Timebase Trigger Display Cursors Measure Math Analysis Utilities Help
ACTs
PREs
REFRESH

```
Measure   P1:mean(Math)  P2:rms(Math)  P3:pkpk(Math)  P4:---  P5:---  P6:---
value     45.74 mV       45.81 mV      13.9 mV
mean      45.8460 mV     45.9154 mV    15.024 mV
min       44.87 mV       44.94 mV      13.0 mV
max       46.60 mV       46.67 mV      19.0 mV
sdev      598.5 µV       596.0 µV      1.424 mV
num       15             15            15
status    ✓              ✓             ✓

Math  eres((C2-...      Timebase   -160 ns   Trigger   Line
      7.0 mV/div                 200 ns/div  Stop
      82 ns/div         2.00 kS    1.0 GS/s  Edge     Positive
↓     40.50 mV          X1=  396.1 ns   ΔX=    -278.0 ns
↑     40.44 mV          X2=  118.1 ns   1/ΔX=  -3.60 MHz
Δy    -60 µV
F1 has been saved in C:\Documents and Settings\LeCroyUser\Desktop\kt2\F1picossync00000.csv   2/5/2014 12:26:31 PM
```

Figure A.26: ACT - PRE to Refresh

These experiments provide sufficient data to validate the DRAMPower model and verify its power estimates against real hardware measures. These results are employed in Chapter 4 in Table 4.6 and Figure 4.13 to verify DRAMPower's estimates.

# Chapter B
# List of Publications

## Articles Included in this Thesis

- K. Chandrasekar, B. Åkesson, and K. Goossens, *Improved Power Modeling of DDR SDRAMs*, **In Proc. DSD 2011**.
- K. Chandrasekar, B. Åkesson, and K. Goossens, *Run-time power-down strategies for real-time SDRAM memory controllers*, **In Proc. DAC 2012**.
  **HiPEAC Paper Award**.
- K. Chandrasekar, C. Weis, B. Akesson, N. Wehn, and K. Goossens, *System and Circuit Level Power Modeling of Energy-Efficient 3D-Stacked Wide I/O DRAMs*, **In Proc. DATE 2013**.
- K. Chandrasekar, C. Weis, B. Akesson, N. Wehn, and K. Goossens, *Towards Variation-aware System-level Power Estimation of DRAMs: An Empirical Approach*, **In Proc. DAC 2013**.
  **HiPEAC Paper Award**.
- 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 Proc. DATE 2014**.

## Tools

- K. Chandrasekar, C. Weis, Y. Li, B. Akesson, N. Wehn, and K. Goossens, *DRAMPower: Open-source DRAM power & energy estimation tool.*
  **`www.es.ele.tue.nl/drampower`**.
  **Downloaded by more than 125 Universities and Companies and integrated with GEM5 Simulator**.

## Other Articles

- K. Goossens, A. Azevedo, K. Chandrasekar, M. D. Gomony, S. Goossens, M. Koedam, Y. Li, D. Mirzoyan, A. Molnos, A. B. Nejad, A. Nelson, and S. Sinha, *Virtual execution platforms for mixed-time-criticality systems: The CompSOC architecture and design flow*, In ACM SIGBED Review, vol. 10, issue. 3, 2013.
- M. Jung, C. Weis, K. Chandrasekar, and N. Wehn, *TLM modelling of 3D stacked wide I/O DRAM subsystems: A virtual platform for memory controller design space exploration*, In Proc. RAPIDO 2013.
- G. Thomas, K. Chandrasekar, B. Åkesson, B. Juurlink, K. Goossens, *A Predictor-Based Power-Saving Policy for DRAM Memories*, In Proc. DSD 2012.

## Posters

- K. Chandrasekar, *High-Level Power Estimation of DRAMs*, PhD Forum - DATE 2014.
- K. Chandrasekar, B. Åkesson, and K. Goossens, *Predictable Power-Down Policies for SDRAMs*, ICT.OPEN 2011.
- K. Chandrasekar, B. Åkesson, and K. Goossens, *Modeling and Optimizing Power for a Real-Time SDRAM Controller*, PROGRESS 2010.

# Chapter C
# About the Author

Karthik Chandrasekar was born on 3$^{rd}$ July 1984 in Chennai, India. He received his Bachelor of Engineering degree in Computer Science and Engineering in 2006 with Distinction from Anna University in Chennai, India. Between August 2004 and August 2007, he worked as a research trainee at Waran Research Foundation (WARFT), Chennai, India, as a member of their High Performance Computing group. In September 2007, he moved to Delft to pursue his Master's degree in Computer Engineering from TU Delft. For his Masters' thesis, he worked on performance validation of Network On Chip architectures at EPFL, Switzerland. In December 2009, having completed his Master's degree, he started his PhD also in the Computer Engineering group at TU Delft under the guidance of Prof. Kees Goossens. During his PhD he visited the Microelectronic Systems Design research group at TU Kaiserslautern, Germany in 2012, for a three month collaboration project funded by HiPEAC. During his PhD, he also received the HiPEAC paper award twice for his publications at DAC 2012 and DAC 2013. His research interests include Real-Time Embedded Systems, System On Chip Architectures, Power and Performance Modeling and Optimization, DRAM Memories and Memory Controllers, and Networks-On-Chip.