UNDERSTANDING AND IMPROVING THE ENERGY EFFICIENCY OF DRAM

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF ELECTRICAL ENGINEERING
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Heonjae Ha
October 2018

© 2018 by Heonjae Ha. All Rights Reserved.
Re-distributed by Stanford University under license with the author.

This work is licensed under a Creative Commons Attribution-Noncommercial 3.0 United States License.
http://creativecommons.org/licenses/by-nc/3.0/us/

This dissertation is online at: http://purl.stanford.edu/yp843xn4828

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Mark Horowitz, Primary Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Christos Kozyrakis

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

S Wong

Approved for the Stanford University Committee on Graduate Studies.

Patricia J. Gumport, Vice Provost for Graduate Education

This signature page was generated electronically upon submission of this dissertation in electronic format. An original signed hard copy of the signature page is on file in University Archives.

Abstract

Human created data has been growing exponentially for decades and there are many reports stating 90% of all data were created in the past couple of years [1]. Moreover, this exceptionally high data growth rate is predicted to continue in the future. Increased data volume then drives storage capacity growth, and also places high demands on high bandwidth and large capacity DRAM [2]. However, growing capacity and memory performing also grows memory power making DRAM power consumption one of the most important component to optimize in power limited modern computer systems.

This power issue led to the development of specialized DRAMs, such as Low Power Double Data Rate (LPDDR) and High Bandwidth Memory (HBM), that are much more energy efficient than conventional DDR DRAM. While minimizing memory power is a popular research topic, the DRAM design space is highly constrained by the heavy cost-per-bit optimization done in DRAM. Surprisingly, these constraints are difficult to find as those information are known only within the small community of DRAM circuit designers.

To address this issue, we worked with former and current DRAM designers to build an open-sourced DRAM modeling framework named *DramDSE*, short for DRAM Design Space Exploration. The modeling framework incorporates crucial design constraints posed by modern DRAM and provides the area, detailed power breakdown, and current values that are needed to explore the design space of DRAM efficiently. We used DramDSE to create the first public domain models of two state-of-the-art DRAMs, LPDDR4 and HBM, for the first time in public domain and validated the correctness of our model using measured data collected from mass-produced LPDDR4

and HBM DRAMs.

With a better understanding of DRAM gained from DramDSE, we propose three DRAM energy reduction schemes. The first two are built on top of a unique cell array design that can access only half of the page. *Half Page DRAM* reduces the row energy due to the row buffer overfetch problem in multi-core systems, which yields 38% row energy savings without any bandwidth loss. *Charge Recycling Refresh* reduces up to 32% of refresh energy by recycling charges from fully refreshed half page rows to the other half page rows that are to be refreshed. Each of them can be implemented in a modern DRAM with less than 1.5% area overhead and when both schemes are used together, the total power consumption is reduced by 15% on average across various workload. This is equivalent to the power savings achieved by scaling the DRAM technology node from 20 nm to a 10 nm class [3]. Finally, we propose *Smart Refresh* to further reduce refresh energy with negligible area overheads by utilizing the retention time distribution of the cells. Reducing refresh energy becomes more important as DRAM technology scales and memory density increases. Our two refresh energy reduction schemes also work in self-refresh mode, where refresh energy consumption is even more significant.

Acknowledgements

There were so many delightful life-changing events that happened during my PhD program. I tied the knot with my other half and we were gifted with two energetic sons; all along the journey to earn the PhD. Looking back on my life as a PhD student, I was fortunate to have such amazing people that supported and encouraged me to stay on track. Without them, I would not have been able to come this far, and I would like to take this opportunity to send my utmost gratitude to them.

First and foremost, I would like to thank my advisor, Professor Mark Horowitz. I don't think Mark remembers, but he handed over the Master's degree certificate to me back in 2009. After three years from that graduation ceremony, we talked about potentially starting the PhD program and I cannot believe that I am close to the end of it. He was an amazing advisor who made me focus more on the big picture and not get distracted by details, which was not easy after four years of my stay in the industry. I just cannot imagine how I would have gone through the first few years that I struggled without his advice and patience, which I am forever grateful of. I thank Professor Simon Wong and Professor Christos Kozyrakis, for their insightful comments on the work presented in this thesis as well as those that were left behind. I am also grateful to Professor Phil Levis, who served as my oral defense committee and Professor David Miller, who happily accepted to be the chair of my oral defense despite him being on sabbatical. Special thank you to Professor Nick McKeown and Dr. John W. Lockwood for their support when I was applying for the PhD program. Dr. Stephen Richardson and Mary Jane Swenson helped me so much throughout my PhD and I cannot express enough gratitude to them. Many thanks to my colleagues in the VLSI research group for broadening my knowledge with diverse research topics.

This thesis would not have been possible without the fellowship from Kwanjeong Educational Foundation. I was fortunate to receive the fellowship that lasted initially for four years to begin with, which Kwanjeong generously extended it for another year to support my entire PhD years. The work presented in this thesis were supported in part by NSF and C-FAR, one of the six SRC STARnet Centers, sponsored by MARCO and DARPA. Samsung and SK Hynix also supported my work by providing generous gift funding as well as valuable information that is not easily obtainable elsewhere. I would like to take this opportunity to extend my gratitude to former and current DRAM engineers at SK Hynix, in no particular order: Yongkee Kwon, Sangkug Lym, Taeksang Song, Hyun Gon Kim, Jun Hyun Chun, Joo Hwan Cho, and Jong Hoon Oh. They were always open to chat with me and updated me with the newest technology, which helped me throughout my PhD.

I am so grateful for the continuous support and encouragement from my parents, Kiryong Ha and Kweeysoon Kim, and my parents-in-law, Jueng Kwen Park and Seung Hee Ko. Without them, I would not have been able to send Minjoon to preschool which was a life saver for us and a fun time for him. I cannot thank Seung Hee Ko and Kweeysoon Kim enough for staying with us for months to help our family when both Minjoon and Yujoon were born. My oral defense and this thesis would not have been possible without the recent visit from my mother-in-law, Seung Hee Ko.

My greatest appreciation goes to my wife, Yeonhee Park, whom I dedicate this thesis to. My PhD started right after our wedding ceremony and in part, the entire PhD journey was our marriage life. She always said that she was raising three sons, including me, and I am forever grateful for her love and support that fueled me to this date. Lastly, thank you Minjoon and Yujoon for growing up healthy and happy.

Contents

Abstract **iv**

Acknowledgements **vi**

1 Introduction **1**

1.1 Constraints Posed by Modern DRAM . . . 1
1.2 Power Consumption of DRAM . . . 2
1.3 Thesis Contributions . . . 2
1.4 Thesis Organization . . . 3

2 DRAM Basics **5**

2.1 DRAM Cell . . . 5
2.2 Basic Operations . . . 8
2.2.1 Data Access . . . 8
2.2.2 Refresh . . . 11
2.3 Data Transfer . . . 12
2.3.1 Double Data Rate and Core Frequency . . . 12
2.3.2 Bandwidth and DRAM Configuration . . . 14
2.4 Power Consumption . . . 15
2.4.1 IDD Specification . . . 15
2.4.2 Dynamic Power Consumption . . . 16

3 DRAM Design Constraints **18**

3.1 Structural Organization . . . 18

3.2 Core Design 20
3.2.1 Hierarchical Wires 21
3.2.2 Shared Everything 24
3.3 Repair Scheme and Redundant Cells 26

4 DRAM Modeling Framework **30**
4.1 Background 31
4.2 Prior DRAM Modeling Work 32
4.3 DramDSE: DRAM Design Space Exploration 33
4.4 Configuration Parameters 36
4.4.1 Architecture Parameters 37
4.4.2 Technology Parameters 40
4.5 Validation of DramDSE 43
4.5.1 Low Power DDR4 (LPDDR4) 43
4.5.2 High Bandwidth Memory (HBM) 47
4.6 Power Breakdown Analysis 48
4.7 System Simulation Methodology 52
4.8 Use Case Examples 53
4.8.1 Density Explorations 54
4.8.2 Core Architecture Explorations 56
4.8.3 Re-evaluating Prior Works 58

5 Efficient Half Page Access **62**
5.1 Motivation 62
5.1.1 Row Buffer Overfetch Problem 63
5.1.2 Refresh as Technology Scales 64
5.2 Re-organizing the Sub-array 65
5.3 Half Page DRAM 67
5.3.1 Proposed Design 67
5.3.2 Memory Controller Support 69
5.4 Charge Recycling Refresh (CRR) 69
5.4.1 Proposed Design 73

5.4.2 Extending to Multiple Rows 73
5.5 Detailed Analysis 77
5.5.1 Additional Parallelism 77
5.5.2 Energy Saved Using Half Page DRAM 78
5.5.3 Charge Transfer Time of CRR 79
5.5.4 Energy Saved Using CRR 80
5.5.5 Area Overhead 83
5.6 Evaluation Methodology 84
5.7 Evaluation Results 87
5.8 Related Work 90

6 Smart Refresh 93
6.1 Cell Retention Time Characteristic 94
6.1.1 Leakage Mechanisms of DRAM Cell 94
6.1.2 Pause Refresh Test 96
6.1.3 Retention Time Distribution 99
6.2 Prior Work 102
6.3 Proposed Design 102
6.3.1 Storing Weak Cell Information Inside the DRAM 103
6.3.2 Refreshes to Weak Cells 105
6.4 Memory Controller Support 106
6.5 Evaluation Results 107

7 Conclusions 110

A Acronyms and Definitions 113

B DramDSE Configuration Example 115

Bibliography 118

List of Tables

2.1 Data rate per pin, core frequency, prefetch width, and bank group feature of various DRAMs. 13
2.2 The peak bandwidth and two factors determining it, the I/O width and the number of channels, for various DRAMs. 14
2.3 DRAM IDD specification 16

4.1 Complete list of architecture parameters. An example architecture definition is listed in Listing B.1 of the Appendix. 38
4.2 Complete list of technology parameters. An example technology configuration definition is listed in Listing B.2 of the Appendix. 41
4.3 IDD and power % error from measured values compared to the results of DramDSE. Parenthesis represents its contribution to the total power. 44
4.4 IDD and power % error from measured values compared to the results of DramDSE. Parenthesis represents its contribution to total power. . 47
4.5 System Configuration 52
4.6 Workload Setup 53

5.1 Percentage of charge recycled as charge transfer time and temperature vary. 80
5.2 System Configuration 84
5.3 Workload Setup 85

6.1 Number of failed rows in 4 different 1GB DDR3 SO-DIMM parts and each of the chips on the DIMM when refreshes are issued at $2 \cdot t_{REFI}$. 101

6.2 Number of failed rows for the first two DDR3 SO-DIMM of Table 6.1. The refreshes are issued at $4 \cdot t_{REFI}$ and only the results of the first two parts are shown. 101

A.1 List of acronyms used throughout this thesis. 114

List of Figures

2.1 Bird's-eye view of the DRAM cell array (left) and the cross-sectional view of the enclosed yellow box (right) are shown. The storage node capacitor colored in red and the bitline (BL) colored in blue connects to the source and drain of the access transistor that is controlled by the wordline (WL) colored in orange. 6

2.2 Logical view of a DRAM cell is shown. Data stored on the storage node capacitor leaks over time through the access transistor as shown on the right. 7

2.3 Cell access sequence. (a) both the bitlines and the sense amplifier (BLSA) are precharged, (b) activate command connects the cell to the bitline initiating charge sharing, (c) data transferred to the bitline is sensed and amplified by the BLSA, and (d) read/write command transfers data between the BLSA and IO wires. 9

3.1 Hierarchical organization of DDR DRAM in bottom-up order. Cells connected to the wordline (WL) and bitlines (BL) form a MAT. Collections of MATs form sub-arrays which are further grouped into banks, DRAM chips, and ranks within a DRAM module. 19

3.2 Hierarchical wordline structure: wordline is segmented to SWLs and each SWL is selected using MWL and FX sent from the row decoder. 21

3.3 Hierarchical data wire structure of BL – SIO – LIO – GIO. CSL selects which BL connects to the SIO and IOSW transfers data from SIO on the activated sub-array to LIO. LIO then connects to GIO. 23

3.4 SWLs are driven from SWDs on left and right of the MAT in an interleaved manner. SWD is shared with two adjacent MATs on a sub-array. 24
3.5 One dummy sub-array (sub-array$_0$) is added on top of the 32 sub-arrays that form a bank in an open bitline scheme (left). BLSA is shared between adjacent sub-arrays in an interleaved bitline structure (right). 25
3.6 A typical MAT consists of (1) 512 logical wordlines and 512 logical bitlines, (2) dummy wordlines and bitlines, and (3) redundant wordlines and bitlines. 27

4.1 The level of abstractions used in our DRAM modeling framework. Although it is not explicitly shown, *sub-DRAM* consists of bank(s) and row/column decoder(s). In this example, sub-DRAM is formed with four banks and two row/column decoders that are shared with adjacent banks. 34
4.2 DramDSE models DRAM using the configuration parameters that define architecture and technology as inputs and reports the area, energy breakdown and IDD values. This work is a collaboration work with SK Hynix, one of the three major DRAM manufacturing company. . 34
4.3 Side-by-side comparison of the DramDSE's visualization feature result (left) and the trace of the actual die (right) for 20 nm 8Gb HBM Gen2 [4]. 35
4.4 Modeling result of 2y nm 8Gb LPDDR4 using DramDSE. 45
4.5 Modeling result of 2x nm 2Gb HBM Gen1 using DramDSE. 46
4.6 2y nm 8Gb LPDDR4 power breakdown for each categories (a) row, (b) refresh, (c) read and (d) write. 49
4.7 2x nm 2Gb HBM power breakdown for each categories (a) row, (b) refresh, (c) read and (d) write. 51
4.8 Energy/bit of 32Gb LPDDR4 built with QDP of 8Gb LPDDR4 and SDP of 32Gb LPDDR4. The I/O component includes energy consumed by the read driver and termination. 54
4.9 Energy/bit of 8Gb LPDDR4 for various MAT structures. 58

4.10 Energy/bit of 4 Hi-stack of 2Gb HBM for various MAT structures. Base die energy is not included. 59
4.11 Bank conflict caused by SALP. Two rows, row_0 and row_1, are on different salp-subarrays but are placed on adjacent sub-arrays. The bitlines and BLSAs that store the cell's value on row_0 are marked in orange. . 60

5.1 Row buffer overfetch problem shown for 8n-prefetch DDR4 DRAM. Data access starts by fetching data stored on 8,192 cells to the row buffer. But only 64 bit data is transferred in each column operations wasting most of the energy spent to fetch data to the row buffer when only few column requests are issued to the row. 63
5.2 Ratio of row, refresh, and refresh/row energy for 4Gb DDR4 relative to 4Gb DDR3 that are manufactured from the same company. 64
5.3 Distributed refresh issued every tREFI (top) and refresh issued in a burst followed by long periods of no refreshes (bottom) [5]. 65
5.4 Comparison of the wordline hierarchy between the (a) baseline sub-array and (b) proposed sub-array. Both sub-array design enables half page access. To manage the RC wire delays, each vertical set of FX wires is driven by its own set of repeaters (buffers) that are placed in the "hole" above the sub-wordline drivers. 66
5.5 Comparison of the data wire connections between the (a) baseline sub-array and (b) proposed sub-array. SIO wires are doubled on proposed sub-array. 68
5.6 Conventional refresh done to the cell on the left: (a) Idle. BL and BLB are equalized, and cell is isolated from the sense amplifier. (b) Refresh. Bitlines and cell are fully restored using the charge supplied by the power supply lines, SAP and SAN. 70

5.7 CRR done to the cell on the right by recycling charges from the cell on the left: (a) Charge Recycling. Bitlines on the left cell supply half of the charge needed to fully restore the bitlines on the right cell. (b) The rest half of the charge needed to fully refresh the cell on the right is supplied by the SAP and SAN. 71
5.8 Design of the proposed charge recycling refresh scheme (top) and its operation (bottom). Switches are added between two adjacent MATs that are on different half pages. Charges are recycled from the bitlines of even half page to odd half page through SAP and SAN wires by turning switches on. 74
5.9 Comparing refresh sequence and refresh scheduling scheme between conventional and $\times4$ CRR when four full page rows are refreshed by issuing 4 refresh commands. (a) Order of the rows refreshed in conventional refresh. (b) Conventional refresh scheduling scheme where refreshes are issued periodically every tREFI and each refresh consuming tRFC time to complete refresh. (c) Order of the rows refreshed in $\times4$ CRR. Charges are recycled to half page rows except for even half page of first refresh. (d) Proposed refresh scheduling scheme where 4 refreshes are issued back-to-back. 76
5.10 SPICE simulation of CRR when charge transfer time is 20ns; bitline development (top) and V_{ss} current (bottom). Narrower and shorter current peak shown during amplification of odd half page bitlines indicates that less charge is supplied by the power supply than before. . 79
5.11 Refresh energy saved for V_{DD} power supply with CRR at 85°C. tRFC change as charge transfer time varies is also shown for 8Gb DDR4. . . 81
5.12 DRAM standby energy saved when CRR is used during self-refresh at 25°C. Refresh energy portion during self-refresh mode is assumed to be 80% of DRAM energy [6]. 82
5.13 Energy breakdown of each workloads for DDR4 (top) and LPDDR4 (bottom). 86

5.14 Weighted speed-up [7] improvements of proposed Half Page DRAM and CRR as well as when both schemes are combined for DDR4 (top) and LPDDR4 (bottom). 88
5.15 Energy/bit saved by proposed Half Page DRAM and CRR as well as when both schemes are combined for DDR4 (top) and LPDDR4 (bottom). 89
5.16 Weighted speed-up improvements (top) and energy saved (bottom) by Fine-grained activation and Half-DRAM relative to our Half Page DRAM. 91

6.1 Flow chart of the pause refresh test routine used to characterize DRAM cell retention time. The pause refresh time will increment in every test routine and the retention time of the cells is the last pause refresh time without an error. 97
6.2 Block diagram of the pause refresh test routine shown in Figure 6.1 implemented in Xilinx Artix-7 and Zynq SoC FPGAs. Traffic generator schedules write and read commands based on the test routine and stores the test results to the FIFO. Then, the content of the FIFO is read and transferred to the host computer using UART to be displayed on the terminal. 98
6.3 Cumulative bit failure probabilities with t_{ret} from 100 ms to 10 s and ambient temperatures from 30°C to 80°C. 99
6.4 Probability distribution of the bit failures with t_{ret} from 1 s to 8,192 s and ambient temperatures of 30°C, 50°C and 70°C. 100
6.5 Refresh scheduling done by the memory controller for (a) conventional DRAM and (b) DRAM with smart refresh scheme. 106
6.6 Weighted speed-up improvements (left axis) and energy saved (right axis) by proposed smart refresh for various workloads. 107
6.7 Energy saved by REFLEX and Multiple CRR relative to smart refresh. 108

Chapter 1

Introduction

Dynamic Random Access Memory (DRAM) is used as main memory for a wide class of computer systems ranging from large-scale data centers to battery operated mobile devices. Today, a typical personal computer has 8 GB of DRAM, formed by more than 64 billion DRAM cells. Each DRAM cell consists of a storage node capacitor that stores a single bit of data and an access transistor that selectively transfers data in and out of the DRAM cell. This simple 1 transistor 1 capacitor (1T1C) structure allowed cost-per-bit of DRAM to be attractive compared to other random access memories and contributed to the popularity of DRAM in various modern computer systems.

1.1 Constraints Posed by Modern DRAM

The storage node capacitor of a DRAM cell is stacked on top of the access transistor to form a dense cell array. The high temperature process associated with the fabrication of the stacked capacitors results in every transistor on DRAM to be much slower than even the logic transistors fabricated in a technology node that is few generations behind the DRAM. This physical limitation resulted in a relatively constant DRAM internal data fetch rate and the newer generation DRAMs had no choice but to fetch more data from the cell array to meet the high bandwidth requirement of the modern computer systems. However, routing more data wires on top of the most dense region of the DRAM, the cell array, is challenging. To address this issue, DRAM cell array

is formed in a specialized hierarchical structure where all of the resources are shared between adjacent neighboring cells. Hence, even a small change in the cell array may result in substantial overheads if the customization was done without careful consideration of the low-level process and circuit design constraints posed by modern DRAMs.

1.2 Power Consumption of DRAM

About a decade ago voltage scaling slowed and the breakdown of Denard's scaling [8] resulted in power consumption becoming the primary limitation in any computer system. A major source of power consumption in computer systems built these days is the DRAM: more than 15% of the total power consumption in a personal computer are due to DRAM [9]. DRAM power can be categorized into four categories, background, row, column, and refresh, depending on the state of the DRAM. The energy ratio between categories differs greatly depending on the memory usage patterns. This thesis focuses on the DRAM energy consumed to fetch data to the row buffer, which is the major energy consumption component for the row and refresh operations of the DRAM. Row power is dominant when there are abundant DRAM accesses, while refresh power becomes noticeable when DRAM is put idle for long periods of time. Hence, this thesis work satisfies both ends of DRAM usage cases showing significant DRAM energy savings on average, across many types of computer systems.

1.3 Thesis Contributions

The goal of this thesis is to make DRAM a better fit for modern computer systems by improving the energy efficiency of DRAM. To achieve this goal, it leverages detailed understanding of the design constraints posed by modern DRAMs to propose customized DRAM designs that introduce only small area overheads specifically. This thesis makes the following contributions:

- It provides a concise discussion of the cost-per-bit optimization done in DRAM.

In particular, it explains how the serialization and de-serialization of data transferred between the cell array and the I/O interface due to the slow DRAM transistors, and the hierarchical wire structure on the cell array to reduce the RC loading caused by sharing the wires with many neighboring cells.

- It introduces DramDSE, a new open-source DRAM modeling framework, using the constraints posed by modern DRAMs. For the first time in public domain, detailed power breakdown of two-state-of-the-art DRAMs from the industry, LPDDR4 and HBM, are provided by modeling those DRAMs using DramDSE. The correctness of the model we built is validated with mass-produced LPDDR4 and HBM parts which, to the best of our knowledge, is the first public DRAM modeling work to be validated in this way.
- It proposes three energy reduction schemes, one reducing the row energy and the other two reducing the refresh energy. First, we propose a new sub-array design that enables a unique way of accessing half page row to mitigate the row buffer overfetch problem in multi-core systems and to reuse charge from fully refreshed cells to the cells that are being refreshed. Then, we characterize the retention time distribution of DRAM cells and use this information to retain data stored on DRAM cells reliably even with less frequent refreshes. All three energy reduction schemes introduce the smallest area overhead in their respective fields. Two refresh energy reduction schemes also work during self-refresh mode where refresh energy is most significant but many prior refresh energy reduction work fail to work.

1.4 Thesis Organization

The next chapter reviews the structure of a DRAM cell and discuss the process technology used in building such cells which makes DRAM unique compared to other CMOS technologies. The same chapter also discusses two basic operations of DRAM: 1) how data are stored and read from the cell, and 2) how the data stored on the cell is retained over time. Since the two basic operations of DRAM are common regardless

of its type, the chapter shows how DRAM power consumption can be broken down into four different categories and briefly introduces a well known method of estimating the DRAM power consumption.

Chapter 3 reviews DRAM's basic hierarchical structure. While this organization is well known, we repeat it here to allow us to clearly define the terms we will use throughout this thesis. The chapter also reviews DRAM core constraints that force commodity DRAMs to all have a common hierarchical structure. It then introduces how remapping of faulty cells to functional redundant cells are performed followed by how this repair operation affects functionality of DRAM which is often neglected in DRAM related studies.

With this background, Chapter 4 describes the details of the new DRAM modeling framework we built. The chapter also introduces modeling results of two state-of-the-art DRAMs, LPDDR4 and HBM, and use those results to validate the correctness of the model itself. Power breakdown for each DRAM operations are then analyzed in detail to show both the difference as well as the commonality between LPDDR4 and HBM. The usefulness of the DRAM modeling framework we built is demonstrated by using it to re-evaluate the overheads introduced by some prior DRAM research proposals.

Three DRAM designs that improve the energy efficiency of DRAM are then proposed. This thesis focuses on improving the row and refresh energy where the energy consumed to charge and discharge bitlines along with the cells is the dominant energy consuming component during both row and refresh operation for all commodity DRAMs. Chapter 5 proposes a new sub-array design that allows half page row access along with two extensions improving both row and refresh energy efficiency. Then, Chapter 6 proposes to utilize the retention time characteristic distribution of DRAM cells to improve refresh energy efficiency even further. The chapter proposed to store the weak cell information in anti-fuses on each of the chips during wafer-level test. The effectiveness of the proposed designs is presented by evaluating the area, performance, and energy consumption on a computer system running various workloads.

Chapter 2

DRAM Basics

In this chapter, we start by analyzing the characteristics of a single 1T1C DRAM cell focusing on the properties that make DRAM unique compared to other CMOS technologies. Then, we discuss the need for both the data access and refresh operations which explains why DRAM is a ***D**ynamic **R**andom **A**ccess **M**emory*. Finally, we will discuss how data is transferred in and out of the DRAM, the factors affecting DRAM bandwidth, and DRAM power consumption estimation methods.

2.1 DRAM Cell

In a Gb-scale DRAM, there are billions of DRAM cells, each storing a single bit of data. A typical DRAM cell array is shown on the left of Figure 2.1. Each striped string enclosed in a yellow box consists of two DRAM cells and the cross-sectional view of the enclosure is shown on the right of Figure 2.1. A DRAM cell consists of a transistor and a capacitor where the data is stored on the capacitor colored in red of Figure 2.1. Since the capacitor works as a storage element, it is often referred to the *storage node capacitor* and is stacked on top of the transistor as shown on the right of Figure 2.1. The data is transferred between the storage node capacitor and the bitline (BL), colored in blue, using the transistor named *access transistor*. The wordline (WL), colored in orange, connects to the gate of the access transistor and selectively connects the storage node capacitor to the bitline allowing data stored

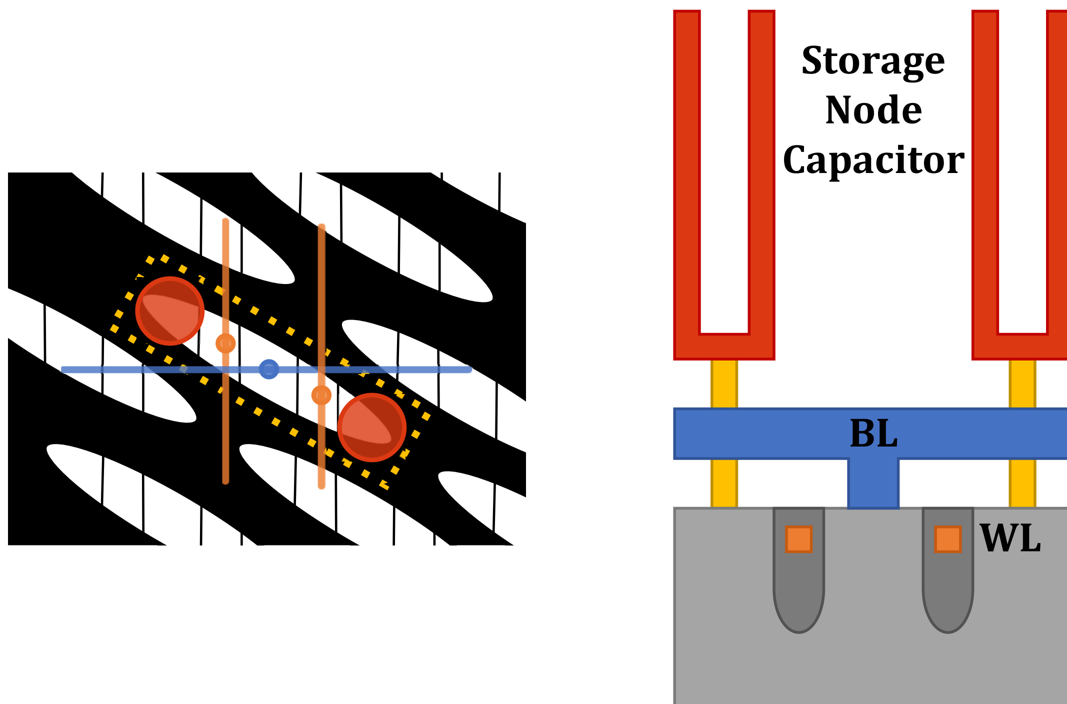


Figure 2.1: Bird's-eye view of the DRAM cell array (left) and the cross-sectional view of the enclosed yellow box (right) are shown. The storage node capacitor colored in red and the bitline (BL) colored in blue connects to the source and drain of the access transistor that is controlled by the wordline (WL) colored in orange.

on the cell to be accessed or retained depending on whether the access transistor is turned on or off. The left of Figure 2.1 shows the layout of the recent $6F^2$ cell architecture, where F denotes the minimum feature size. In other words, the area of a cell is $6F^2$ in the state-of-the-art DRAMs resulting in both the bitline and the wordline pitch to be slightly above 2F. This allows the DRAM to form a dense cell array structure but also introduces many constraints in routing the wires and placing the drivers as it will be discussed in more details on Section 3.2.2.

The logical view of a DRAM cell we discussed is shown on the left of Figure 2.2. Again, the storage node capacitor that stores the data connects to the bitline (BL) using a NMOS access transistor controlled by the wordline (WL). The cell retains data when the wordline is low and the NMOS access transistor is off. However, the charge stored on the storage node capacitor that represents the data does not maintain its

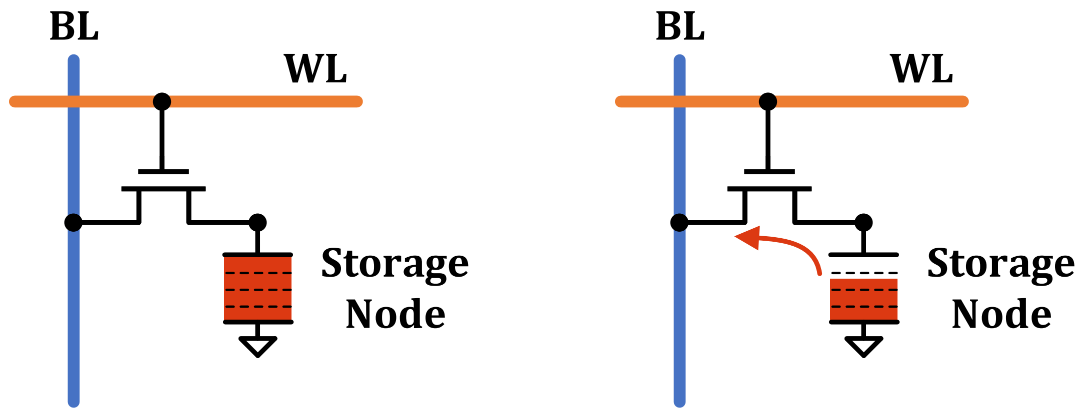


Figure 2.2: Logical view of a DRAM cell is shown. Data stored on the storage node capacitor leaks over time through the access transistor as shown on the right.

value indefinitely as it leaks over time through the NMOS access transistor as shown on the right of Figure 2.2. When enough charge leakes off the cell, the data stored on the cell is lost resulting in an error. Hence, it should not be surprising that the advancement of DRAM process technology has been geared towards achieving long data retention time, which can be achieved by reducing the leakage of the access transistor and increasing the storage node capacitance.

The dominant leakage mechanism on the access transistor is the junction leakage [10] and it is affected by the doping concentration of the substrate and the junction area. Junction as well as the sub-threshold leakage of the access transistor are reduced by increasing the physical channel length of the transistor. To achieve the required channel length while the dimensions scale, DRAM has been using recessed channels [11, 12]. The shape of the recessed channel varies by generations [13] but it is a trench that has been formed by etching out silicon where the current flows around this trench. Recessed transistors use the third dimension to increase the length of the transistor, in contrast to modern logic technologies (FinFETs) which use the third dimension to increase the width of the transistor. Recent $6F^2$ cells are using buried wordline [14] where, as the name indicates, the wordlines are also below the silicon surface. This structure reduces the coupling between the wordline and the bitline improving sensing margin and reducing power consumption. Sensing margin is a

measure of how reliably data stored on the cell can be sensed under the worst operating conditions of DRAM. Large sensing margins are needed to deal with multiple sources of noise, power supply fluctuation, and charge loss due to leakage.

Commodity DRAM also use the third dimension to increase the capacitance of the storage node capacitor by stacking it above the transistor [15] as shown on the right of Figure 2.1. The stacked capacitor enables one to maintain capacitance as technology scales by growing vertically and having 3D surfaces [16]. However, since this structure is fabricated after the silicon transistors, the high temperature process associated with its fabrication affects every transistor on the DRAM. The higher thermal processing time makes junctions deeper than normal technologies, which means that channel lengths must be longer to control leakage. All of these issues make even the leakier DRAM transistor slower than planar logic transistors. Today, the aspect ratio or the ratio between the height and the width of the storage node capacitor is becoming the major bottleneck in continuing the technology node scaling of DRAM as it is becoming increasingly difficult to push the aspect ratio higher. The state-of-the-art 1x nm technology node already achieved an aspect ratio of 60 : 1 [17].

2.2 Basic Operations

DRAM has been evolving by adding new features to operate at better performance and lower power consumption compared to the previous generation [18, 19, 20]. However, there are two operations that are common to every DRAM: *data access* and *refresh*. The two operations self explains the name DRAM (Dynamic Random Access Memory) and enables DRAM to function as a reliable memory.

2.2.1 Data Access

The simplified DRAM core shown in Figure 2.3 consists of a cell connected to the bitline sense amplifier (BLSA). Before accessing data stored on the cell, the BLSA has to be precharged as shown in Figure 2.3a. This is done by enabling EQ (❶) that shorts the bitline (BL) and the reference bitline (BLB) with each other and to the precharge

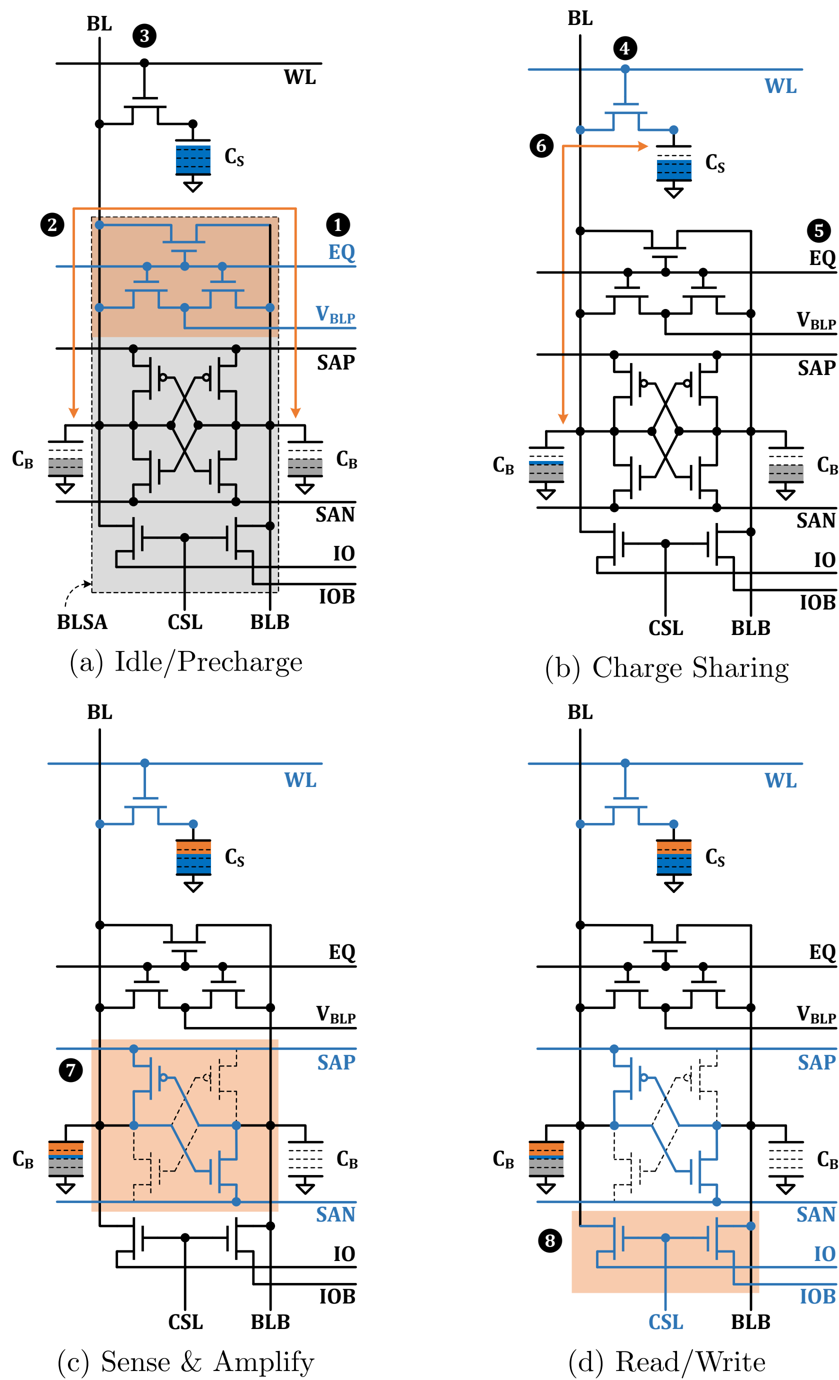


Figure 2.3: Cell access sequence. (a) both the bitlines and the sense amplifier (BLSA) are precharged, (b) activate command connects the cell to the bitline initiating charge sharing, (c) data transferred to the bitline is sensed and amplified by the BLSA, and (d) read/write command transfers data between the BLSA and IO wires.

voltage (V_{BLP})[1] as shown in ❷. The data stored on the cell is unaffected with the BLSA being precharged by deselecting the wordline (WL) (❸) and isolating the storage node capacitor (C_S) from the BL. Once the BLSA is completely precharged, a sequence of three commands, namely activate, read/write and precharge, can be issued to access data stored on the cell.

An *Activate* command provides the row address and selects a row of cells by enabling the WL as shown in ❹ of Figure 2.3b. EQ is also disabled (❺) so that the data stored on the selected cells can be transferred to the BL (❻). This process is often referred to as *charge sharing*, since the charge stored on C_S is shared with C_B on the BL side to create a small voltage deviation between the BL and the BLB. It is a destructive operation because after charge sharing, the voltage in the cell is equal to the BL voltage. Once a sufficient amount of charge is transferred from the cell to the BL, the BLSA can be powered-up to sense and amplify the value on the bitlines to full digital value as shown in ❼ of Figure 2.3c. The charge stored on the cell that were lost by leakage and charge-sharing are also restored as BL is amplified by the BLSA to full logical levels.

Once the data stored on the row of cells are read using the BLSA, a column command such as *Read/Write* is issued to further specify the location of the data to be accessed. Both read and write commands operate on the BLSA by asserting the column select line (CSL) decoded from the column address provided with the commands. CSL connects the bitlines to the IO wires as shown in ❽ of Figure 2.3d to transfer data out of the cell to the IO for read or to update the value stored on the BLSA, which will eventually update the cell, for write.

Finally, a *Precharge* command is issued after the needed data to the selected row has been read/written. A Precharge command puts DRAM core back to the initial state as described earlier in this section (Figure 2.3a), making BLSA ready to sense data stored on other locations of the cell array by issuing again, the same sequence of three commands. This requirement allows DRAM to be classified as a *Random Access Memory*.

[1] Power supplies of the BLSA, SAP and SAN, are also precharged so that BL and BLB stay precharged. V_{BLP} is set to midpoint of SAP and SAN for fast and low power reads.

2.2.2 Refresh

The charge stored on the DRAM cell leaks over time and the data stored on the cell will eventually flip bit if the cell continues to leak charge. Hence, it is important to periodically read the value stored on the cell and write the restore value back to the cell for DRAM to function as a reliable memory. This requirement is the reason why DRAM is classified as Dynamic RAM and the overall operation to fulfill such requirements is called *Refresh*.

All three commands, activate, read/write and precharge, are needed for data access as discussed in Section 2.2.1. To refresh the cells, only the functionality of activate and precharge commands are needed. The activate command senses data stored on selected cells and amplifies it to full digital value while precharge command deselects the selected cells to retain data that has been restored as shown in Figure 2.3a-c. Hence, refresh can be thought of as a fused operation that issues activate and precharge commands to multiple rows in an automated sequence.

A Modern DRAM has dedicated refresh commands that can be used periodically to guarantee correct operation of DRAM and depending on the state of the DRAM, there are two different types of refresh modes. The first and the default type is *auto-refresh*, where the memory controller issues refresh commands periodically to indicate when the refresh should occur while the DRAM determines the rows to refresh. Another type of refresh is *self-refresh*, which refreshes data when the DRAM is in a low power sleep mode. In this mode, refreshes are issued automatically by the DRAM without any additional command issued from the memory controller. As the self-refresh mode continues longer, the operating temperature decreases and DRAM adjusts the rate at which refreshes are to be issued based on the temperature reading from the temperature sensor embedded on the DRAM [5].

The requirement for refresh is provided by the JEDEC standards [21, 22, 5, 23, 24, 25] and is usually represented as the number of refresh commands that have to be issued within a certain time window (t_{REFW}). These two parameters and the number of rows determine the interval between refresh commands (t_{REFI}) and the refresh cycle time (t_{RFC}). For instance, 8Gb LPDDR4 DRAM [24] requires 8,192 refreshes to be issued within a 32 ms time window to guarantee data stored on DRAM cells retain

their value. Since there are total of 256 K rows per channel 32 rows are refreshed by a single refresh command, which must be issued on average, every 3.9 μs, where each refresh command takes 180 ns to perform. In this example, t_{REFW} is 32 ms, t_{REFI} is 3.9 μs and t_{RFC} is 180 ns.

As DRAM density increases, more rows are usually added to maintain the *page size*: the data fetched to the bitline sense amplifier or the number of cells connected to a selected row by the activate command. This means that more rows are refreshed in each refresh command and generally, the time to perform a refresh or t_{RFC} increases due to power constraints. Data accesses are restricted for the entire t_{RFC} during auto-refresh and the increase in t_{RFC} placed high demands on special refresh modes, such as Fine Granularity Refresh [22] and per-bank refresh [5, 23, 24, 25], to maximize the bandwidth utilization ratio.

2.3 Data Transfer

The previous section discussed the data storage aspect of DRAM. Another important aspect of DRAM is how data is transferred in and out of the DRAM. Modern DRAM uses Double Data Rate (DDR) protocol to exchange data between the DRAM and the memory controller. DRAM technology has been evolving over the years by doubling the data transfer rate of the DDR protocol without changing the data transfer rate within the cell array, also known as the *core frequency*. In this section, we will discuss how DRAM technology evolution was achieved by first looking at what DDR is and how it relates to the core frequency. Then, we discuss the relationship between the DDR generation and the memory bandwidth provided by typical DRAM memory systems.

2.3.1 Double Data Rate and Core Frequency

The data transferred between the DRAM and the memory controller has valid information on both the rising and the falling edge of the clock, giving the name Double Data Rate (DDR). Over the years, DDR protocol evolved by doubling the pin data

	DDR3	**DDR4**	**LPDDR3**	**LPDDR4**	**HBM Gen1**
Data Rate	1.6 Gbps	3.2 Gbps	1.6 Gbps	3.2 Gbps	1 Gbps
Core Frequency	200 MHz	200 MHz	200 MHz	200 MHz	250 MHz
Prefetch	8n	8n	8n	16n	2n
Bank Group	N	Y	N	N	Y

Table 2.1: Data rate per pin, core frequency, prefetch width, and bank group feature of various DRAMs.

rate from the previous generation as shown in Table 2.1. However, the core frequency, that measures how fast the data can be transferred in and out of the cell array, remained relatively constant ranging from 200 MHz to 250 MHz even across many generations of DDR protocols. This helped continuous evolution of DDR DRAMs until now, as most of the changes are done on the periphery not on the cell array or the core that is sensitive to even the smallest changes.

The data is transferred between the cell array and the DRAM periphery using the Global IO (GIO) wires. DRAM internal bandwidth is proportional to the core frequency and the amount of bits fetched from the cell array. The bits fetched from the cell array by a column command is called the *prefetch width* and is often presented as xn-prefetch on the DRAM datasheets. 8n-prefetch of LPDDR3 [23] shown on Table 2.1 indicates that 8× the total number of I/O pins are fetched from the cell array in parallel and transferred to the GIO whenever a read command is issued. The data on the GIO wires are then serialized to transfer data at 1.6 Gbps/pin for total of 4 clock cycles to the memory controller. During write, the incoming data collected for 4 clock cycles are de-serialized on the periphery and the entire data is then transferred to the cell array in parallel at a much slower 200 MHz core frequency. Limiting the high speed operation on a small region (periphery) of the DRAM and running most of the region (cell array) at a much lower rate, enabled DRAM to be cost effective over the years even though it had to evolve to have have a higher data rate.

Interestingly, DDR4 operates at 8n-prefetch, same as DDR3, but has a 2× pin

	DDR3	DDR4	LPDDR3	LPDDR4	HBM Gen1
Total I/O	64	64	64	64	1,024
Channels	1	1	1	2	8
Bandwidth	12.8 GB/s	25.6 GB/s	12.8 GB/s	25.6 GB/s	128 GB/s

Table 2.2: The peak bandwidth and two factors determining it, the I/O width and the number of channels, for various DRAMs.

data rate compared to that of DDR3 as shown in Table 2.1. This was achieved using a feature called *bank grouping* [26]. This feature groups a couple of banks and physically separates the GIO wires from banks in different groups, allowing data to be sent independently between the groups, each at the rate of the core frequency. Thus, 8n data fetched from two independent bank groups can be sent at 2× core frequency but only 1× core frequency for the same bank groups. This feature is a low cost solution in providing additional bandwidth and is also used in many high bandwidth DRAMs such as HBM: the cost of increasing the core frequency and fetching more data than what they already provide is getting pricey. It should be noted that HBM already operates at a higher core frequency[2] of 250 MHz – 500 MHz paying a price premium to achieve such high data rates.

2.3.2 Bandwidth and DRAM Configuration

Multiple DRAMs are grouped together to meet the density as well as bandwidth requirement of the system. Table 2.2 shows the peak bandwidth for a typical DRAM configuration used in modern computer systems. The peak data rate is set by the pin data rate described in Table 2.1 times the total I/O width. For instance, HBM operates at only 1 Gbps/pin but provides the highest peak bandwidth of 128 GB/s using 1,024 I/O pins. The enormous 1,024 I/O pins are possible by having 8 channels of HBM each with 128 bit I/O pins. DRAM channels also contribute to the overall

[2] Graphics DDR (GDDR), another high performance DRAM, operates at even higher core frequency of 875 MHz to support 7 Gbps/pin data rate of GDDR5 [26]. Unlike HBM, bank grouping feature is often not used to provide low access latency and better data transfer efficiency [26, 27].

DRAM bandwidth as channels can be operated independently from each other. But adding more channels requires a separate memory controller for each of the channel. Since the memory controllers are embedded on the processor, the cost of adding additional channels to increase the peak bandwidth is high. This cost to bandwidth relationship resulted in DDR3 and DDR4 to be popular in desktops and large-scale servers, whereas LPDDR3 and LPDDR4 have been used mostly in premium battery operated mobile devices, and HBM in high-end accelerators and networking equipment.

2.4 Power Consumption

Based on the DRAM operations mentioned earlier in Section 2.2, we can break DRAM power into four categories; background, row, column, and refresh. *Background* is the power that is constantly being dissipated on top of the power consumed in each DRAM operations. The background power is mostly leakage and depends on how many banks are activated and whether the device is in a special power-down mode. The other three power categories are the dynamic power dissipated to perform the specific DRAM operations. *Row* power relates to activate and precharge commands, *Column* is the power spent to read and write data in and out of the DRAM, and *Refresh* power is dissipated whenever periodic refresh is issued.

2.4.1 IDD Specification

The power consumption of a DRAM is specified through a set of chip currents (IDD) on the datasheet for different types of operations. A brief description of what the various IDD symbols represent is listed in Table 2.3. IDD2 and IDD3 measure the static leakage power of a DRAM when no banks or one/all banks are activated. IDD0 gives the row power consumed to activate and precharge a bank, while IDD4 measures the column power used to read and write data. The power dissipated to refresh cells is measured by IDD5 and IDD6 which are separated by the refresh mode. IDD5 is further specified into all bank or per-bank while IDD6 is usually represented in

Symbol	Description
IDD0	one bank activate and precharge current
IDD2	idle standby current where all banks are precharged
IDD3	active standby current where banks are activated
IDD4	read (IDD4R) or write (IDD4W) current
IDD5	auto-refresh current for all banks or per-bank
IDD6	self-refresh current for normal or extended temperature range

Table 2.3: DRAM IDD specification

different temperature bins.

2.4.2 Dynamic Power Consumption

One important power component that is not captured by the IDD specification is the I/O power consumed to transfer data between the DRAM and the memory controller during read/write commands. Datasheets do not specify IDD values for the I/O because the power consumption depends heavily on how the memory system as a whole is configured, *i.e.* termination topology, the number of ranks on the data bus, and whether special features such as data bus inversion (DBI) [24] are used. When viewed from the DRAM side, I/O power can be broken down into the power consumed to drive the data bus during read and the on die termination (ODT) power during write and read, depending on the memory configuration. One popular method to estimate I/O power for DRAM with ODT is to use the Thevenin equivalent circuit method [28] with the proper ODT configuration [29] for each systems. Then, the DC power consumed on the IO bus between the memory controller and DRAM can be found by simply analyzing the Thevenin network.

$$Power_{row} = \left(IDD0 - \frac{IDD3N^* \cdot t_{RAS} + IDD2N \cdot t_{RP}}{t_{RC}}\right) \cdot V_{DD} \tag{2.1}$$

$$Power_{read} = \left(IDD4R - IDD3N^*\right) \cdot V_{DD}$$

$$Power_{write} = \left(IDD4W - IDD3N^*\right) \cdot V_{DD}$$

$$Power_{refresh} = \left(IDD5 - IDD3N^*\right) \cdot V_{DD}$$

Multiplying the dynamic portion of the IDD values by the power supply voltage (V_{DD}) gives the power-per-op [28] that DRAM consumes for the components other than the I/O, as shown in Equations 2.1[3]. The power-per-op can be converted into energy-per-op by multiplying the resulting power by the time needed to perform each operation, which is $t_{RC} = t_{RAS} + t_{RP}$ for row, t_{CCD} for both read and write, and t_{RFC} for refresh. All of these parameters used to be specified in the DRAM datasheet [30, 31] but the IDD values are often proprietary for the newer generation devices such as LPDDR4 and HBM. There are also limited number of publicly available DRAM modeling tools that can generate these IDD values [32, 33, 34].

Similar to other Very-Large-Scale Integrated (VLSI) chips, the dynamic power consumption of DRAM is determined by the on-chip interconnect. Hence, to understand the source of each dynamic power consumption, it is important to have a better understanding of how DRAM is designed. The next chapter discusses how billions of 1T1C cell are connected to each other to form a DRAM and the constraints caused by how DRAM is currently designed.

[3]IDD3N* is the effective active standby current where the leakage current for the number of row(s) that are activated for each operation is scaled from the original IDD3N.

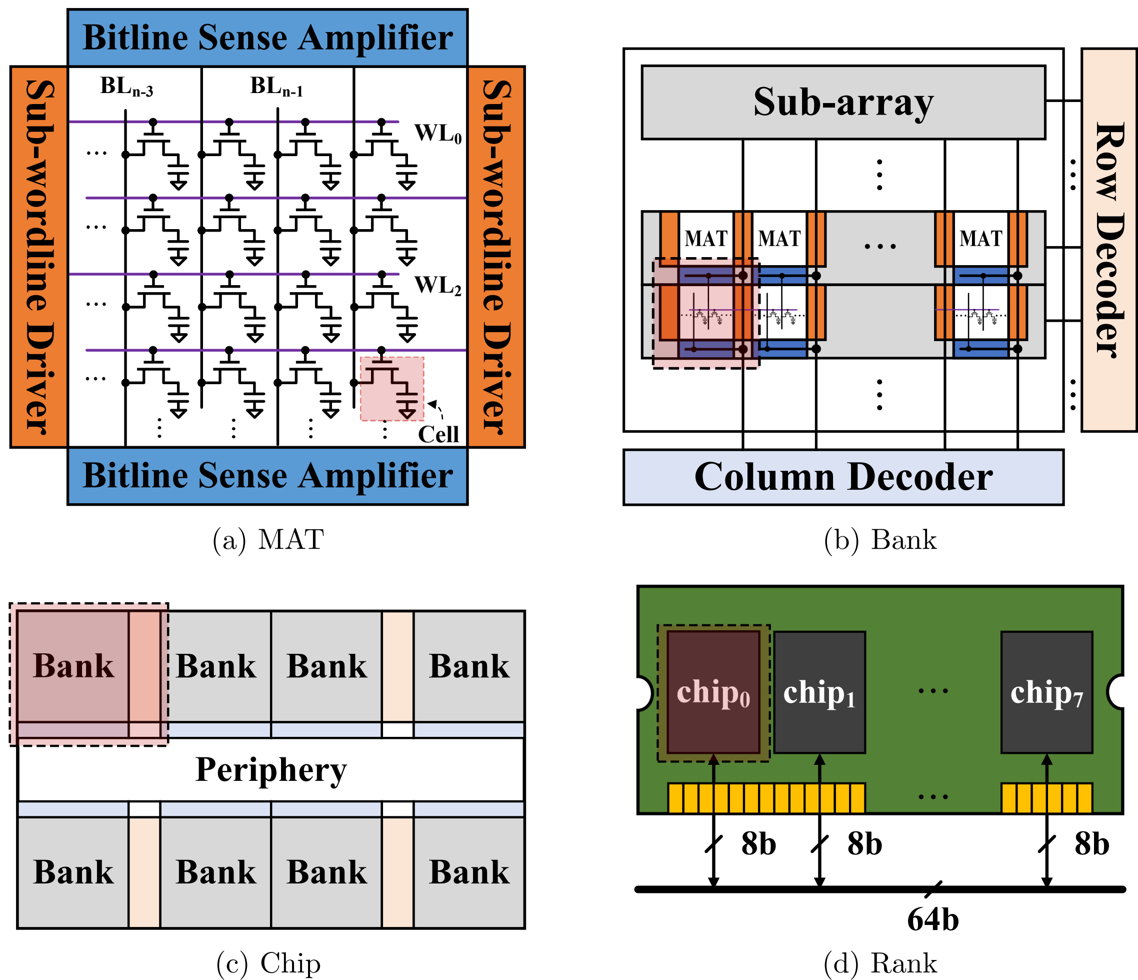
Chapter 3

DRAM Design Constraints

Both the process and circuit design of DRAM is highly optimized to reduce the cost-per-bit as low as possible since tens of billions of DRAM chips are manufactured and shipped worldwide every year [35]. For example, even though DRAM uses leading edge technology nodes (1x nm), the high temperature processing needed to create the stacked capacitor [36] and the need for specialized low leakage access transistor [15] means that even the "fastest" DRAM transistors are much slower than the logic transistors in a technology that is a couple of generations behind the DRAM. Even stranger is that DRAM uses very few metal layers; today most DRAMs only have 3 metal layers [20]. Both slow transistors and limited wiring layers introduce many design challenges in routing wires within the dense cell array that will be discussed in the following sections.

3.1 Structural Organization

Each DRAM cell stores a single bit of data using a capacitor and an access transistor as shown in bottom right of Figure 3.1a. The cells are then densely packed with each other, forming a 2D-matrix of cells called a *MAT*. The tight pitch of the cells, usually only slightly larger than twice the lithographic feature size in each direction, yields a



(a) MAT

(b) Bank

(c) Chip

(d) Rank

Figure 3.1: Hierarchical organization of DDR DRAM in bottom-up order. Cells connected to the wordline (WL) and bitlines (BL) form a MAT. Collections of MATs form sub-arrays which are further grouped into banks, DRAM chips, and ranks within a DRAM module.

cell area of $6F^2$ today.[1] The cell pitch (repeat distance) is less than the finest pitch of the normal metal layers, so the bitlines are wired in silicided polysilicon, and the wordlines in tungsten, a high-temperature metal with high resistance [38]. The high resistance of these layers plus the constraint on the total bitline capacitance limit the size of a MAT, and a typical configuration has 512 wordlines and 512 bitlines storing a total of 256 K bits [39]. In addition to the cells, each MAT is surrounded by peripheral circuits: *sub-wordline drivers* to drive a wordline that connects a row of cell capacitors to the bitlines, and *bitline sense amplifiers* to amplify the small signals produced on the bitlines up to full digital values.

Today's Gb-scale DRAM contains thousands of MATs arranged in a hierarchical structure to achieve the low latency and high bandwidth requirements. First, around 16 MATs are grouped into a *sub-array* [18] as shown in Figure 3.1b. Sub-arrays are then stacked on top of each other to form a sub-bank and depending on the size of the DRAM one or more sub-banks combine to form a *bank*. Similar to the MAT, the bank also has a set of peripheral circuits labeled *row* and *column decoder* to select a group of cells and transfer data into or out of these selected cells. Finally, a *chip* shown in Figure 3.1c has multiple banks to provide the bank-level parallelism crucial to high performance DRAM and additional periphery circuit that contains pads or TSVs to communicate with the system. The memory module used in modern computer systems is simply a collection of multiple DRAM chips with 64 bits of I/O per *rank* for the DDR4 DRAM as shown in Figure 3.1d.

3.2 Core Design

The wordlines and the bitlines that connect to the cells on the DRAM core are routed at the most finest pitch on the DRAM where both are less than 0.07 um in 2y nm technology node. Also, the width and height of a bank with 32 sub-arrays where each sub-array is formed using 16 MATs, are quite large, being roughly 650 μm wide and 1,500 μm tall respectively. This makes it difficult for the row decoder to drive all

[1] Note that the cell is slanted [37] so the pitch in both directions is slightly greater than 2F, rather than having a 2F$\times$3F cell.

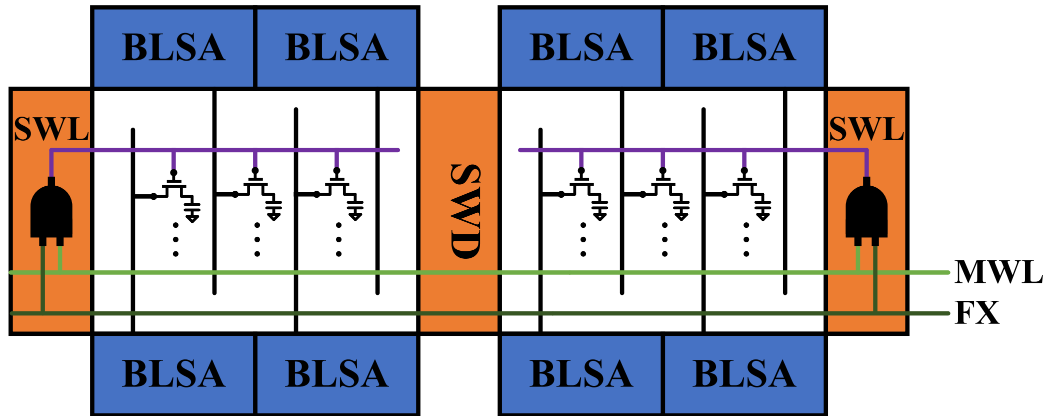


Figure 3.2: Hierarchical wordline structure: wordline is segmented to SWLs and each SWL is selected using MWL and FX sent from the row decoder.

16,384 wordlines in a bank from the row decoder. Similarly, the 8,192 bitlines on a bank cannot all go down to the column decoder. Instead, a hierarchical wordline and data wire structure is formed on the bank to efficiently select one of the wordlines on the core and transfer a prefetch width of data between the core and the column decoder. Moreover, the wordline drivers and the bitline sense amplifiers are shared between adjacent MATs within a sub-array and adjacent sub-arrays within a bank respectively, as a result of DRAM's cost-per-bit optimization. In this section, we will discuss the hierarchical wire structure as well as the cost-per-bit optimization done on the core. We will also introduce the redundant cells that are crucial to meeting commercial-level yields, and basic repair schemes that maps faulty cells to the working redundant cells.

3.2.1 Hierarchical Wires

The wordline that selects thousands of cells for each activate command is divided into a much shorter segments named the *sub-wordline* (SWL) as shown in Figure 3.2. Instead of sending all 512 decoded wordlines into each MAT, a two stage decoder is used. The main decoder provides 64 main wordline (MWL) and 8 pre-decoded row address (FX) lines. Sub-wordline drivers (SWD) on left and right of the MAT selects a SWL using the MWL and FX signals sent from the row decoder.

The MWL and FX lines are routed across the entire sub-array connecting to multiple SWDs so that every SWL forming a wordline can be selected. FX wires are often repeated in the region named sub-hole that is located below the SWD and next to the bitline sense amplifier. One of the main advantages of using the hierarchical MWL, FX, and SWL structure to select a wordline is the reduction of wire RC delays. By reducing the number of long global wires, the MWLs and FXs are routed in a larger pitch lower resistance metal and only see the loading of the SWDs. The higher resistance SWLs are much shorter, having to drive only 512 cells. This hierarchical structure enables fast and power efficient Gb-scale DRAM.

Data wires routed from the cell array all the way to the pads or TSVs are decomposed to segmented IO (SIO), local IO (LIO), and global IO (GIO) as shown in Figure 3.3. SIO wires run horizontally over the bitline sense amplifiers (BLSA) and transport data from a few selected bitlines to the LIO wires. Since the number of SIO wires is small, the collection of SIO wires on all the MATs within a sub-array will provide the entire data for a column access. The column select line (CSL) decoded from the column address, determines which bitline should connect to the SIO wire. Data on the SIO wires are then transferred to the LIO wires that run vertically across the entire bank. The IO switch (IOSW) isolates the SIO wires in other deselected sub-arrays in the same bank from loading the LIO wires. This is critical since both SIO and LIO wires are small swing differential signals that are initially precharged to V_{CORE}, the power supply voltage used in the cell array, and during reads there are no drivers on the SIO wires except for the wimpy bitline sense amplifier. Without perfect isolation from SIO wires on non-active sub-arrays, whatever small data that was on the SIO wires of the active sub-array will be wiped out to V_{CORE}. This loading constraint must be considered for any proposal that involves accessing multiple sub-arrays in a bank as has been considered recently [40, 41].

At the side of the bank closest to the pads, LIO wires connect to GIO wires which route the outputs from the selected bank(s) to the I/O pins. For a read, the small difference on the LIO wire pair is sensed by the IO sense amplifier which is in the column decoder block. For writes, write driver on the column decoder drives the LIO wires strongly so that it can even flip the BLSA. Since GIO wires are single-ended

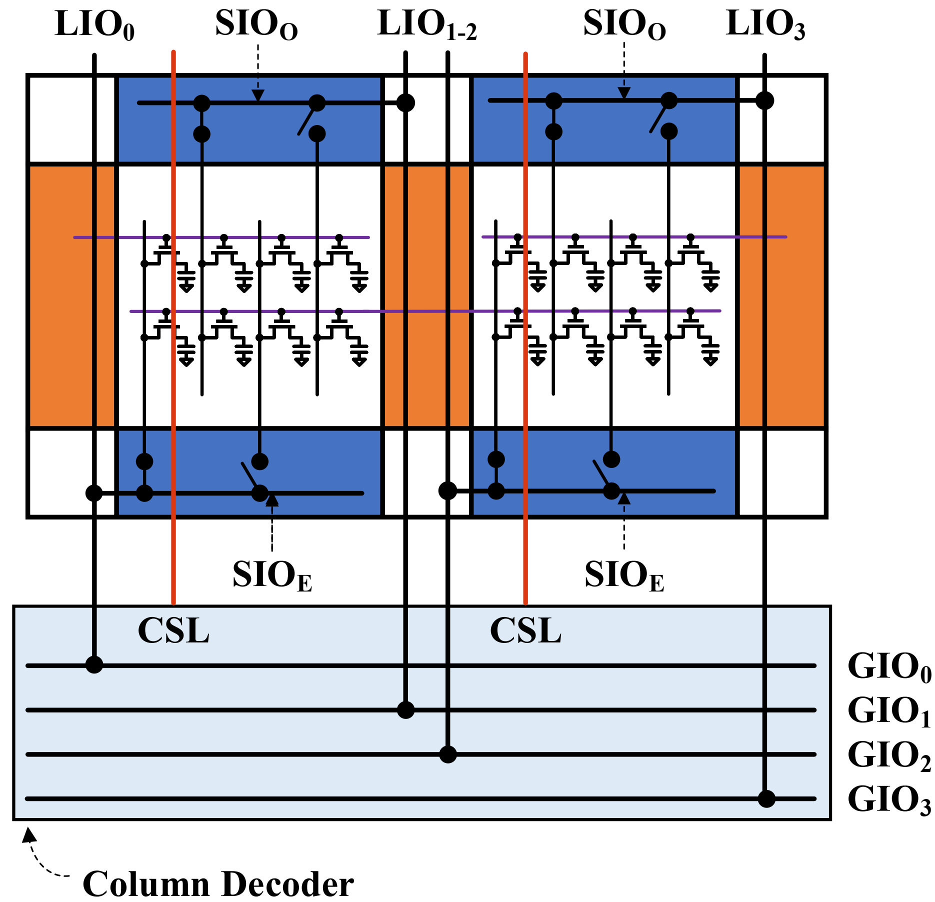


Figure 3.3: Hierarchical data wire structure of BL – SIO – LIO – GIO. CSL selects which BL connects to the SIO and IOSW transfers data from SIO on the activated sub-array to LIO. LIO then connects to GIO.

like the I/O pins, single-ended to differential conversions are also done at the column decoder.

This hierarchical data wiring reduces the capacitance that needs to transition in each operation and speeds up operations by reducing RC delays. Even with these changes, the technology, remaining long wires, and the need to reduce the area overhead of the peripheral circuits (BLSA, SWD, IOSW, etc.) constrain transistor sizes which means that these circuits cannot run at high clock speeds. To support high bandwidth, more data has to be fetched from the cell array on each core frequency as discussed in Section 2.3.2 to supply the data needed. The large prefetch of data

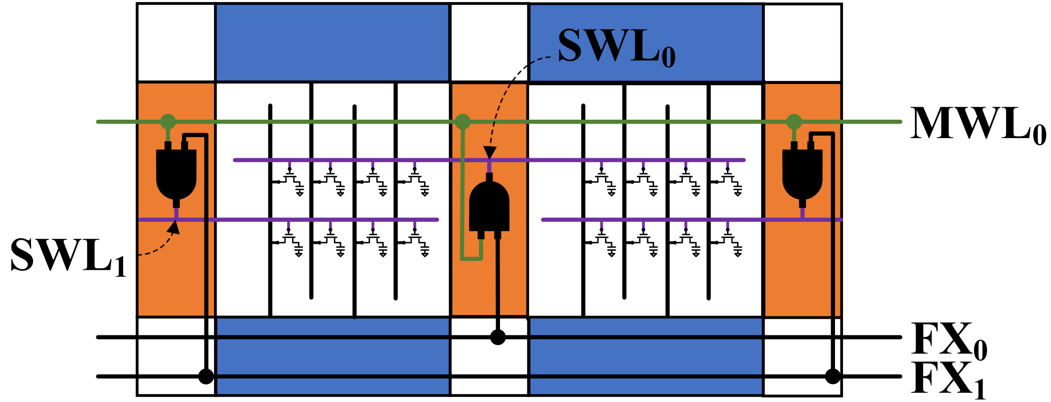


Figure 3.4: SWLs are driven from SWDs on left and right of the MAT in an interleaved manner. SWD is shared with two adjacent MATs on a sub-array.

is becoming increasingly costly as the trend is to reduce the page size for energy efficiency [24, 25].

3.2.2 Shared Everything

Every wordline on a MAT must be driven by the sub-wordline drivers (SWD) and every bitline must connect to the bitine sense amplifiers (BLSA) to have access to the cells. However, the tight pitch of the wordline and the bitline makes it extremely inefficient to layout the SWD and BLSA within a wordline and bitline pitch, respectively. To address this issue, DRAM designers have been using the interleaved wordline and bitline structure [42, 43] for decades, but the impact of this designs were not discussed enough outside of the DRAM community.

Figure 3.4 shows how two sub-wordlines (SWL) in a MAT are driven from the SWD. SWL_0 is driven from the SWD on the right of the MAT where as SWL_1 is driven from the SWD on the left for the leftmost MAT. The next SWL will again be driven from the SWD on the right of the MAT and so on, forming an interleaved wordline structure. Now each SWD can occupy two wordline pitch instead of one, which makes the layout more area efficient. To further save area, each SWD is shared with the adjacent MAT in the same sub-array as shown in Figure 3.4.

While this optimization roughly halves the number of SWD needed to drive the

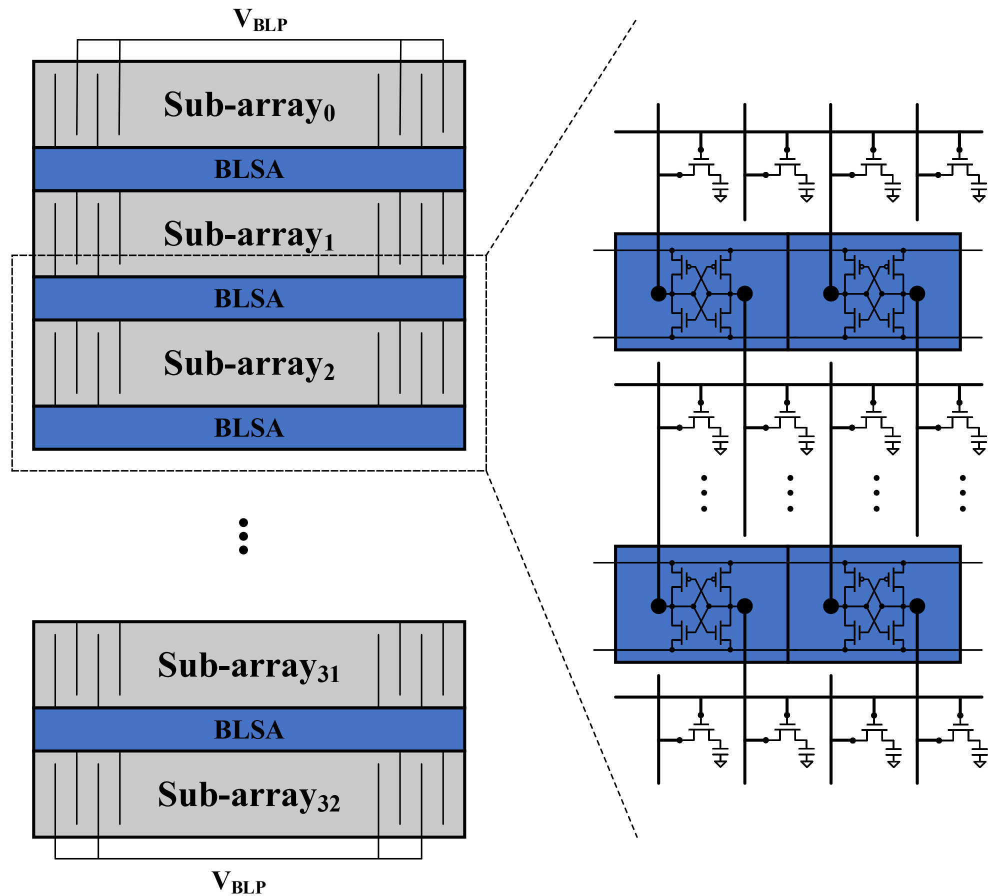


Figure 3.5: One dummy sub-array (sub-array$_0$) is added on top of the 32 sub-arrays that form a bank in an open bitline scheme (left). BLSA is shared between adjacent sub-arrays in an interleaved bitline structure (right).

wordlines, it also tightly couples the MATs within a sub-array to each other. This coupling limits proposals for fine-grained row activation [44, 42, 45], since they do not activate all MATs in a sub-array. The most efficient way to do fine-grained activation requires adding an additional SWD strip on every breakpoints of the smaller segments similar to that of DDR4 $\times 4$ half page architecture [18]. Unfortunately, this means that not all the SIO wires in the sub-array will be active, which decreases the bandwidth to this segment, unless the number of SIO or LIO wires in all MATs is increased to compensate the bandwidth loss [46].

Today's $6F^2$ cells use an open bitline scheme [43] where the reference bitline is

not on the same MAT as the bitline. DRAM designers leveraged the fact that only one row can be accessed within a bank to form a dense interleaved bitline structure for the open bitline scheme as shown in Figure 3.5. Bitlines connect to the BLSAs on above or below the MAT and each BLSA is shared between adjacent MATs on different sub-arrays to reduce area. Similar to the SWD, the interleaved structure allows the BLSA to fit in two bitline pitch instead of a single bitline pitch that yields a much more efficient layout than the alternatives. In this structure, the two sub-arrays adjacent to the one being accessed provide the reference bitlines and they are inaccessible until the bank is precharged.

The open bitline scheme incurs an overhead due to the lack of reference bitlines for the sub-arrays located on the edges of the bank. One dummy sub-array [47], sub-array$_0$, is added to provide the reference bitlines for the top edge sub-array, sub-array$_1$ shown in Figure 3.5. Half of the bitlines on the edge sub-arrays (sub-array$_0$ and sub-array$_{32}$) cannot be accessed, as they do not have the reference bitlines and these bitlines are set to the bitline precharge voltage (V_{BLP}) to reduce the noise and to improve sensing [47]. Since edge sub-arrays only provide half of the page each, both edge sub-arrays always have to be accessed in pair. The area overhead of the dummy sub-array becomes a burden as the number of sub-arrays in a bank decreases, which is the main reason why reducing the size of a bank is rarely done to increase the core frequency.

3.3 Repair Scheme and Redundant Cells

While a MAT might support 512 logical wordlines and 512 logical bitlines, the number of physical memory cells in a MAT is much larger than 512×512 for a couple of reasons – see Figure 3.6. First, there are many *dummy cells*, which are never used, but placed to improve the matching of the cells that are used. Dummy row and columns are placed where abrupt changes in layout pattern exists, to ensure that the actual memory circuitry is well matched. These dummy cells also provide noise isolation between the cell array and the peripheral circuitry, reducing secondary noise that can be injected to the cell and helps in improving the yield. The dummy lines

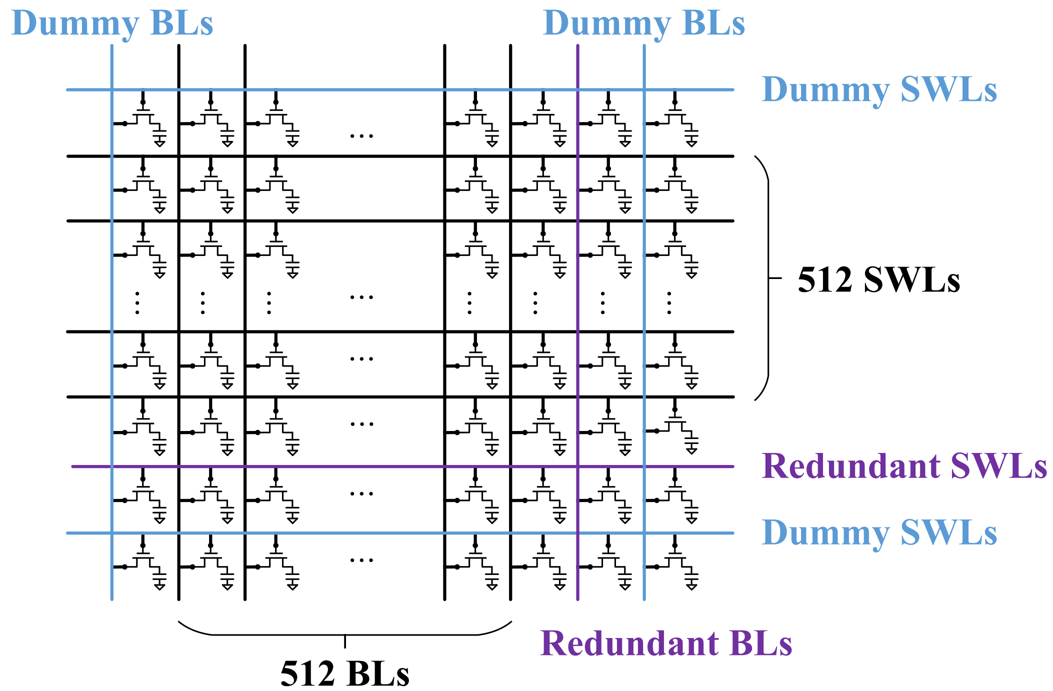


Figure 3.6: A typical MAT consists of (1) 512 logical wordlines and 512 logical bitlines, (2) dummy wordlines and bitlines, and (3) redundant wordlines and bitlines.

are added to the MAT boundaries next to the BLSA and the SWD or even another MAT.

In addition to the dummy cells, DRAM uses *redundant cells* to improve the overall die yield. Redundant lines improve yield by remapping faulty WL or BL to redundant lines that are tested to be functional. The remapping process, often called *repair*, is indispensable in DRAMs that are manufactured in large volumes. This repair operation is performed using special one time programmable fuses to store the mapping of the faulty and the repaired address. The fuses are programmed usually during wafer-level test due to the test cost. Many DRAMs now use an electric anti-fuse that can be ruptured using high electric field instead of the traditional metal fuse that had to be cut using a laser. So repair can be done post package and potentially even after it has been shipped [24, 22]. After repair, each die is unique which means for packages that contain multiple DRAMs, each DRAM in the package are all different.

There is one redundant MWL, which leads to 8 redundant SWL, and 4 redundant

CSL, which lead to 16 redundant BL, in a typical MAT shown in Figure 3.6. This is 1.56% and 3.13% of the normal WL and BL. While the number of redundant cells greatly exceeds the number of DRAM bit errors, the tight wiring in a DRAM limits how these redundant cells can be deployed. When a row is replaced, entire sub-wordlines forming that row are replaced as a unit. However, instead of limiting the repair of a faulty row to the redundant row on the same sub-array, the faulty row is remapped to a redundant row anywhere in the bank [48]. The column repair is both more restrictive and more flexible than row repair. It is more restrictive since the faulty column has to be remapped to a redundant column within the same MAT in order to meet the bandwidth constraints: every MAT in a sub-array has to provide data. However, it is more flexible since the replaced column does not have to run through all the sub-arrays in the bank. While all the sub-arrays share the same column decoder, because 1) only one row is selected in a bank at any given time, and 2) the sub-array being accessed is already known well before a column access, the remapped column address presented to the column decoder can depend on which sub-array is accessed. To reduce the complexity of the remapping logic, the bank is divided into a number of vertical segments (*i.e.* 4 or 8 in a bank with 32 sub-arrays) and treating the column repair independently for each segment [49]. For instance, if a faulty cell was observed in segment_0 but nowhere else, only the CSL on segment_0 remaps to the redundant CSL when a row in segment_0 is accessed, but not when a row in other segments are accessed.

Despite the importance of redundant cells to DRAM, few prior DRAM studies take this into account. Even the few [40, 50] that consider redundant cells propose to put repair information in serial presence detect (SPD) on the DIMM [51]. This proposal is not practical because repair information is proprietary and even if the manufacturers would release it, the repair information would require significant size memory to store; each die on a chip are repaired differently and there are multiple chips on a DIMM. Others propose to limit repair flexibility by forcing cells to be remapped in the same MAT. To ensure the same yield, more redundant cells will be needed in such designs.

LISA proposed by Chang *et al.* [50] is one of the examples that demonstrates how

important understanding redundant cells are. This work proposes to add links (transistors) to the cell array to move bulk page size data between sub-arrays within a bank. Since knowing the row address is crucial to identify the location of the source and the destination, they propose to put repaired row information to SPD. However, column redundancy is not considered at all which can cause failed bits if the destination row is at different column repair segment than the source. The source might have had a column repaired whereas the destination might not have or vice versa. Thus, LISA can potentially access columns that do not store the data resulting in failed bits.

Chapter 4

DRAM Modeling Framework

The bifurcation of computing into large-scale data centers and battery operated mobile devices made optimizing the performance and the energy consumption of DRAM even more important. However, exploring this design optimization space is difficult because as mentioned in the previous chapter, the cost constraints leads to specialized structures and design constraints that are not widely known outside of the DRAM manufacturers.

This chapter introduces DRAM Design Space Exploration (DramDSE), an open-source DRAM modeling framework, that is built with the constraints posed by modern DRAM in mind. Two state-of-the-art DRAMs, Low Power DDR4 (LPDDR4) and High Bandwidth Memory (HBM), are modeled with DramDSE. The area and power consumption results of those DRAMs are compared with measured values of mass-produced LPDDR4 [19] and HBM [52] manufactured by SK Hynix to ensure the correctness of our model. Then, the detailed power breakdown of each DRAM operations generated by DramDSE is used to analyze the energy efficiency of modern DRAM in depth. DramDSE's ease of use is also shown by exploring different DRAM architectures using DramDSE and comparing the area and power consumption between them. Finally, the effectiveness of DramDSE in computer architecture studies is demonstrated by re-evaluating the overheads of recent DRAM work.

4.1 Background

The exponential growth of data resulted in high demands for large capacity high performance DRAM. Unfortunately, the power consumption of DRAM has not been scaling at the same rate the data has been growing. Now, the power consumption of DRAM is a major bottleneck in building efficient computer systems resulting in traditional DDR DRAM, such as DDR3 and DDR4, no longer a viable option to meet the thermal and power budget of the system. This ultimately led to the recent industry developments of specialized DRAMs that focus on low power (LPDDR) [5, 23, 24] and high bandwidth (HBM) [25].

LPDDR4 is one of the newest generation of the LPDDR family that has been widely adopted in battery operated devices, such as laptops and smartphones. Many innovative techniques improving the energy efficiency of the predecessor (LPDDR3) have been added in LPDDR4, including lower power supply voltage, reduced page size, and small swing I/O signaling.LPDDR4 has two independent channels on a die and each column operation is a 16n-prefetch [20] that fetches 256 bits of data to meet the high pin data rate requirements of modern computer systems.

HBM is a 3D Stacked (3DS) DRAM that provides high memory bandwidth using Through-Silicon Via (TSV) technology [53]. Like LPDDR4, HBM also has two channels on a die where each column operation fetches 256 bits of data. However, HBM has many more I/O pins (256) compared to LPDDR4 (32), allowing the data rate to be reduced to 1 Gbps/pin instead of 3.2 Gbps/pin. Up to 4 stack of HBM can be stacked using TSVs to provide 8 channels of DRAM with 1,024 I/O pins and a total bandwidth of 128 GB/s [52].

Both LPDDR4 and HBM fetch 256 bits of data out of a 2 KB page size row in each column operation. To reduce the number of data wires routed in each bank, 128 bits are prefetched from two separate sub-banks [52, 19] which is different from DDR4 DRAM that prefetches 64 bits out from a single sub-bank [18]. The two sub-banks that are accessed simultaneously are located on the left and right halves of a channel, and the corresponding I/O pads or TSVs are also split accordingly to minimize the total wire length of the GIO wires that route to the pads or TSVs. This

unique architecture of LPDDR4 and HBM enables high bandwidth at less power consumption and silicon die area (cost).

In addition to the industry's effort, there are many proposals [40, 54, 50, 46] that customize the internal circuitry of the DRAM to make DRAM a better fit for modern computer systems. Despite substantial research effort, few of these ideas are relevant to the DRAM being manufactured because constraints posed by modern DRAMs are often not understood or misinterpreted. When the constraints are correctly considered, most of the previous work cause unexpected overheads that diminish the benefits or in some cases even change the DRAM's functionality. While understanding the low-level process and circuit constraints of DRAM is critical to DRAM studies, it is surprisingly difficult to find as it is often considered proprietary information.

Many existing DRAM simulators are capable of evaluating the energy [55] or power [56] consumption of DRAM for a given workload and memory configuration when IDD values, power supply voltage, and timing constraints are provided as inputs to the simulator. For standard commodity DRAM, the power supply voltage and timing constraints that are needed to find energy consumption of DRAM are well specified values. IDD values, on the other hand, vary depending on the transistor technology and the design of the DRAM itself. Hence, IDD values are generally not provided with the product specification datasheet that are open to public. This lack of public data is often the case for newer devices, such as LPDDR4 and HBM. To address this problem, and to allow researchers to analyze their new designs, prior research built DRAM modeling frameworks that generate the IDD values.

4.2 Prior DRAM Modeling Work

CACTI-D by Thoziyoor *et al.* [32] is an extended version of the CACTI [57], which enables modeling commodity DRAMs on top of SRAMs that CACTI originally supported. CACTI-D uses a detailed wire delay and power model to provide the area, performance, and energy consumption of a DRAM. The technology parameters needed to generate those numbers were estimated using the values from the ITRS, which are a good reference to predict the future process technology, not the technology used for

commodity DRAMs. To simplify the model, CACTI-D assumes all of the global wires to be routed in an H-tree structure, and while these assumptions make the CACTI-D model easy to use, these assumptions limit the types of DRAM that can be modelled, and are not representative of today's DRAMs.

DRAMSpec by Naji *et al.* [34] takes a different approach. Its high-level abstracted model provides IDD values and area as well as timings for DRAM. The high-level abstraction makes this model easy to use even if you are not an expert. However, it severely reduces the flexibility in exploring the design space of DRAM because it only has one fixed structure for the bank, similar to CACTI-D.

Rambus Power Model by Vogelsang [33] provides a flexible model that can analyze DRAM area and power consumption of different architectures. It was constructed to address the modeling limitation of CACTI-D which had a fixed architecture for DRAM. The Rambus Power Model uses a detailed area model, a capacitance estimation approach based on wire length, and provides a very large number of parameters with which the user can adjust their design. While the large number of parameters improve flexibility and accuracy, they also make this model difficult for non-experts to use. Even with this large number of parameters, it uses a restricted architectural template for banks that does not match today's high-bandwidth DRAMs such as LPDDR4 and HBM. We address all these issues in *DramDSE*, which is described in the next section.

4.3 DramDSE: DRAM Design Space Exploration

We build upon prior DRAM modeling work to create a hierarchical DRAM modeling framework that is flexible enough to model any JEDEC standard DRAM known to this date. While our new modeling framework has a large number of parameters, it also includes intelligent methods of estimating most parameters if they are not provided. At the highest level of abstraction, all commodity DRAMs look fairly similar. A DRAM has multiple banks, where each bank is attached to a set of peripheral circuits connecting the memory banks to the chip pads or TSVs. The diversity of design increases as we push into these structures: bank organization

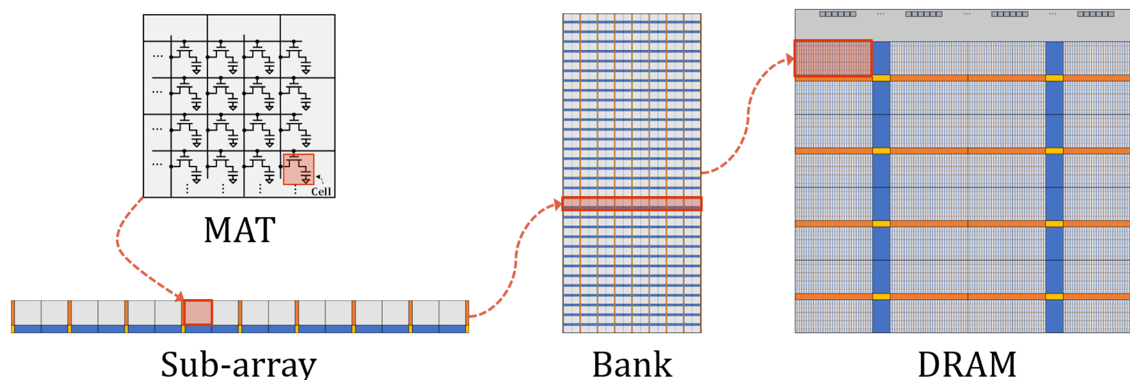


Figure 4.1: The level of abstractions used in our DRAM modeling framework. Although it is not explicitly shown, *sub-DRAM* consists of bank(s) and row/column decoder(s). In this example, sub-DRAM is formed with four banks and two row/column decoders that are shared with adjacent banks.

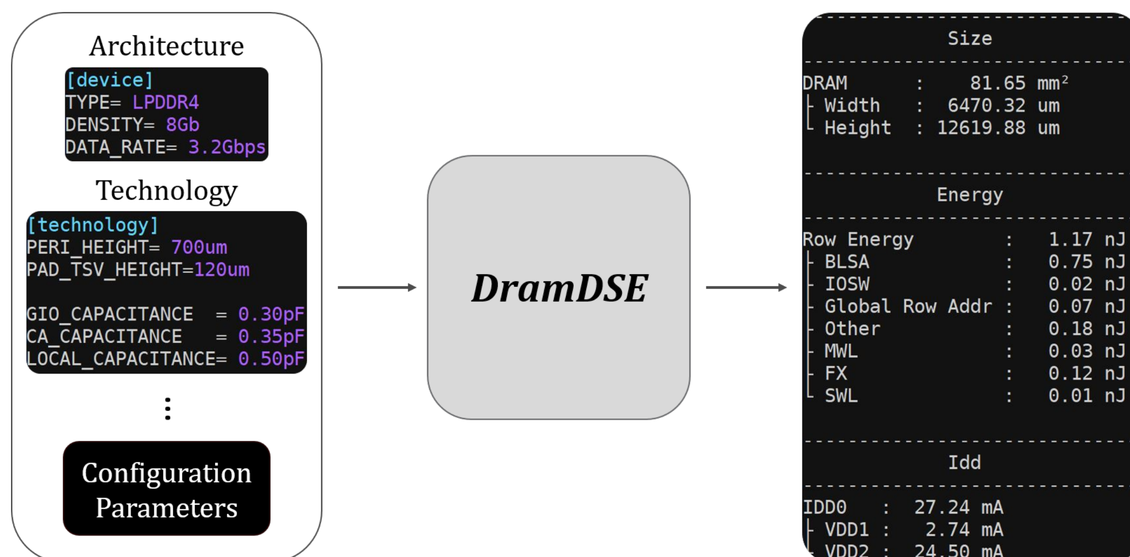


Figure 4.2: DramDSE models DRAM using the configuration parameters that define architecture and technology as inputs and reports the area, energy breakdown and IDD values. This work is a collaboration work with SK Hynix, one of the three major DRAM manufacturing company.

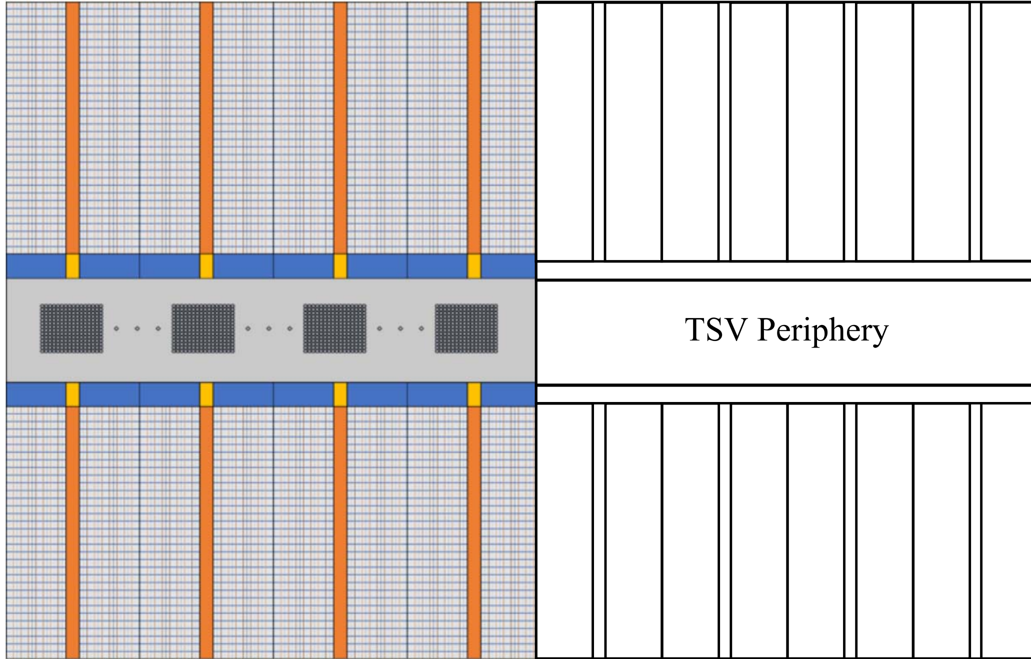


Figure 4.3: Side-by-side comparison of the DramDSE's visualization feature result (left) and the trace of the actual die (right) for 20 nm 8Gb HBM Gen2 [4].

and pad location will differ in different types of devices, and pushing deeper, things like MAT organization and redundancy can differ by manufacturer or technology generation. To deal with this diversity and still provide a framework that is easy to use and understand, we characterize a DRAM using 5 levels of hierarchy: DRAM device, sub-DRAM, bank, sub-array and MAT as shown in Figure 4.1. Each level of hierarchy encapsulates the details of the levels below it, providing a higher level abstraction to the user. At the same time, for users who want complete control, we provide access as deep as the MAT, which is the basic building block of all DRAMs. The result is a framework that can model any commodity DRAM, and we use it to create models for two state-of-the-art DRAMs, LPDDR4 and HBM.

We also worked with current and formal DRAM designers to include constraints posed by modern DRAMs to a level at which no other existing DRAM modeling work have. An executable version of the modeling framework we built was created using C++11 and we open-sourced it under the name *DramDSE* [58], short for DRAM Design Space Exploration. DramDSE reports detailed energy breakdown as well as

```
1  [device]
2  TYPE = LPDDR4
3  DENSITY = 8Gb
4  DATA_RATE = 3.2Gbps # Gbps or Mbps
5
6  [external_voltage]
7  VDD2  = 1.1V
8  VDD1  = 1.8V
9  VDDCA = 1.1V
10 VDDQ  = 1.1V
11
12 # internal overrides external
13 [internal_voltage]
14 VPERI = 1.1V
15 VCORE = 0.95V
16 VDDY  = 1.5V, EXT_VDD = VDD1
17 VPP   = 2.7V, EXT_VDD = VDD1, EFF = 33%
```

Listing 4.1: Simple Configuration File

IDD values for each DRAM operation based on the configuration parameters given by the user, as illustrated in Figure 4.2. Visualization of the DRAM modeled by DramDSE is also provided using OpenCV library [59]. Figure 4.3 shows an example of the visualization feature by comparing DramDSE's output and a trace from the actual die side-by-side. It should be emphasized that DramDSE estimates area and energy but not performance. The limited metal layers of DRAM make performance sensitive to voltage drops on the power distribution network, which is difficult to model without the entire circuit and transistor technology details.

4.4 Configuration Parameters

Throughout this section we will refer to the configuration parameters that are specified in one or more files. The configuration parameters are organized into different abstraction levels, making it easy for a user to experiment with different DRAM organizations. The user need not specify all or even most of the parameters. The modeling framework will estimate the missing parameters from those provided. This parameter estimation feature allows the framework to be used by researchers having

different levels of DRAM expertise. A simple example template for the configuration file is listed in Listing 4.1. The headers placed in '[ ]' brackets are used to group parameters that have common functions. Comments are also allowed following a '#' symbol. The complete list of predefined headers and parameters are described in the following sections.

4.4.1 Architecture Parameters

The list of architecture parameters are listed in Table 4.1. The headers of the architecture parameters match with the hierarchy level shown in Figure 4.1 as the architecture parameters determine the details of each level of the DRAM hierarchical structure. TYPE, DENSITY, and DATA_RATE parameters under the [device] header are the only architecture parameters that are required for DramDSE to construct a DRAM. The rest can be omitted and if so, some of the parameters will be assigned with predefined values as stated in Table 4.1 or will be estimated using other parameters as shown in Listing 4.2 . When DramDSE estimates those parameters, it tries to build a DRAM with the most commonly observed architecture in practice. For example, a MAT will have 512 wordlines by 512 bitlines and number of sub-drams in the x and y direction will be chosen to make the chip as square shape as possible to potentially maximize the net die. If the decisions made by DramDSE do not comply with the DRAM of interest, the user can specify some of the parameters that were previously not defined to the best of their knowledge. The parameters that are still unspecified will be re-adjusted based on the new set of parameters given by the user making DramDSE easy to explore different DRAM architectures.

Header	Parameter Name	Default Value	Optional
[device]	TYPE	N/A	N
	DENSITY	N/A	N
	DATA_RATE	N/A	N
	IO_WIDTH	-	Y
	PAGE_SIZE	-	Y
	PREFETCH	-	Y
	NUM_BANK	-	Y
[sub-dram]	NUM_X_SUBDRAM	-	Y
	NUM_Y_SUBDRAM	-	Y
	TRANSPOSE_BANK	FALSE	Y
	SHARED_ROW_DEC	TRUE	Y
	SHARED_COL_DEC	FALSE	Y
[bank]	NUM_SUBBANK	-	Y
	NUM_SUBARRAY	-	Y
	SWL_PER_MWL	8	Y
	BL_PER_CSL	-	Y
[sub-array]	PAGE_SIZE	-	Y
	PREFETCH	-	Y
	NUM_SWD	-	Y
	NUM_MAT	-	Y
	CRR	FALSE	Y
	HALF_PAGE	FALSE	Y
[mat]	NUM_SWL	512	Y
	NUM_BL	512	Y
	NUM_REDUNDANT_SWL	-	Y
	NUM_REDUNDANT_BL	-	Y
	NUM_DUMMY_SWL	4	Y
	NUM_DUMMY_BL	4	Y

Table 4.1: Complete list of architecture parameters. An example architecture definition is listed in Listing B.1 of the Appendix.

```
/* sub-dram */
// NUM_X_SUBDRAM
num_x_subdram = device_prefetch/subarray_prefetch;
if(num_subdram/num_x_subdram > 2*num_x_subdram) {
  num_x_subdram = 2*num_x_subdram;
}
// NUM_Y_SUBDRAM
num_y_subdram = num_subdram/num_x_subdram;

/* bank */
// NUM_SUBBANK
num_subbank = (num_rows/num_subbank_rows) * (device_page_size/
subarray_page_size);
// NUM_SUBARRAY
num_subarray = (int)std::ceil((float)num_subbank_rows / num_swl);
// BL_PER_CSL
bl_per_csl = subarray_prefetch / num_mat;

if(half_page) {
  bl_per_csl = bl_per_csl * 2;
}

/* sub-array */
// PAGE_SIZE
subarray_page_size = std::min(8192, device_page_size);
// PREFETCH
subarray_prefetch = device_prefetch * (float)(subarray_page_size)/
device_page_size;
// NUM_SWD
num_swd = num_mat + 1;
// NUM_MAT
num_mat = subarray_page_size / num_bl;

/* mat */
// NUM_REDUNDANT_SWL
num_redundant_swl = 0;
if((num_mwl != 0) && (num_mwl % 64 == 0)) {
  num_redundant_swl = (num_mwl/64) * swl_per_mwl;
}
// NUM_REDUNDANT_BL
num_redundant_csl = std::max(4, (int)std::ceil(32*num_bl/512/
bl_per_csl));
num_redundant_bl  = num_redundant_csl * bl_per_csl;
```

Listing 4.2: Architecture Parameter Estimation Method

4.4.2 Technology Parameters

The list of technology parameters is shown in Table 4.2 and they are used to estimate the area, detailed energy breakdown, and the IDD values of the DRAM modeled using the architecture parameters listed in previous section.

Header	Parameter Name	Default Value	Optional
[external_voltage]	VDD2	1.2V	N
	VDD1	1.8V	Y
	VPP	2.5V	Y
[internal_voltage]	VPERI	VDD2	Y
	VCORE	VDD2	Y
	VPP	VDD2	Y
	VDDY	VDD2	Y
	EXT_VDD	VDD2	Y
	EFF	100%	Y
[idd]	IDD2N_VDD1	0	Y
	IDD2N_VDD2	0	Y
	IDD3N_VDD1	0	Y
	IDD3N_VDD2	0	Y
	tCK	-	Y
	tRASmin	-	Y
	tRP	-	Y
	tCCD	-	Y
	tRFC	-	Y
	PERI_HEIGHT	-	Y
	CA_PERI_HEIGHT	0	Y
	DQ_PERI_HEIGHT	0	Y
	PAD_TSV_HEIGHT	120/700 μm	Y
	SUBDRAM_X_SPACING	0	Y

SUBDRAM_Y_SPACING	0	Y
ROW_DEC_WIDTH	-	Y
COL_DEC_WIDTH	-	Y
SWD_WIDTH	5 um	Y
BLSA_HEIGHT	11 um	Y
FEATURE_SIZE	26 nm	Y
PITCH_SWL	-	Y
PITCH_BL	-	Y
PITCH_LOCAL	-	Y
CA_CAPACITANCE	0.40 pF	Y
LOCAL_CAPACITANCE	-	Y
CSL_CAPACITANCE	0.40 pF	Y
GIO_CAPACITANCE	0.35 pF	Y
LIO_CAPACITANCE	-	Y
SIO_CAPACITANCE	-	Y
SWL_CAPACITANCE	-	Y
MWL_CAPACITANCE	-	Y
FX_CAPACITANCE	-	Y
EQ_CAPACITANCE	-	Y
SAN_CAPACITANCE	-	Y
SAP1_CAPACITANCE	-	Y
SAP2_CAPACITANCE	-	Y
IOSW_CAPACITANCE	-	Y
CELL_CAPACITANCE	10 fF	Y
BLSA_CAPACITANCE	40 fF	Y

Table 4.2: Complete list of technology parameters. An example technology configuration definition is listed in Listing B.2 of the Appendix.

The dynamic energy consumed in DRAM is the energy used to charge the capacitance of the transistors and the wires. Unlike most logic devices, DRAM often

regulate external voltages so the internal voltage swing is not full rail. Thus, you need to consider the voltage swing on the capacitor as well as the supply voltage when calculating DRAM's energy. DramDSE provides up to four different internal power supply voltages, V_{PERI}, V_{CORE}, V_{DDY}, and V_{PP}. By default, V_{PERI} is used to estimate the energy consumption of clock tree, global row/column address, GIO, and local wires on the periphery circuitry. V_{CORE} is used to estimate the energy dissipated to operate the bitline sense amplifier (BLSA) and V_{DDY} is used to estimate the energy consumed to enable CSLs for column operations. Finally, V_{PP} is the wordline boost voltage that is used to estimate the energy consumed to enable a wordline during row operations. Since this voltage is above the external power supply voltages, to calculate the power correctly, one needs to provide the energy efficiency of the charge-pump voltage boost circuit denoted as EFF in Table 4.2.

The dynamic energy E consumed at V_{ext} power supply to charge the capacitance C to V_{int} with a switching activity α is,

$$E = \alpha \cdot C \cdot V_{int} \cdot V_{ext} \tag{4.1}$$

DramDSE provides the energy breakdown for each DRAM operations using 16 different capacitance values listed in the bottom portion of Table 4.2. Among those capacitance values, only six (CA, LOCAL, CSL, GIO, SWL, and CELL) are required by DramDSE and the rest are estimated if they are not specified; using those six capacitance values that must be defined by the user. The capacitance values are the load capacitance which is a lumped sum of wire and transistor capacitance. On the configuration file, capacitance values for CA, LOCAL and GIO are represented in pF per 1,000 μm while capacitance of CSL, LIO and SIO are represented in pF per sub-array, and SWL, MWL, FX, EQ, SAN, SAP1, SAP2 and IOSW are represented in pF per MAT.

Once the energy consumption is found, the average IDD values for each power supply V_{ext} that is measured for the event duration of t can be estimated using the following equation (Equation 4.2).

$$IDD = \frac{E}{V_{ext} \cdot t} = \sum_{i=1}^{n} \frac{\alpha \cdot C_i \cdot V_{int_i}}{t} \tag{4.2}$$

The t of Equation 4.2 is already known, because it can be obtained straight from the specification datasheet and the respective t for each IDD is automatically assigned by the model based on the device type, data rate and density. Parameters associated with t are tRASmin, tRP, tCCD, and tRFC in Table 4.2.

4.5 Validation of DramDSE

To check the correctness of the model used in DramDSE, we compared area and power results generated by DramDSE against measured results of mass-produced 2y[1] nm 8Gb LPDDR4 [19] and 2x nm 2Gb HBM [52] parts from SK Hynix.

4.5.1 Low Power DDR4 (LPDDR4)

Figure 4.4a shows the hierarchical architecture of the 2y nm 8Gb LPDDR4 that was modeled using DramDSE. Both the row and the column decoders are shared between adjacent sub-banks resulting in a sub-DRAM with four sub-banks. It also has a transposed bank where the sub-wordline direction is perpendicular to the pad arrays. There are four sub-banks per bank and two sub-banks, one from Sub-DRAMxL and the other from Sub-DRAMxR, are activated simultaneously for each access. Each activated sub-bank transfers 128 bits of data. The size of each sub-bank is 128 Mbit, having 32 sub-arrays and a dummy stacked on top of each other. The actual DRAM device has two of these blocks shown in Figure 4.4a, where the second block is mirrored about the x axis to drive the second channel pads at the bottom of the die. The 90° counterclockwise rotated view of the whole 2y nm 8Gb LPDDR4 is shown in Figure 4.4b. To visualize how accurately DramDSE modeled the DRAM, the visualization result of DramDSE and the half die photo of the actual part are shown

[1]DRAM process technology are specified differently from logic technology. Since they are continually improving the process, DRAM technologies are marked by generations at a given lithography range. 2y is a 2nd generation technology in the 20-30nm range.

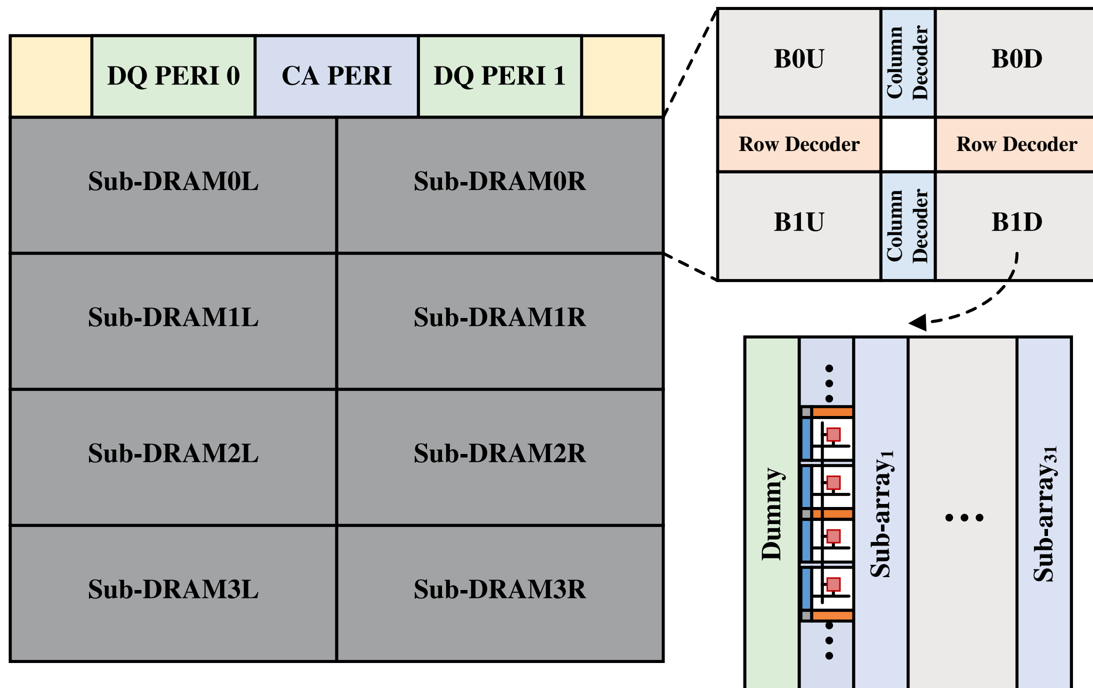
Symbol	VDD2	VDD1	Power
IDD0	0.10% (0.08%)	5.72% (0.98%)	1.06%
IDD4R	2.23% (2.03%)	0.40% (0.03%)	2.06%
IDD4W	1.65% (1.50%)	0.40% (0.03%)	1.53%
IDD5	0.67% (0.50%)	0.85% (0.22%)	0.72%

Table 4.3: IDD and power % error from measured values compared to the results of DramDSE. Parenthesis represents its contribution to the total power.

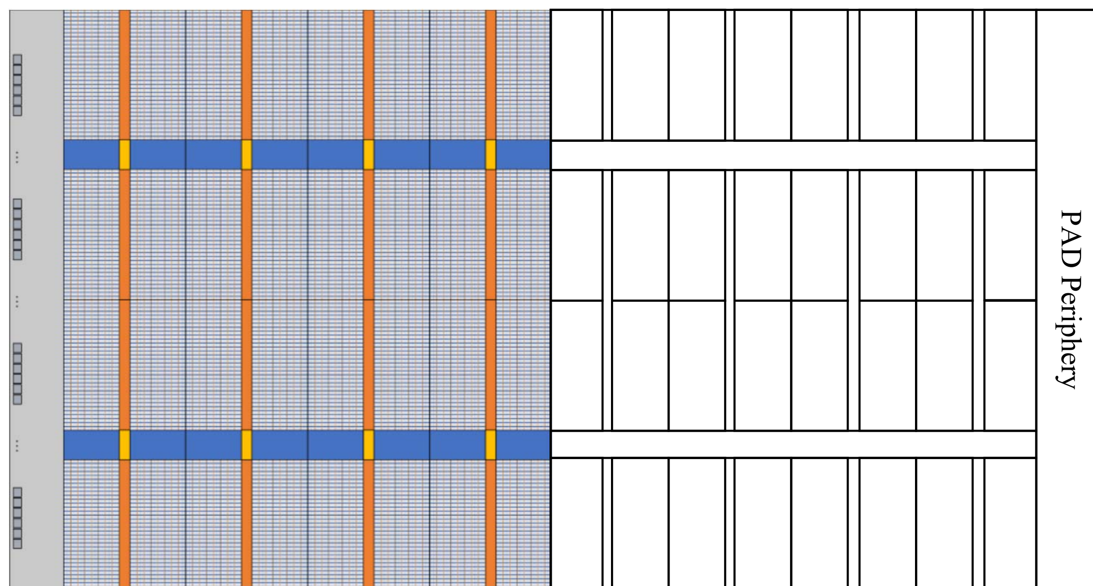
side-by-side in Figure 4.4b. The die area error between the resulting DRAM built with our model and the actual part was only 1.13%. We believe the error comes from the scribe lanes which were neglected in this analysis.

Table 4.3 shows the absolute value of the error of each IDD for both VDD2 and VDD1 power supply as well as the total power error when the results from our model were compared with the measured values of the device.[2] The parenthesis next to each IDD error value denotes its contribution to the total power error. Because refresh is just an automated activate and precharge to many rows in many banks as discussed in Section 2.2, DramDSE calculates IDD5 using IDD0, IDD2N and IDD3N.

The % error is well within 5% except for the VDD1 power supply of IDD0 which showed 5.72% error. However, the contribution of VDD1 power supply to the total power is relatively small compared to VDD2 resulting in a total row power error of only 1%. Another interesting point to note is that the error of the VDD1 power supply is only 0.85% in IDD5; much less than that of IDD0. In fact, there is a constant 0.2 mA error for both IDD0 and IDD5 which we conjecture that the error is caused by a reduction in standby current during IDD0 and IDD5 condition compared to the IDD2N and IDD3N condition. This can be due to power gating of the wordline boot voltage (V_{PP}) that connects to the deselected row decoders.

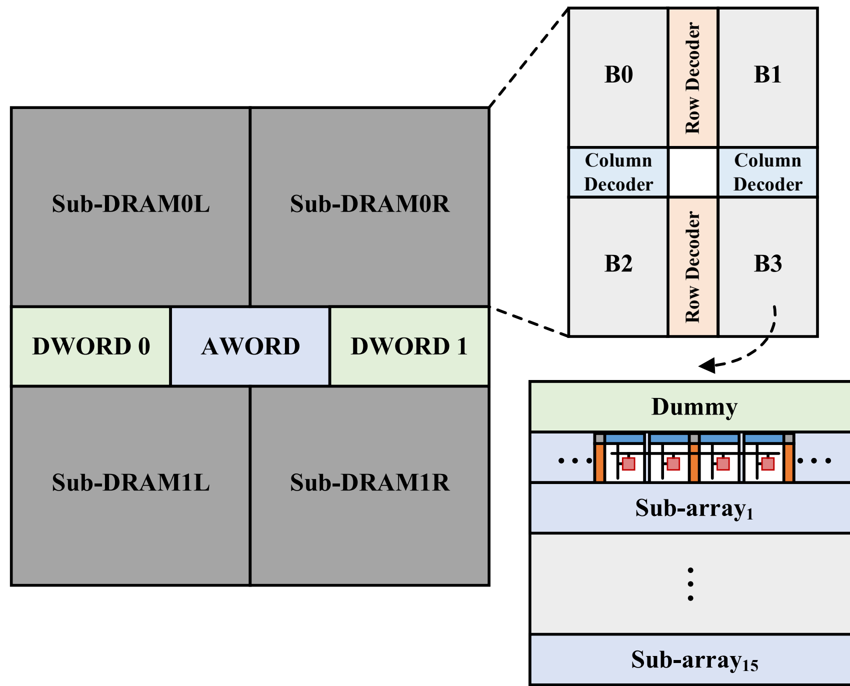


(a) Architecture breakdown of a single channel.

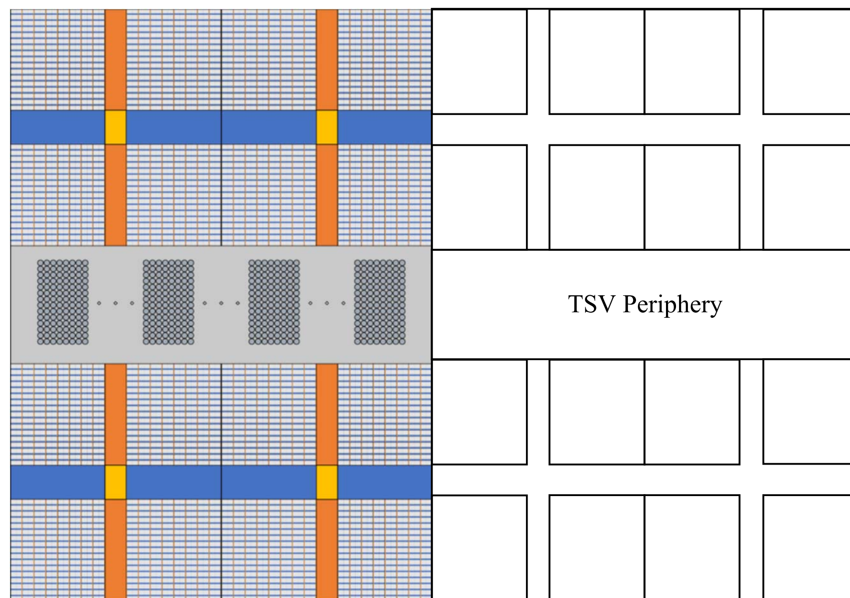


(b) Side-by-side comparison of the DramDSE's visualization feature result (left) and the trace of the actual die (right) [19].

Figure 4.4: Modeling result of 2y nm 8Gb LPDDR4 using DramDSE.



(a) Architecture breakdown of a single channel.



(b) Side-by-side comparison of the DramDSE's visualization feature result (left) and the trace of the actual die (right) [52].

Figure 4.5: Modeling result of 2x nm 2Gb HBM Gen1 using DramDSE.

Symbol	VDD2	VPP	Power
IDD0	6.18% (5.61%)	5.82% (0.54%)	6.15%
IDD4R	0.30% (0.30%)	19.7% (0.06%)	0.36%
IDD4W	0.39% (0.39%)	19.7% (0.07%)	0.46%
IDD5	4.23% (3.61%)	2.58% (0.38%)	3.99%

Table 4.4: IDD and power % error from measured values compared to the results of DramDSE. Parenthesis represents its contribution to total power.

4.5.2 High Bandwidth Memory (HBM)

To further validate the correctness of DramDSE, we modeled 2x nm 2Gb HBM [52]. The architecture detail of the HBM we modeled is shown in Figure 4.5a which shows only one channel of the HBM die. Similar to the sub-DRAM of 2y nm 8Gb LPDDR4 we modeled in the previous section, this HBM also has the row and column decoders shared between adjacent sub-banks but without the bank transposed. The bank is split into two sub-banks of 64 Mbit each insetad of 128 Mbit, where one is located again on Sub-DRAMxL and the other on Sub-DRAMxR to transfer a total of 256 bits of I/O divided evenly in half to left and right, just like the LPDDR4. The left half of Figure 4.5b visualizes the 2x nm 2Gb HBM modeled using DramDSE and the half die photo of the actual part is placed next to it for comparison. The resulting die area of the left half of Figure 4.5b was off by 4.7% compared to the measured die area shown on the right half of Figure 4.5b. The direct access (DA) ports that are on the edge of each channel [52] results in the additional die area error aside from the scribe lanes. It should be noted that the area of 2x nm 2Gb HBM is only 2.5× smaller than that of 2y nm 8Gb LPDDR4 despite the density being 4× smaller. This is mainly due to the thousands of TSVs on the center stripe to stack up to four dies in the same package.

Table 4.4 shows the absolute value of the error of each IDD and power when the results from the 2x nm 2Gb HBM modeled using DramDSE were compared with the measured values from devices (again with unknown process skew). Instead of coming up with a new set of technology parameters, capacitance values, we scaled the

[2] We were not given the process skew of the measured devices.

technology parameters of the 2y nm process technology that we already had hands on to model the LPDDR4 instead of coming up with a new set of technology parameters.

Percentage error is within 10% except for the VPP of IDD4R and IDD4W which has exceedingly high error of 19.7%. DramDSE assumes that the VPP component for IDD4 is equivalent to the VPP component of IDD3N as IDD4 is a column operation and there should be no additional power consumed on the VPP power supply other than the wordline leakage that is captured in IDD3N. However, the measured results had a smaller IDD4 VPP component than the IDD3N VPP component. We conjecture that power gating of unused row decoders is performed during IDD4 condition to reduce the overall leakage of the VPP power supply. Power gating is especially effective in this power domain because GIDL is the main contribution source due to the high supply voltage. We are planning to investigate this further by measuring the VPP leakages at colder temperatures since GIDL is relatively insensitive to temperature. Nevertheless, VDD2 is the major source of power consumption during IDD4 and the large error of IDD4 VPP component does not cause an appreciable error in the overall column power consumption as can be seen from Table 4.4. For IDD4R, we enabled the local sense amplifier (LSA) [47] scheme and assumed the Δ of LIO/B that is sensed at the column decoders to be 600mV. We think the LSA scheme is a must in HBM where the core frequency as well as other timing parameters are relatively short compared to other commodity DRAMs as discussed in Section 2.3.1 and shown in Table 2.1.

4.6 Power Breakdown Analysis

One of the outputs that DramDSE generates is the detailed breakdown of each power category described in Section 2.4. Figure 4.6a shows the breakdown of row power for the 2y nm 8Gb LPDDR4. Most of the power (47%) for the row operation is consumed by the BLSA which charge and discharge a page size of bitlines and the cells. This is one of the reason why row buffer overfetch [60] is one of the most popular subject in DRAM optimization studies. The WL component sums the entire power consumed to select a wordline which includes driving/toggling power of MWL, FX, and SWL.

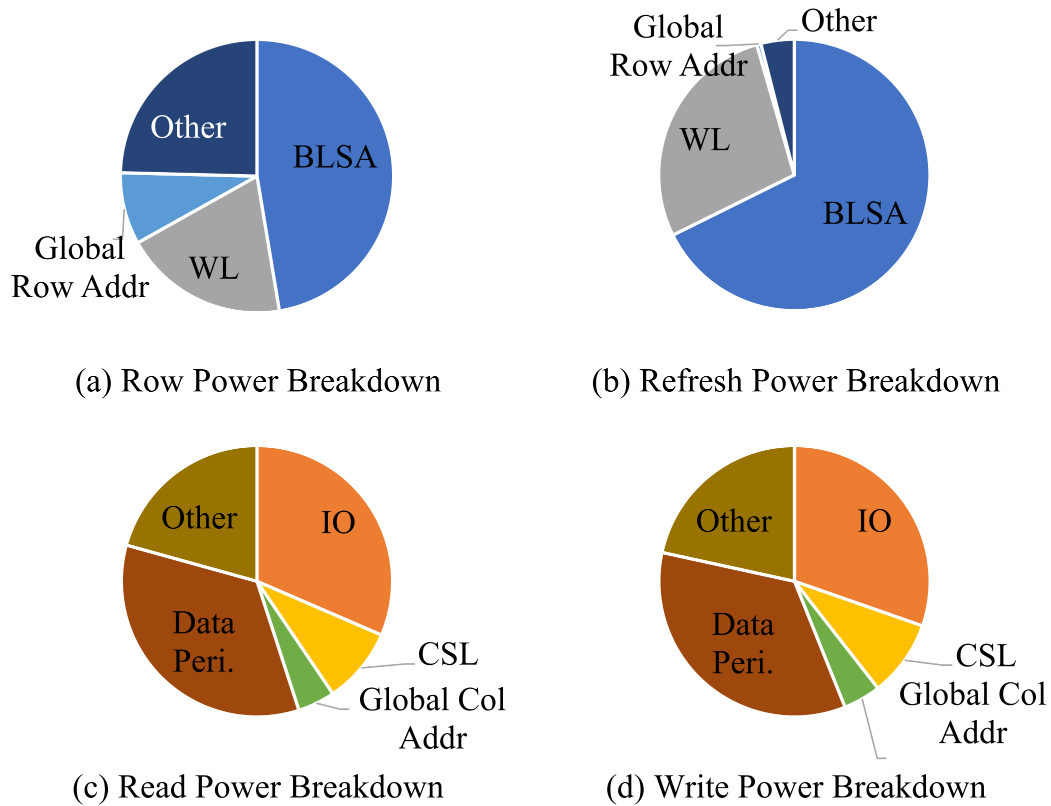


Figure 4.6: 2y nm 8Gb LPDDR4 power breakdown for each categories (a) row, (b) refresh, (c) read and (d) write.

The high boosted power supply voltage (V_{PP}) is used on all of these signals to ensure that the access transistors with high threshold voltage is fully on and results in next highest power consuming component (20%) of the row operation. The wire length of the global row address sent from the pad to the row decoder of a bank is quite long in LPDDR DRAMs because the pads are located on the edge of the die instead of the center [61, 62, 19]. Simply driving and toggling the row address wires from the pads to the banks cost 8% of the entire row power. Selecting IOSW for the activated sub-array to connect SIO to LIO wires, as discussed in Section 3.2.1, and driving/toggling control wires needed for the row operation contribute the most on the component denoted as 'other' that consumes a total of 25% row power.

Since the refresh operation is essentially a number of row operations done in parallel to multiple banks, the power breakdown is quite similar to that of the row with

BLSA being the most power hungry component with 68% as shown in Figure 4.6b. But unlike the row operation, the power consumed by the global row address and control signals are suppressed significantly making the sum of the two components only 4% instead of 33% that we saw from the row power breakdown. In the case of 8Gb LPDDR4, a total of 32 rows, 4 rows on each bank, are refreshed during a refresh operation. Because so many rows are refreshed, piled refresh [63] is used, which divides the rows to be refreshed to different piles and spaces out refreshes of each pile from each other, to reduce peak power consumption. Single row address can be used to refresh all of the rows that are to be refreshed during a refresh cycle and the control wires that are to be toggled can be reduced significantly depending on the number of piles. As a result, global row address energy and other components become insignificant for refresh. Considering that dynamic power due to refresh operation is dominant during self-refresh [64], reducing BLSA and WL energy consumption during self-refresh is an effective method to reduce power for LPDDR DRAMs that are known to be put in self-refresh mode frequently for long periods of time.

Figure 4.6c and d show the breakdown of read and write power respectively. In this breakdown, we merged the power consumed by SIO, LIO, and GIO wires into a single 'IO' component. IO is one of the major power consuming component aside from the data periphery, consuming roughly 31% of both read and write power. DramDSE assumes local sense amplifiers [47] to speed-up read times and no precharge of SIO and LIO wires for consecutive writes to reduce power consumption. Data serialization/de-serialization for reads/writes as well as data alignments for both column operations are the main activity done in the data periphery, the most power consuming component. These circuits run at the maximum operating frequency of 3.2Gbps in our LPDDR4 example resulting in high power consumption of 35% column power. Similar to the row energy breakdown, other component for column operations is mostly due to the control wires and consumes roughly 20% of the power consumed during column operation. CSL power is amortized by the power consumed on the IO wires since 8 IO wires are selected using a single CSL. Still, CSL consumes 10% of the column power because a separate power supply voltage V_{DDY} that is higher than V_{CORE} is used to give high core frequency. Global column address consumes only 4% of the

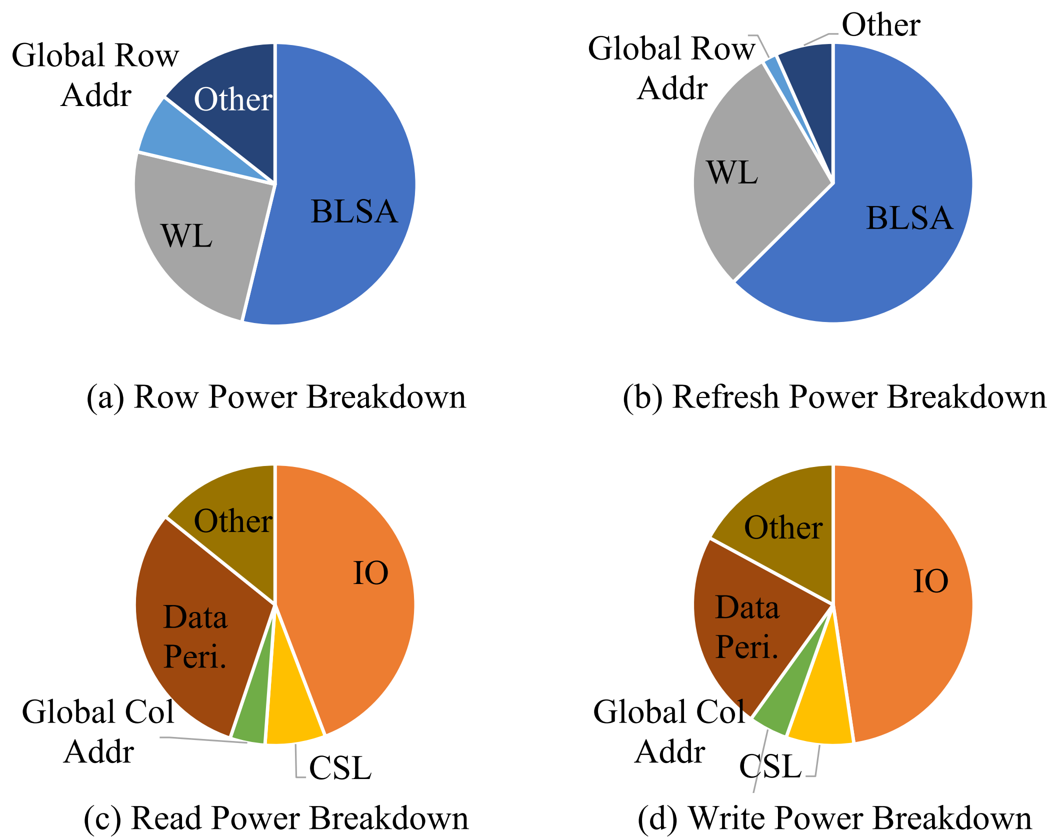


Figure 4.7: 2x nm 2Gb HBM power breakdown for each categories (a) row, (b) refresh, (c) read and (d) write.

column power.

With the power breakdown analysis of LPDDR4 in mind, we look at HBM power breakdown shown in Figure 4.7 to better understand the common trend in DRAM power consumption as well as the difference between low power (LPDDR4) and high performance (HBM) DRAMs. The major power consuming component for each operation is similar for both LPDDR4 and HBM. BLSA consumes the most power for row and refresh, and IO is one of the most power consuming component for column operations. However, the BLSA and WL portion of the share for row is slightly larger whereas other component is smaller on HBM compared to LPDDR4. This is because power consumption in DRAM is mostly due to the wires and the dimension of 2Gb HBM is smaller than that of 8Gb LPDDR4. The area of 2x nm 2Gb HBM is only 40%

Processor	4-core, 2.4GHz, Nehalem architecture [65]
LLC	shared 8MB, 16-way LRU, 32B cacheline
DRAM Controller	FR-FCFS, relaxed close-page policy, 64-entry read/write queue
DRAM	LPDDR4: single die, 2y nm, 8Gb/die HBM: 4 Hi-stack, 2x nm, 2Gb/die

Table 4.5: System Configuration

of 2y nm 8Gb LPDDR4. Hence, the lengths of the wires, especially the global ones, are much shorter in HBM. On top of this, BLSA and WL energy consumption being roughly the same for both devices due to the same page size results in larger share for BLSA and WL and smaller share for other. Similar to the row, column operations of HBM also show less percentage for other component but more IO and less data periphery percentage compared to LPDDR4. To clarify, HBM consumes 15% less IO energy than LPDDR4 due to smaller dimension despite the older technology node and higher external power supply voltage used in HBM. It is the data periphery power consumption shrinking more than IO that results in this breakdown characteristics. HBM operates at a much lower frequency per IO pin than LPDDR4, 1000Mbps vs. 3200Mbps, allowing much smaller sized drivers that consume less power sufficient enough to do the job on the data periphery. This significantly reduces data periphery energy consumption of HBM compared to LPDDR4 hence, boosting the share of IO more than that of LPDDR4.

4.7 System Simulation Methodology

We evaluate the energy consumption of 2y nm 8Gb LPDDR4 and 2x nm 2Gb HBM in a typical computer system with a quad-core processor running various workloads. The effect of larger capacity and different MAT architectures are analyzed to demonstrate how useful DramDSE can be to computer architecture studies.

The workloads used for evaluations are composed with a mix of medium to high memory intensity applications of SPEC CPU2006 benchmark suite [66] and the setup

Symbol	Benchmark Mix	MPKI
mix1	xalancbmk-wrf-zeusmp-gcc	M-M-M-M
mix2	astar-dealII-sphinx3-soplex	M-M-M-H
mix3	bzip2-cactusADM-bwaves-milc	M-M-H-H
mix4	gcc-leslie3d-GemsFDTD-mcf	M-H-H-H
mix5	mcf-lbm-libquantum-omnetpp	H-H-H-H

Table 4.6: Workload Setup

is listed in Table 4.6. The memory footprints for the multi-programmed workloads were collected using Sniper [67] in a computer system configured as Table 4.5. PinPoints methodology [68] was used to extract a total of 2 billion instructions, 500 million instructions for each of the threads, that best represents the characteristic of the entire workload. Then, USIMM [56] was used to estimate the energy-per-bit of each DRAM designs using IDD numbers generated from DramDSE. We assumed a single socket of the processor described in Table 4.5 to access the LPDDR4 DRAMs, but four sockets to access the HBM DRAMs. The latter is to stress the peak bandwidth of HBM DRAMs similar to that of the LPDDR4 systems. The average peak bandwidth utilization ratio of the workloads shown in Table 4.6 was 30% for both LPDDR4 and HBM systems.

4.8 Use Case Examples

In this section, we explore different DRAM architectures and their impact to energy consumption as well as area. We also re-evaluate the area overheads of DRAM circuit changes proposed by some of the recent work in the computer architecture field using DramDSE to demonstrate the importance of having better understanding of the design constraints posed by modern DRAMs as discussed in Chapter 3.

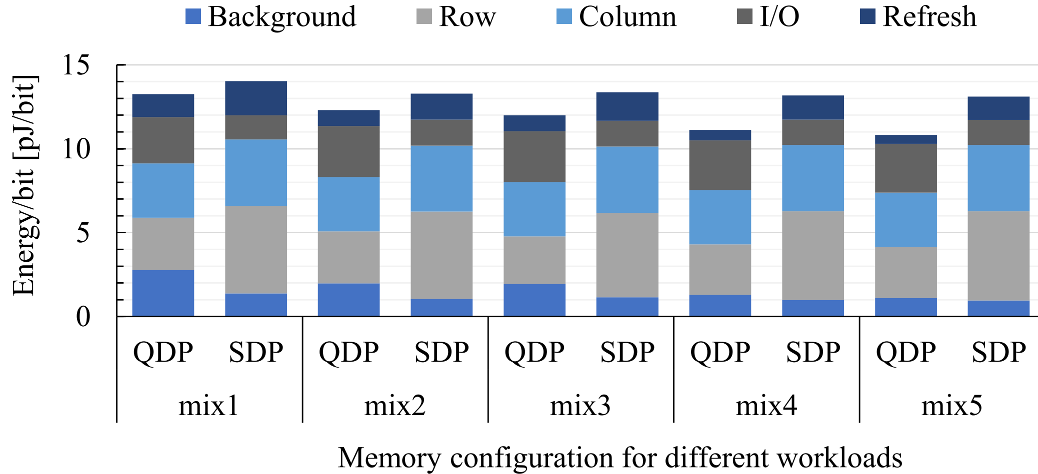


Figure 4.8: Energy/bit of 32Gb LPDDR4 built with QDP of 8Gb LPDDR4 and SDP of 32Gb LPDDR4. The I/O component includes energy consumed by the read driver and termination.

4.8.1 Density Explorations

The density-per-die is a complex function of the market's need, whether the package solution exists, and of course the cost. DramDSE is a good resource to help such decisions since it is easy to estimate area and power of different designs. To have better insight on the benefits and the overheads of having larger density-per-die, we build a 32Gb LPDDR4 using the same 2y nm we used for 8Gb LPDDR4. The die area of 32Gb LPDDR4 is 3.5× larger than the 8Gb LPDDR4 as only the periphery circuitry is amortized as density increases per die. Due to such large die size, 32Gb will only make sense using the split-die architecture of LPDDR4X [69].

Figure 4.8 compares the energy consumption for single die package (SDP) 32Gb LPDDR4 with a quad die package (QDP) of 8Gb LPDDR4s. In this analysis, we configure the four dies of the QDP to be four channels two ranks and the single die of the SDP into two channels one rank for the SDP, so that both SDP and QDP can be used in a system with a 32B cache line. The energy consumption of SDP is 1.3× the energy consumption of QDP, where the increase comes from every energy component except the background and IO. Although not shown, the QDP's IPC (Instruction Per Cycle) is on average 2× the IPC of SDP, due to QDP having

double the number of channels. Despite the performance improvements of QDP, the background energy is 1.3× of SPD's. This is because there are more physical dies on QDP that constantly burn static power in the background. The high per pin data rate of LPDDR4 (3.2 Gbps/pin) makes background energy sensitive to the number of dies. IO channel energy of QDP is 1.9× of SDP because there are two ranks instead of one. The rank other than that being accessed on QDP also has to be terminated during both write and read to provide adequate eye-opening, contributing to more IO channel energy consumption.

Column energy of SDP is increased by 21.8% compared to QDP due to the large die area. Internal IO wires such as LIO and GIO consume the most power for column access as it was discussed in Section 4.6. SDP shows significant increase in row and refresh energy, 1.73× and 1.96× respectively, compared to QDP. The page size of 32Gb is doubled from 8Gb by the JEDEC standard [24]. This increases the BLSA power, and since it dominates the power consumption for row and refresh it causes large additional energy consumption. More refreshes being issued as SDP needs more clock cycles to complete the task, also contributes to the enormous increase in refresh energy.

The energy consumption analysis might make one think moving to 32Gb is not reasonable. However, as we described earlier in this section, many factor determines density-per-die. One of the main drawback of QDP is the cost, as QDP uses more silicon area by having four separate dies to form the same density. Adding another memory channel also causes a spike in cost, because another memory controller has to be added to support the other channel and more traces have to routed on the board to connect the DRAM to the memory controller. The high demands for a large density-per-die DRAMs to configure an even larger main memory simply to avoid frequent page faults is also a trend in the big data era making SDP more favorable. As more die is put in a package, the signal integrity (SI) becomes an issue and there might not be a package solution to use QDP of smaller density-per-dies to meet the density requirement. Moreover, technology scaling will reduce the energy consumption, including row and refresh, to make larger density-per-die DRAMs viable.

4.8.2 Core Architecture Explorations

As the MAT size grows, the area efficiency of the DRAM improves and the cost-per-bit goes down. The ultimate size of a MAT is limited by either the *sensing margin* or the *core frequency* since both get worse as MAT size increases.

Sensing margin is a measure of how reliable data stored on the cell can be sensed under the worst operating conditions of DRAM with multiple sources of noise, power supply fluctuation, and charge loss due to leakage. The charge shared by the cell to the bitline (ΔV) as a wordline is selected is,

$$\Delta V = \frac{V_{BLP}}{1 + C_B/C_S} \tag{4.3}$$

when C_S is the capacitance of the cell capacitor, C_B is the bitline capacitance, and V_{BLP} is the bitline precharge voltage.[3] Since the charge sharing of DRAM is a destructive read process, ΔV has to be large enough for the bitline sense amplifier to sense and amplify the data stored on the cell correctly. From Equation 4.3, C_S to C_B ratio has to be large and with a given cell capacitance this provides a constraint in how many cells can be connected to the bitline.

Technology node scaling is closely coupled with sensing margin since coupling noise increases as bitlines and wordlines get closer to each other. A few recent industry efforts in continuous improvement of sensing margin are air spacer [14] and offset cancellation sense amplifier scheme [70].

2y nm 8Gb LPDDR4 and 2x nm 2Gb HBM used in this work have only 512 cells on the BL. This limit on the bitline length is to ensure good sensing margin for reliable memory operations. We can tweak this number, the number of SWLs on a MAT, to understand how architecture changes that are likely to happen in the future will affect the power consumption.

We will use 512 SWL as the baseline and increase the number of SWLs on a MAT to 640 SWL, 768 SWL, and 1,024 SWL. Each sub-bank for the 640 SWL

[3] The sense signal is proportional to the $V_{cell} - V_{BLP}$ but since the cell voltage is either 0 or the core voltage and V_{BLP} is the midpoint of these voltages, the change is either plus or minus V_{BLP}.

configuration will have total of 26 sub-arrays to provide the 16,384 SWLs for a sub-bank. 24 sub-arrays logically have 640 SWL, and 2 sub-arrays only need 512 logical SWL to create 16,384 SWLs. To ensure sensing margin, we make each and every sub-array symmetrical by using the remaining 128 SWL on the two MATs with 512 SWL as redundant SWLs for the entire sub-bank. In other words, instead of distributing redundant SWLs on each MAT, the redundant SWLs for a sub-bank are centralized in two of the MATs on a sub-bank. This should not be an issue as the flexibility of the row repair is high. In fact, this organization has been also proposed by industry [71]. Similarly, the 768 SWL configuration will have total of 22 sub-arrays on a sub-bank where 20 of the sub-arrays are MATs with 768 SWL. The remaining two sub-arrays will also have 768 SWL but only 512 will be visible to the user space and the rest will be filled with redundant and dummy wordlines. Finally, 1,024 SWL configuration will have a total of 16 sub-arrays on a sub-bank and each sub-array will have 16 redundant SWLs instead of 8 to have same redundant rows as before. For this many cells on a bitline, the sensing margin is so low that we will need to use an offset cancellation sense amplifier (OCSA) [70]. This scheme allows the BLSA to sense data correctly even with less sensing margin by reducing the noise seen by the BLSA. The area overhead caused by adding OCSA is assumed to be 70% of the BLSA [70] in this study. Even though we used OCSA to make up for the sensing margin of 1,024 SWL MAT, the huge overhead of OCSA only makes it reasonable to be used for MATs with 1,024 SWL.

The die size reduction of different MAT structure relative to the baseline is 3.7%, 4.9%, and 2.2% for MATs with 640, 768, and 1,024 SWLs (with OCSA) respectively in LPDDR4. For HBM, it is 2.1% reduction for both 640 and 768 SWL MAT while die size is increased by 0.6% for 1,024 SWL MAT (with OCSA) from the baseline. Connecting more cells to the bitlines amortizes the bulky bitline sense amplifier by having less bitline sense amplifiers. But it also introduces an overhead due to a larger dummy sub-array needed for each sub-bank. Thus, devices that have more sub-arrays on a sub-bank will benefit more from connecting more cells to the bitlines which is what we are seeing from LPDDR4 and HBM. Recall that 2Gb HBM we used has only half the number of sub-arrays in a sub-bank compared to the 8Gb LPDDR4 because

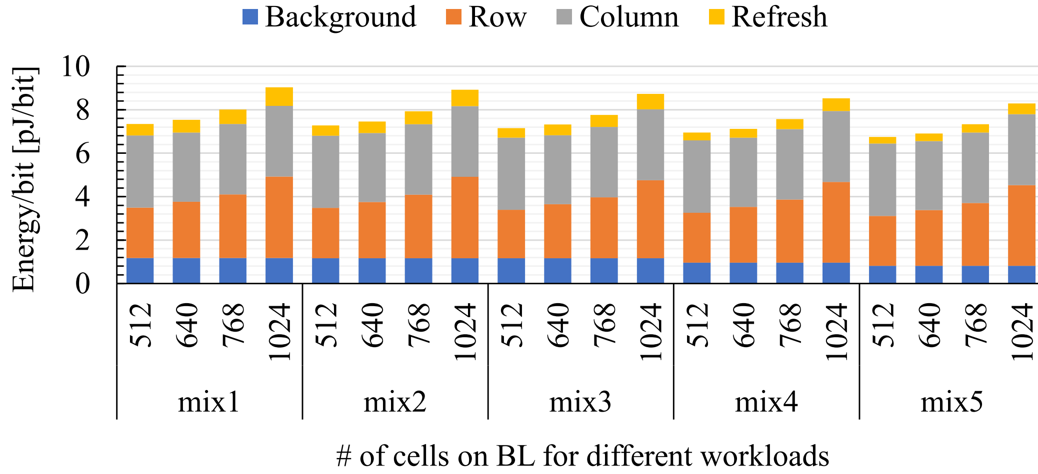


Figure 4.9: Energy/bit of 8Gb LPDDR4 for various MAT structures.

of how the data are prefetched.

Figure 4.9 and 4.10 show the energy consumption of 2y nm 8Gb LPDDR4 and 2x nm 4 Hi-stack of 2Gb HBM for the four MAT structures we are interested in exploring. As we add more cells to the bitline, bitline capacitance increases accordingly which increases both row and refresh energy even though column energy reduces slightly due to smaller die size. The increase in refresh energy is especially significant as BLSA consumes the majority of the refresh energy as we already discussed in Section 4.6. Refresh energy increases on average 13%, 28%, and 63% for LPDDR4 and 10%, 20%, and 50% for HBM when MATs with 640, 768, and 1,024 SWL are compared to 512 SWL MAT. One thing that is sure to happen as technology node scales is additional circuitry, such as OCSA to compensate for the noise floor set by the fabrication process, will become necessary to handle manufacturing fluctuations that are increasingly difficult to control only by the process itself. Based on our analysis, we project that reducing refresh energy will be increasingly important as DRAM technology nodes continue to scale and the need for larger density prevails.

4.8.3 Re-evaluating Prior Works

Our DRAM model also can be used to estimate the overhead of prior DRAM optimization. For example, **SALP** [40] proposed by Kim *et al.* mitigates bank conflict

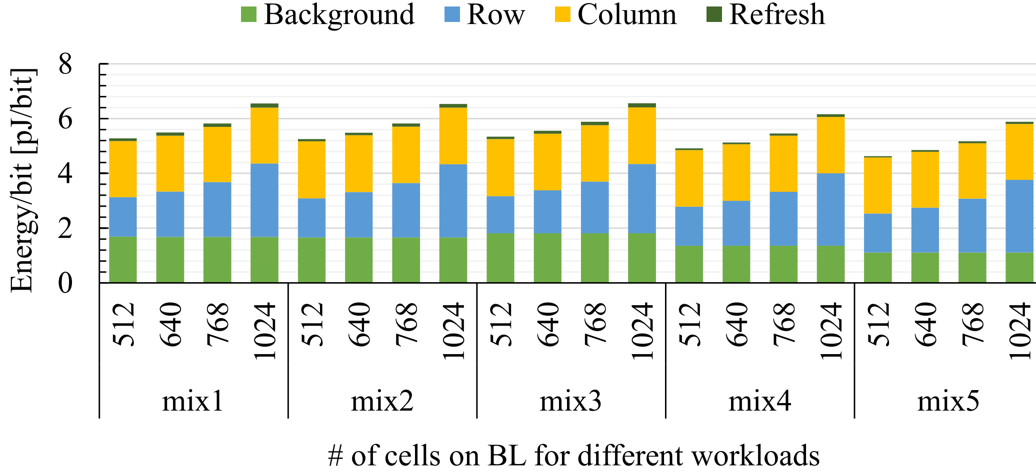


Figure 4.10: Energy/bit of 4 Hi-stack of 2Gb HBM for various MAT structures. Base die energy is not included.

overheads by exploiting the multiple sub-arrays that exist on the bank. SALP demonstrated that having 8 sub-arrays visible to the user space that can be accessed in parallel is enough to see significant improvement in performance for both single and multi-core systems and they claimed this could be accomplished with very low area overhead ($< 0.15\%$). To remove the confusion on the terminology of the *sub-array* used in this work and SALP, we will call the latter salp-subarray from now on. Unfortunately, the interleaved bitline scheme discussed in Section 3.2.2 tightly couples the adjacent sub-arrays with each other causing sub-arrays on the boundaries of different salp-subarrays to be inaccessible when one or the other is being accessed. Figure 4.11 shows the case where row_1, that is on a different salp-subarray but placed on adjacent sub-array of row_0, is accessed before closing row_0. Since half of the bitlines and the BLSAs associated with row_0 are shared with row_1, half of the data stored on row_1 will be overwritten with the values present on the bitlines and BLSAs, the moment row_1 is enabled. The tie between adjacent sub-arrays can be broken by adding one additional dummy sub-array along with a BLSA strip to every distinct salp-subarrays SALP is wishing to access in parallel. With this patch, each salp-subarrays are essentially a virtual sub-bank.

The overhead of our proposed patch to SALP can be easily evaluated using

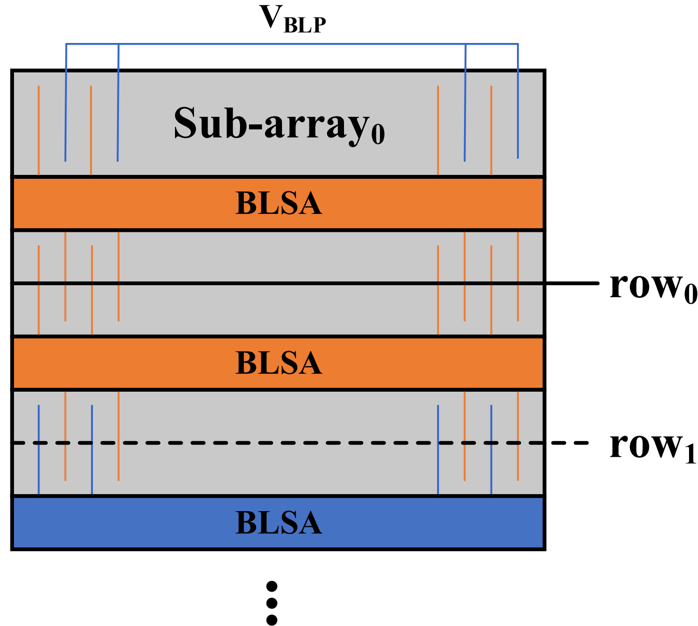


Figure 4.11: Bank conflict caused by SALP. Two rows, row_0 and row_1, are on different salp-subarrays but are placed on adjacent sub-arrays. The bitlines and BLSAs that store the cell's value on row_0 are marked in orange.

DramDSE. 8Gb LPDDR4 has two independent sub-banks that we can utilize as salp-subarrays. Hence, only three additional dummy sub-arrays and bitline sense amplifier strips are added to each sub-banks to fully utilize SALP with 8 salp-subarrays. This increases the height of the sub-bank by $135\mu m$ and the resulting area overhead to the total die is 8.2%. The area overhead of SALP becomes even more significant in DRAMs with fewer sub-banks as is the case for 2Gb HBM. Seven dummy sub-arrays and bitline sense amplifer strips have to be added to each sub-bank, resulting in an unacceptable area overhead of 29.2%. This is $\times 190$ of the reported value (0.15%).

Another example is **Half-DRAM** proposed by Zhang *et al.* [45] which splits a MAT to even and odd halves by having the SWD on the left of the MAT to drive only half of the cells and the SWD on the right to drive the other half. In this architecture, each of the SWD has to drive every half length SWL in a MAT which is twice the amount of the SWD in an interleaved wordline (Figure 3.4) would drive. The cost to support this is 15.5% for both 2y nm 8Gb LPDDR4 and 2x nm 2Gb HBM.

The next chapter uses DramDSE to help create DRAM optimization that work

within the highly constrained DRAM design space. It creates a scheme for reducing row buffer overfetch, without area or bandwidth issues, and shows this scheme can be used to also recycle charge to reduce refresh costs.

Chapter 5

Efficient Half Page Access

One outstanding source of DRAM energy consumption is the energy to fetch data stored on cells to the row buffer, which occurs during two DRAM operations, *row activate* and *refresh*. This chapter exploits previously proposed half page row access, modifying the wordline connections within a bank to halve the number of cells fetched to the row buffer, to save energy in both operations. To accomplish this, we first change the data wire connections in the sub-array to reduce the cost of row buffer overfetch in multi-core systems which yields a 38% row energy savings and a slight performance improvement by introducing new parallelism. We also propose charge recycling refresh, which reuses charge left over from a prior half page refresh to refresh another half page. Our charge recycling scheme is capable of reducing both auto- and self-refresh energy, saving more than 17% of refresh energy at 85°C, and provides even shorter refresh cycle time. Finally, we propose a refresh scheduling scheme that can extend the number of charge recycled half pages, which can save up to 32% of refresh energy at 85°C.

5.1 Motivation

The bitline sense amplifiers (BLSA), also known as the row buffer, consumes the most energy for both row and refresh operations as discussed in Section 4.6. The importance of the energy used to fetch data to the row buffer is becoming even more

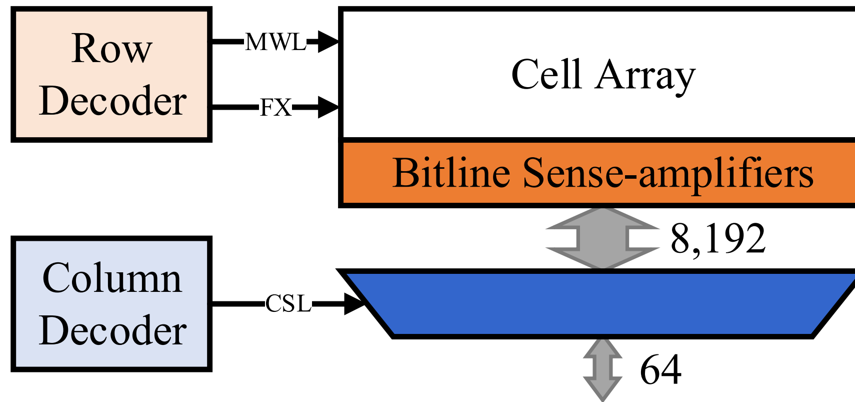


Figure 5.1: Row buffer overfetch problem shown for 8n-prefetch DDR4 DRAM. Data access starts by fetching data stored on 8,192 cells to the row buffer. But only 64 bit data is transferred in each column operations wasting most of the energy spent to fetch data to the row buffer when only few column requests are issued to the row.

important as more rows are accessed simultaneously in modern multi-core systems and more rows are refreshed as the density increases and process technology node scales. In this section, we discuss the row and refresh operation in more detail to find inefficiency in such operations that can be optimized.

5.1.1 Row Buffer Overfetch Problem

Figure 5.1 shows the data movement during the row and column operations for $\times 8$ I/O DDR4 DRAMs. DRAM data access is done by first selecting the row using the activate command and then selecting the column using read or write command as discussed in Section 2.2.1. During activate, a wordline is selected by the row decoder and the bitline sense-amplifiers (row buffer) fetch data stored on 8,192 cells that are connected to the selected worldine. When a read is issued afterwards, one out of 128 columns will be selected transferring only 64 bit data from the row buffer. This is less than 1% of the data already held in the row buffer. In modern multi-core systems, there are only 3 or 4 column requests are serviced on average to an open row before closing it [72]. This is due to the interleaved memory requests between different applications that access random locations of the DRAM. Hence, most of the energy used to fetch data to the row buffer by the row operation is wasted. This

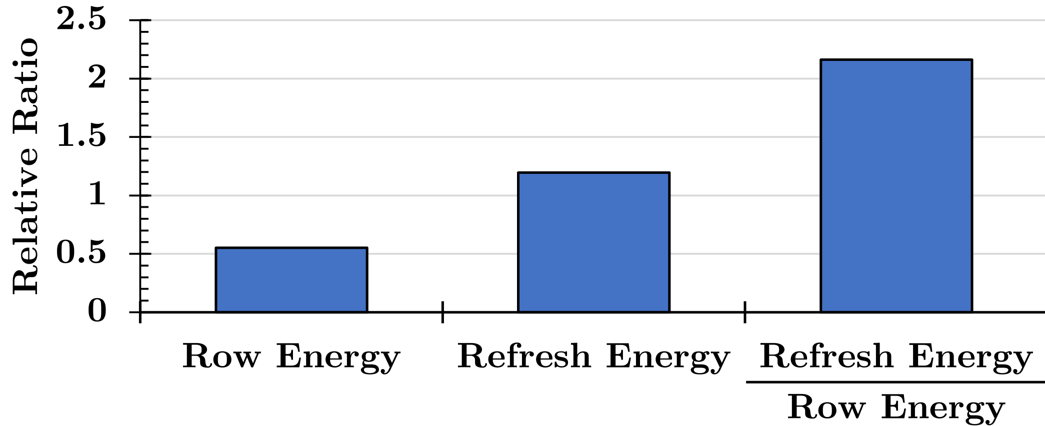


Figure 5.2: Ratio of row, refresh, and refresh/row energy for 4Gb DDR4 relative to 4Gb DDR3 that are manufactured from the same company.

phenomenon is often called row buffer overfetch [60, 73].

5.1.2 Refresh as Technology Scales

Refresh energy is becoming even more important as technology node continues to shrink. Figure 5.2 compares dynamic row and refresh energy for a 4Gb DDR4 and DDR3 manufactured by the same company. Even though row energy of DDR4 decreased by half compared to DDR3, refresh energy is slightly larger. The refresh specifications and the number of rows refreshed in each refresh cycle are the same for both DDR4 and DDR3 according to the standards [21, 22]. Hence, we conjecture that twice as many cells are being refreshed in each refresh cycle for newer technology node devices to compensate for the yield loss. We also believe that this trend will continue, making refresh energy one of the most important uses of energy in DRAM.

LPDDR2 DRAM allowed the memory controller to schedule refreshes in two very different ways as shown in Figure 5.3. The most popular method is called the distributed refresh scheduling as shown on the top of Figure 5.3 where refreshes are issued periodically every tREFI. The other method shown on the bottom of Figure 5.3 is called the burst refresh scheduling where refreshes can be issued in a burst limited by tREFBW specification followed by long periods of idle time without refreshes. However, the successors of LPDDR2 removed the burst refresh scheduling from the

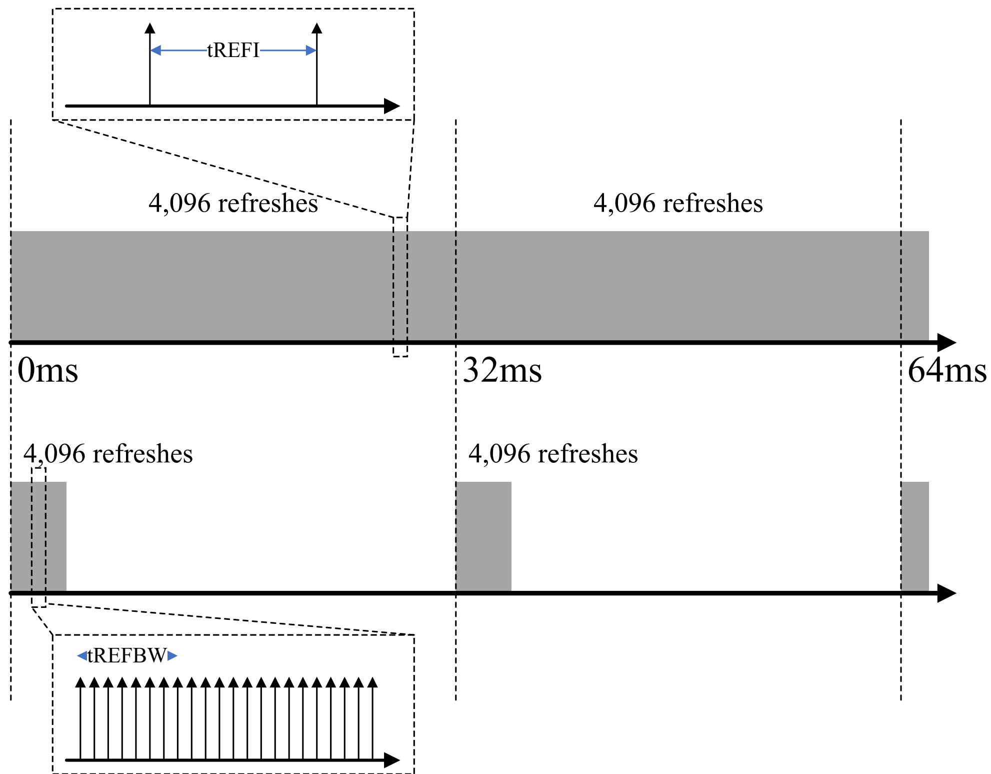
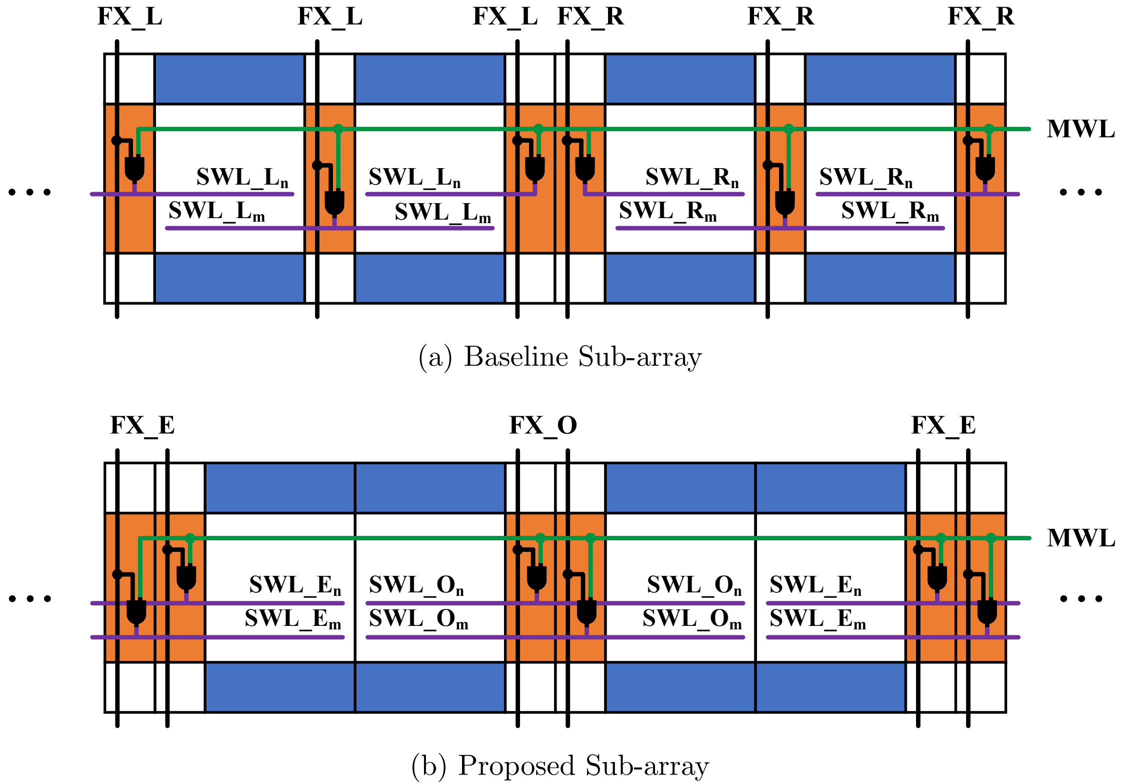


Figure 5.3: Distributed refresh issued every tREFI (top) and refresh issued in a burst followed by long periods of no refreshes (bottom) [5].

allowable refresh scheduling method and only allows distributed refresh scheduling. This is because burst refresh scheduling puts a huge burden to the DRAM manufacturer as more and more cells are refreshed in each refresh cycle to compensate for the yield loss caused by scaling technology node.

5.2 Re-organizing the Sub-array

We propose a new sub-array structure that segments the wordline in half using the $\times 4$ half page architecture of DDR4 [18] as a baseline. Our sub-array is fully compatible with a conventional DRAM and users can switch from half to full page by setting a DRAM register using MRS commands. Like the baseline shown in Figure 5.4a, our sub-array will double the wires that comes from the row decoder, which includes FX,



(a) Baseline Sub-array

(b) Proposed Sub-array

Figure 5.4: Comparison of the wordline hierarchy between the (a) baseline sub-array and (b) proposed sub-array. Both sub-array design enables half page access. To manage the RC wire delays, each vertical set of FX wires is driven by its own set of repeaters (buffers) that are placed in the "hole" above the sub-wordline drivers.

bitline equalization wires, and enable signals for sense amplifier supply voltages over the conventional sub-array. In DDR4, these extra signals are there to segment the sub-array into a right and left half, where the right 8 MATs use one set of signals and the left use the alternate set. This partition of the sub-array makes it possible to access half the sub-wordlines in a row.

We more finely interleave these same resources as shown in Figure 5.4b. To accomplish this, we need to move the wordline drivers that were previously on both sides of the MAT to the same side, and alternate odd and even FX signals to these columns of local wordline drivers. Notice that both the number and pitch of the wordline drivers are unchanged in this transformation, which avoids any area overhead. Like the baseline, activating either the odd or even FX lines enables one to access half of the

sub-worldlines in a row. However, unlike the FX connections of the baseline, the FX selecting half of the sub-wordlines (FX_E) and the other half (FX_O) are connected to alternating pairs of wordline drivers. In our design, one of the MATs neighboring the selected MAT is always not selected, which is unique to this structure, and key to the charge recycling refresh described in Section 5.4.

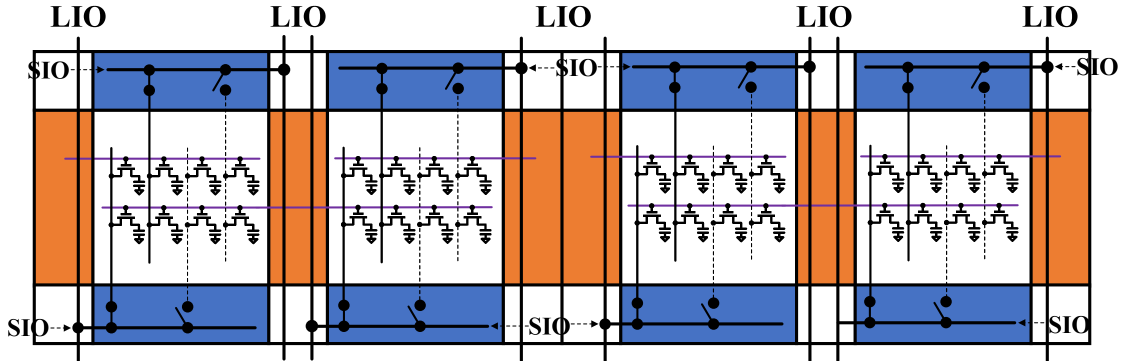
5.3 Half Page DRAM

The new sub-array structure described in the previous section cuts the page size of ×4 I/O in half so that the same page size row buffer is used regardless of I/O organization. In other words, it does not reduce the row buffer overfetch cost, but removes the inherent inefficiency of DDR3's ×4 I/O. When the new sub-array structure is used for ×8 and ×16 I/O organization, bandwidth reduces significantly because only half the LIO wires are active, so half as many bits are transferred from the sub-array with each access [45]. To reduce row buffer overfetch cost while maintaining full bandwidth, we create *Half Page DRAM* by making a small change to the I/O wiring in our previous sub-array.

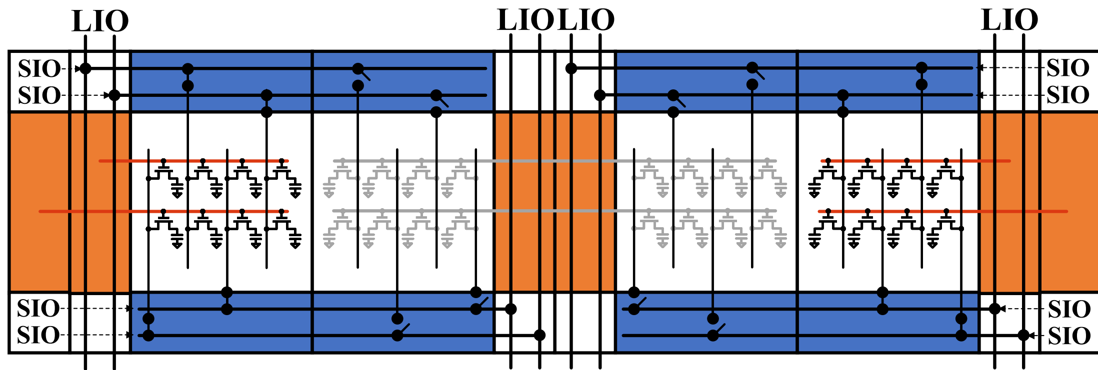
5.3.1 Proposed Design

Figure 5.5a shows the data wire connections of the baseline sub-array. Although not shown in this figure, CSL is decoded from the column address and each MAT has one CSL selected that connects 4 bitlines to the SIO wires. Hence, full I/O bandwidth for ×8 I/O DDR4 is only achieved when all 16 MATs within a sub-array transfer 4 bits data each out of the bank using the LIO wires. This is because DDR4 operates at 8n-prefetch as discussed in Section 2.3.1, so a total of 64 bits data has to be transferred with ×8 I/O DDR4.

Transferring more data from each MAT can be done by either doubling SIO wires or LIO wires. We choose to double SIO wires as shown in 5.5b to minimize area overhead. Because SIO wires are doubled, only half as many bitlines within a MAT connect to each SIO wire, so we are able to double the length of these wires, enabling



(a) Baseline Sub-array



(b) Proposed Sub-array

Figure 5.5: Comparison of the data wire connections between the (a) baseline sub-array and (b) proposed sub-array. SIO wires are doubled on proposed sub-array.

them to span 2 MATs. While this increases the wire capacitance, the number of bitline connections remains constant so the capacitance only increases slightly. SIO to LIO wire connections are also modified so that 4 bits of SIO wires connect to LIO wires on the left and the other 4 bits connect to LIO wires on the right, transferring a total of 8 bits from each MAT without changing the number of LIO wires. Finally, CSL connections are modified so that 8 bitlines, instead of 4 bitlines, are transferred with a single CSL. Instead of naively connecting CSL to twice as many bitlines in each sub-array, we connect CSL to twice as many bitlines in alternating sub-arrays. In other words, CSL connects to twice as many bitlines within each sub-array, but it is associated with only half as many sub-arrays as before. Although this modification

does not reduce the number of CSL wire tracks, it halves the number of CSL wires toggled during column operation.

After this simple modification, a row operation selects half of the MATs within a sub-array in half page mode. Only wordlines and sense amplifiers of selected MATs are enabled, fetching half as many bits to the row buffer as conventional DRAM. Also during column operation, 8 bits of data are transferred in and out of each MAT instead of 4 bits. Therefore, our design fetches only half as many bits to the row buffer while maintaining full bandwidth from the bank.

5.3.2 Memory Controller Support

We make a small change in the activate command interface protocol to enable half page row activation without needing additional pins or spare command codes. One of the existing pins on the column command array is designated as RFU (Reserved for Future Use), so we use that to indicate whether the column command is actually a dummy command. The dummy command can provide the extra address bit needed to initiate half page row activation, while the actual column operation is not performed. With this additional command, the DRAM stores row address information from the activate command for one clock cycle. Then, the dummy command is issued on the next cycle to initiate a row activation by merging the address provided by the dummy and activate commands. The additional cycle used for the dummy command is not expected to degrade performance much due to frequent bubbles present in the CA bus traffic [74], while row activation is ensured to occur on the next cycle of the activate command.

5.4 Charge Recycling Refresh (CRR)

The drivers for the sense amplifier power supply, SAP and SAN, are located at the sub-hole which is the cross-section of the sub-wordline drivers and the sense amplifiers [75]. Each SAP and SAN supplies power to half of the sense amplifiers in each MAT by a metal wire [76]. In conventional refresh, bitline (BL) and reference bitline (BLB)

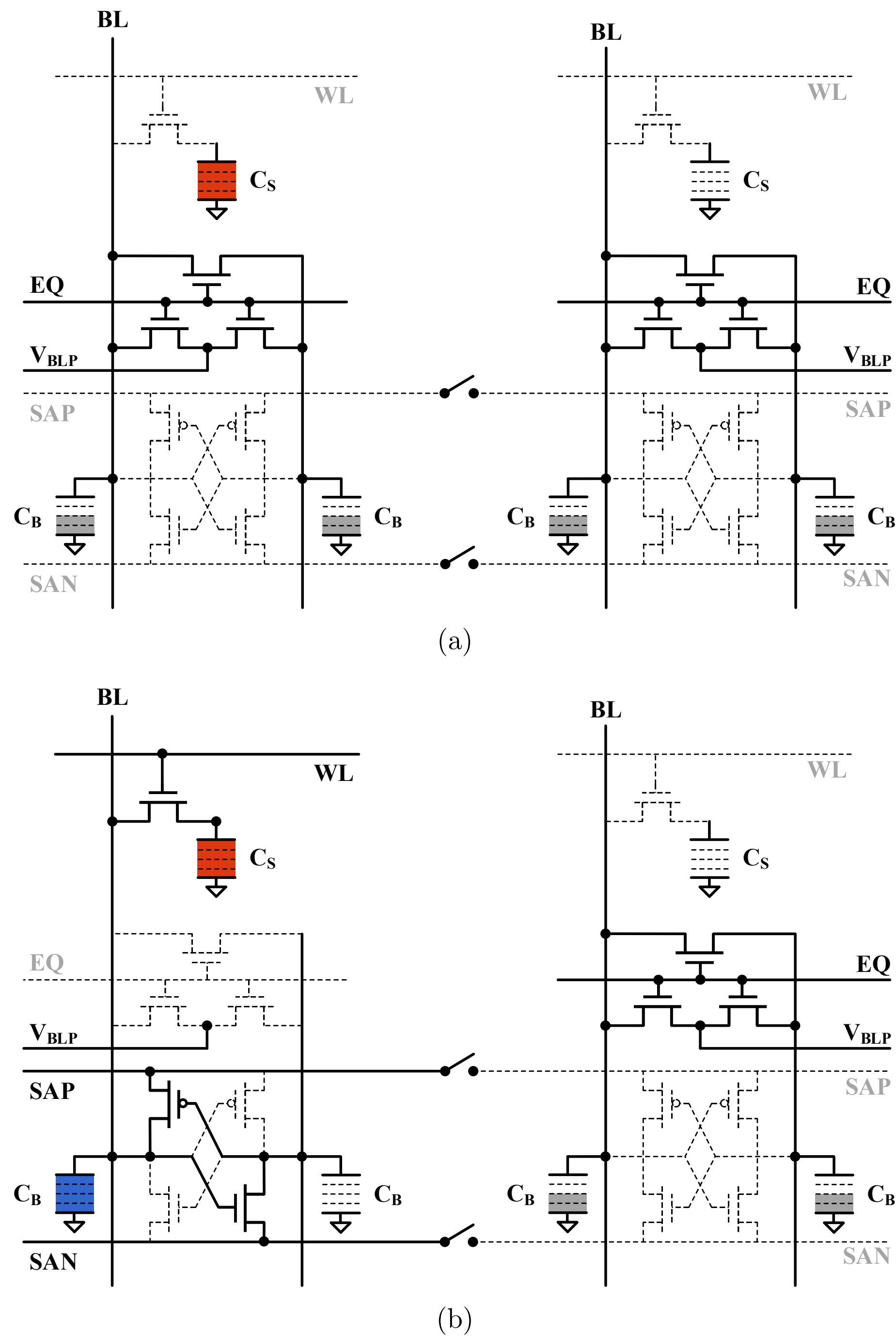


Figure 5.6: Conventional refresh done to the cell on the left: (a) Idle. BL and BLB are equalized, and cell is isolated from the sense amplifier. (b) Refresh. Bitlines and cell are fully restored using the charge supplied by the power supply lines, SAP and SAN.

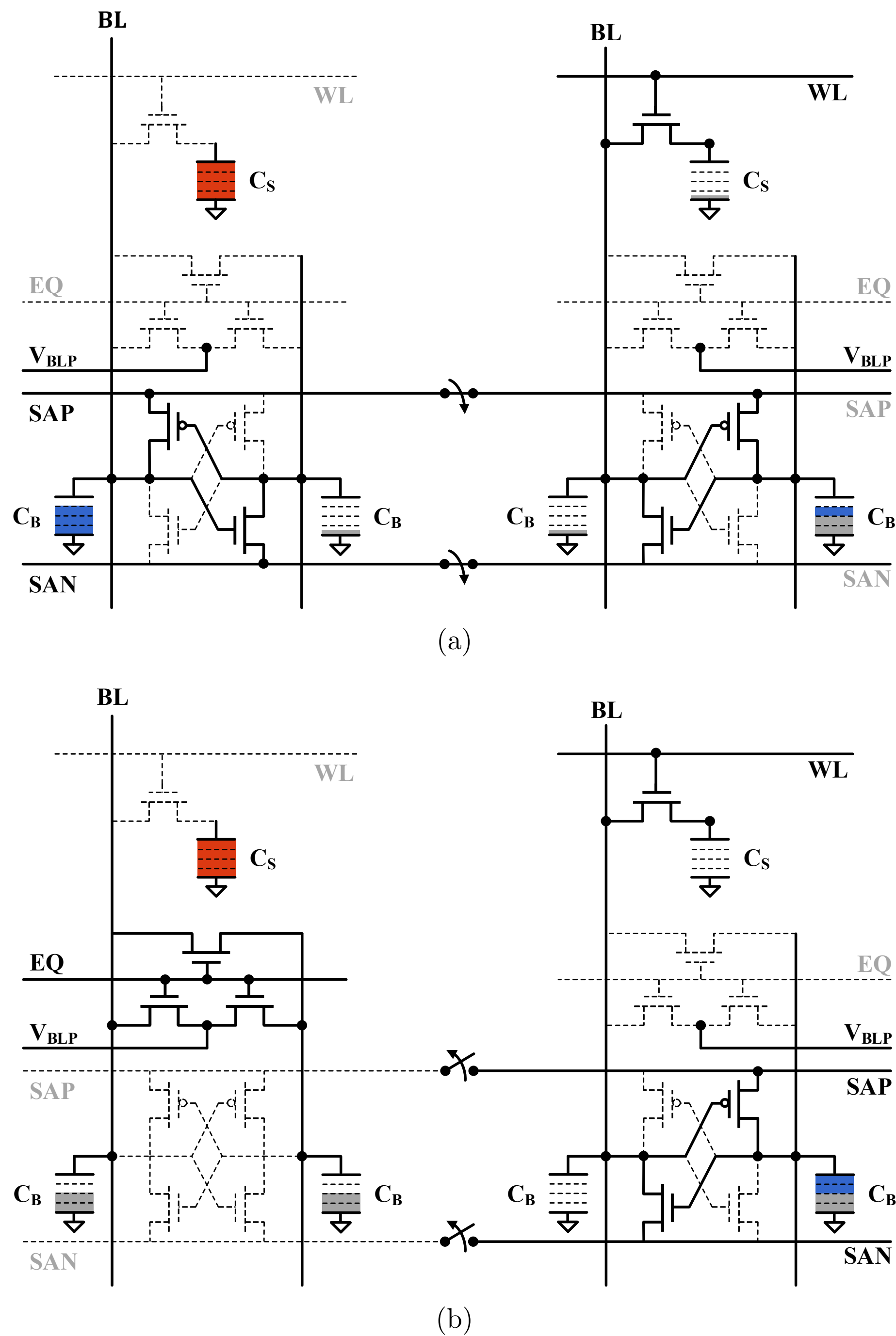


Figure 5.7: CRR done to the cell on the right by recycling charges from the cell on the left: (a) Charge Recycling. Bitlines on the left cell supply half of the charge needed to fully restore the bitlines on the right cell. (b) The rest half of the charge needed to fully refresh the cell on the right is supplied by the SAP and SAN.

are equalized by shorting them to each other, and sharing the charge on the lines as shown in Figure 5.6a. When a cell is accessed, charge sharing between the cell and the BL causes a small perturbation to the charge stored on the BL. The sense amplifier then senses the charge difference between BL and BLB and amplifies them to full digital value using charge supplied by the power supplies, SAP and SAN, as shown in the left half of Figure 5.6b.

The concept of charge recycling is to use the fully amplified charge stored on the BL and BLB of another MAT as a battery which supplies charge to the BL and BLB being sensed. Hence, the sense amplifier on the right half of Figure 5.7a begins to sense the data stored on the cell by recycling charge from unequalized bitlines shown on left half of Figure 5.7a. This is done by simply shorting the SAP and SAN of two different MATs as shown in Figure 5.7a. At the end of this charge recycling process, half of the charge to fully amplify BL and BLB on the right has been supplied from the bitlines on the left. BL and BLB are then fully restored by disconnecting the "battery" bitlines, and connecting the sense amplifier to SAP and SAN as shown in Figure 5.7b. It is notable that the BL and BLB lines that were used as the battery will still result in the correct half $\mathrm{V_{DD}}$ precharge voltage when they are equalized.

Shorting SAP and SAN of two different MATs is the key to the success of CRR, which allows charge recycling possible regardless of the distribution of data stored on the cells. Since the bitline sense amplifier is differential, one of the bitline will store '1' and connect to the SAP while the other bitline will store '0' and connect to the SAN after full amplification. For example, in Figure 5.6 and Figure 5.7, the cell on the left stores '1' hence the BL connects to SAP and BLB connects to SAN, whereas the BL on the right connects to SAN and BLB on the right connects to SAP as the cell on the right stores '0'. Thus, regardless of the data stored on the cells, half of the bitlines on the MAT always connect to SAP and the other half connects to SAN. The same applies to the other MAT, and shorting the SAP and SAN of these MATs always result in the charge stored on the bitlines connected to SAP and SAN to reach roughly 75% and 25% of V_{DD} respectively as shown in Figure 5.7a.

5.4.1 Proposed Design

Our new sub-array design, shown in Figure 5.4b, makes charge recycling refresh (CRR)[1] feasible in modern DRAM for the first time. Since a neighbor MAT always belongs to a different half page, CRR can be performed on half pages simply by shorting SAP and SAN of one half page to the SAN and SAP of the other as shown in the top side of Figure 5.8. For better productivity, dummy cells are also placed on the voids caused by adding the switches. Because dummy cells are solely for pattern matching, wordlines of these cells are connected to V_{BBW}, ground voltage of the wordline, and bitlines are connected to V_{BLP}, bitline precharge voltage.

Operation of CRR is shown in the timing diagram on the bottom side of Figure 5.8. First, the wordlines in an even half page are selected and bitlines associated with the wordlines are amplified by the sense amplifiers. When data is fully restored, wordlines and sense amplifiers are deselected, but bitlines are not yet precharged. Then, the wordlines in the odd half page are selected, sharing charge between the cells and the bitlines. Charge recycling occurs by enabling *RE*, which shorts SAP_n to SAP_m and SAN_n to SAN_m. Because MAT_n and MAT_m are identical, up to half of the charge stored on the bitlines of MAT_n are transferred to the bitlines of MAT_m using SAP and SAN. After recycling charge, bitlines in the even half page can be precharged and the sense amplifiers of the odd half page can be enabled to supply the rest of the charge to the bitlines using SAP_m and SAN_m.

5.4.2 Extending to Multiple Rows

Ideally, half of the charges on the bitlines are recycled using CRR because the bitline capacitance of two half pages is the same. However, one half page is fully charged using the power supply and only one half of the page is supplied with recycled charge, achieving up to 25% less energy (instead of 50%) to charge and discharge bitlines during refresh than conventional. Because there are multiple rows within a MAT, we can continue recycling charge to half pages on different rows using CRR, reducing

[1] CRR is built on top of the new sub-array structure and it is independent of Half Page DRAM. Hence, CRR does not need additional design changes to transfer twice as much data out from each MAT.

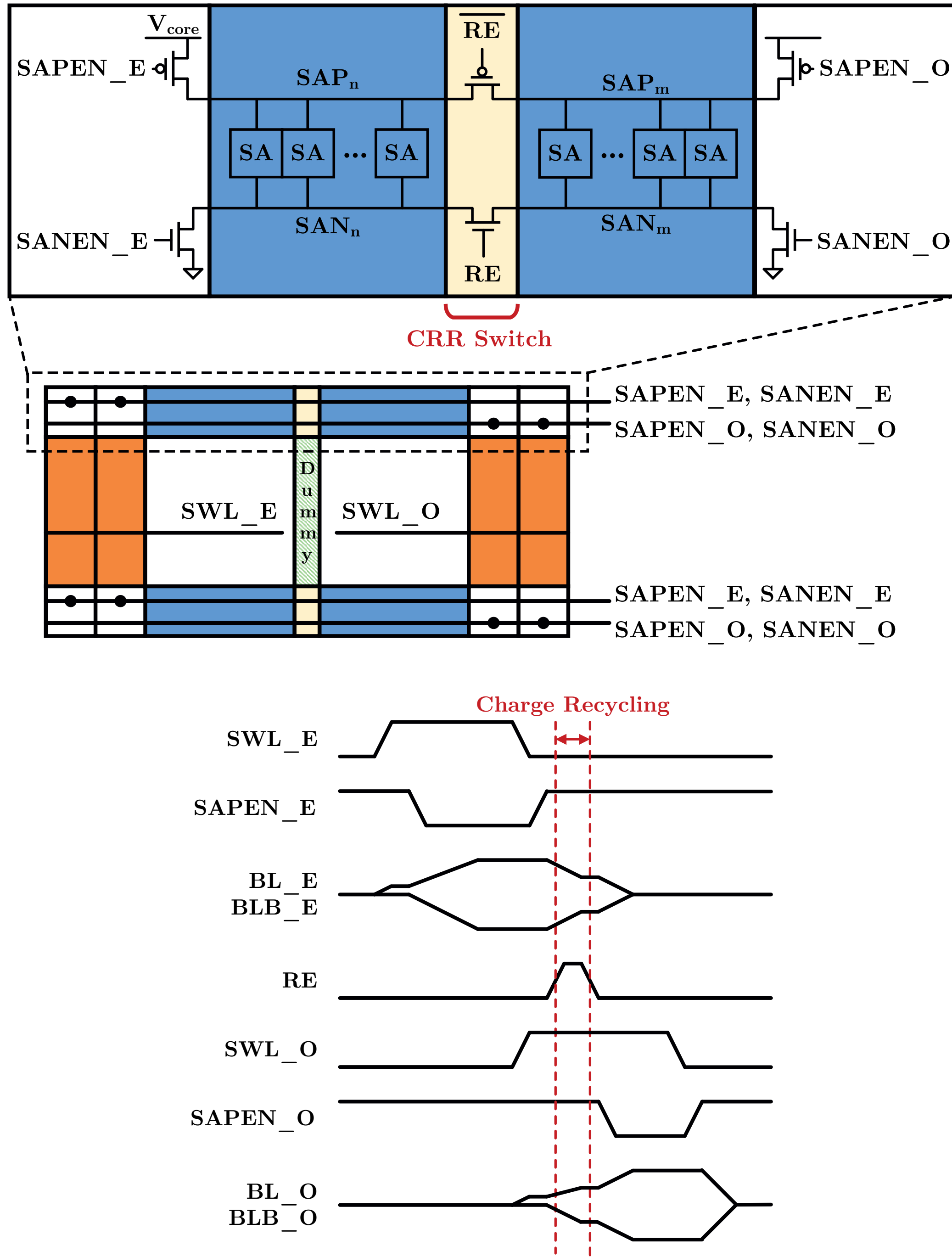


Figure 5.8: Design of the proposed charge recycling refresh scheme (top) and its operation (bottom). Switches are added between two adjacent MATs that are on different half pages. Charges are recycled from the bitlines of even half page to odd half page through SAP and SAN wires by turning switches on.

refresh energy even further. We name this feature *multiple row CRR*.

To support selecting two half page rows with different row addresses, we need to slightly change the peripheral circuitry. We propose to store, in each sub-bank, the row address associated with FX and one column address bit that distinguishes FX_E and FX_O for each half page. This allows us to select both FX_E and FX_O lines that are completely different from the row address, which was not possible before. Typically there are 8 FXs in each MWL [18] hence, the proposed multiple row CRR can recycle charge for up to 15 half page rows.

Each half page row has to be refreshed in consecutive order as shown in Figure 5.9c to enable multiple row CRR. For this purpose, we propose to pull-in multiple refreshes where each refresh is performing CRR as shown in Figure 5.9d. JEDEC standards allow DDR4 to pull in and postpone up to 8 refreshes [21, 22]. This feature provides an extra memory controller knob to flexibly schedule refreshes based on the workloads. For instance, refreshes can be pulled-in when memory is being used less often, so that future refreshes can be postponed to free up extra memory bandwidth later when it is being used more often. Multiple row CRR can pull-in CRRs to concatenate multiple CRRs back-to-back. Once CRRs are issued consecutively, we can continue recycling charge by not closing the row on the previous refresh cycle until it finishes recycling charge to the row on the following refresh cycle. For auto-refresh, the memory controller informs the DRAM that this is the last consecutive refresh to be issued using one address bit of the refresh command. During self-refresh, DRAM can determine by itself when to close a row.

To better illustrate how multiple row CRR works, Figure 5.9 compares the order of the rows being refreshed and corresponding refresh scheduling for conventional refresh versus proposed multiple row CRR. As an example, we show the case where four full page rows recycle charge for multiple row CRR in Figures 5.9c and 5.9d. In conventional DRAM, refreshes are performed at full page granularity as shown in Figure 5.9a. Refreshes are issued periodically every tREFI by the memory controller and each refresh takes time tRFC as shown in Figure 5.9b. In $\times 4$ CRR, however, refreshes are performed at half page granularity and two half pages are refreshed consecutively in one refresh cycle to recycle charge from even to odd half pages as

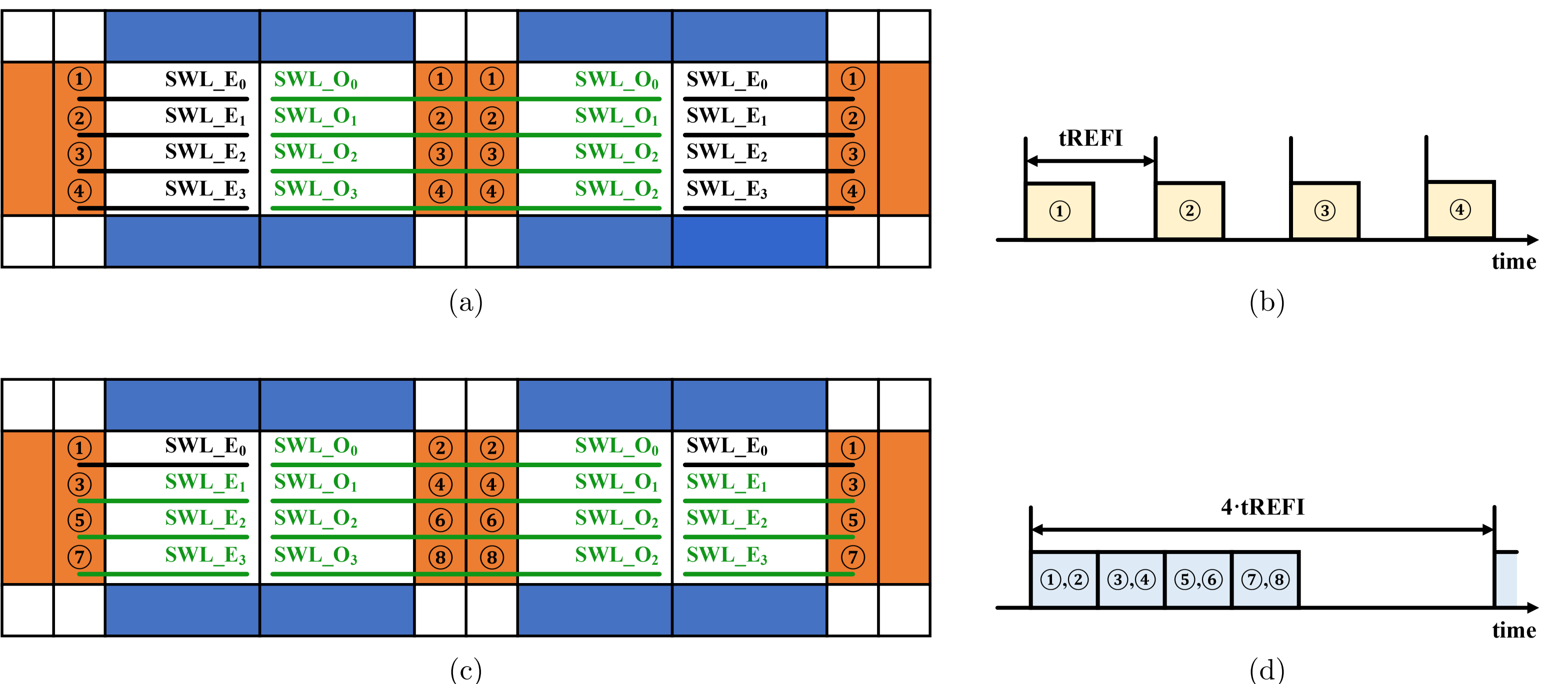


Figure 5.9: Comparing refresh sequence and refresh scheduling scheme between conventional and ×4 CRR when four full page rows are refreshed by issuing 4 refresh commands. (a) Order of the rows refreshed in conventional refresh. (b) Conventional refresh scheduling scheme where refreshes are issued periodically every tREFI and each refresh consuming tRFC time to complete refresh. (c) Order of the rows refreshed in ×4 CRR. Charges are recycled to half page rows except for even half page of first refresh. (d) Proposed refresh scheduling scheme where 4 refreshes are issued back-to-back.

shown in Figure 5.9c. Also, four refreshes are issued back-to-back so that charges are recycled to seven half page rows consecutively, saving even more energy than just using CRR or ×1 CRR. Successive refresh(es) are issued after 4·tREFI from the start of ×4 CRR as shown in Figure 5.9d. This ensures that every cell is refreshed without violating the refresh specification.

The proposed refresh scheduling for multiple row CRR is effective because it provides great flexibility in trading off refresh energy with performance. The memory controller can change the number of CRRs to pull-in on-the-fly to either maximize refresh energy savings or performance, depending on the workload. It is notable that refresh energy is saved even when maximum performance is needed because CRR is used instead of conventional refresh. Moreover, the limit of multiple row CRR matches with the limit of the number of CRRs that can be pulled-in, which allows exploitation of multiple row CRR to its full extent using the proposed refresh scheduling.

5.5 Detailed Analysis

Two new designs, Half Page DRAM and CRR, were proposed in previous sections that exploited half page to reduce row buffer overfetch cost and refresh energy. In this section, we analyze the potential benefits and cost of their implementation.

5.5.1 Additional Parallelism

Two factors degrade performance in Half Page DRAM: (i) a one cycle delay for row activation, and (ii) an additional row activation required when column accesses hit both half pages on the same row back-to-back. However, Half Page DRAM also provides an additional degree of row parallelism, which can potentially improve overall performance even with the above-mentioned performance overheads.

Kim *et al.* [40] proposed SALP, which exploits independence between sub-arrays and overlaps accesses to rows in different sub-arrays to reduce the negative impact of bank conflicts. But because row buffers are shared between two adjacent sub-arrays as described in Section 3.2.2, sub-arrays are not completely independent to

each other, which limits the operation of SALP. This limitation becomes even severe when row repair is considered because multiple chips in the module have different repair mappings, increasing the possibility of adjacent sub-array access. Unlike sub-arrays, row buffers are not shared between any two half pages within the whole bank. Hence, half page level parallelism can provide row parallelism equivalent to having twice as many banks.

We call the additional row parallelism provided with Half Page DRAM as Half Page Level Parallelism (HPLP)[2]. In this work, the precharge time of one half page row is completely overlapped with the activate of any other half page row within a bank, given that both rows are on different half pages.

5.5.2 Energy Saved Using Half Page DRAM

DramDSE [58] was used to break down row energy for each power supply, V_{DD} and V_{PP}. Most, i.e. 68% of the V_{DD} power supply energy, is used to charge and discharge bitlines. The rest of the energy is dissipated in the periphery circuits to generate and transfer necessary control and address signals to the row decoder. The V_{PP} power supply is used to raise wordlines whose energy we break down further into MWL, FX, and SWL as discussed in Section 3.2.1. Based on the model, MWL consumes 20%, FX consumes 72%, and SWL consumes 8%.

Half Page DRAM reduces row energy by activating only half of the page compared to conventional DRAM. This is done by selecting half as many SWLs and sense amplifiers as before. In our design, MWL is shared with every cell on the full page but FX is connected to only half of the cells to selectively enable half of the SWL. This results in 50% of energy to toggle load and wire of SWL, and 50% of energy to toggle load of FX but not the wire of FX. Overall, Half Page DRAM saves 39% of the V_{PP} power supply energy. Selecting only half of the sense amplifiers reduces energy to charge and discharge bitlines by half, which saves 34% of the V_{DD} power supply

[2]In this work, we only consider avoiding precharge time penalty caused by bank conflicts, which is equivalent to SALP-1 of SALP [40]. Although half page level parallelism can be used to completely overcome the row buffer limitation of two extensions of SALP-1, SALP-2 and MASA, it does not remove the row repair limitation, which affects die yield.

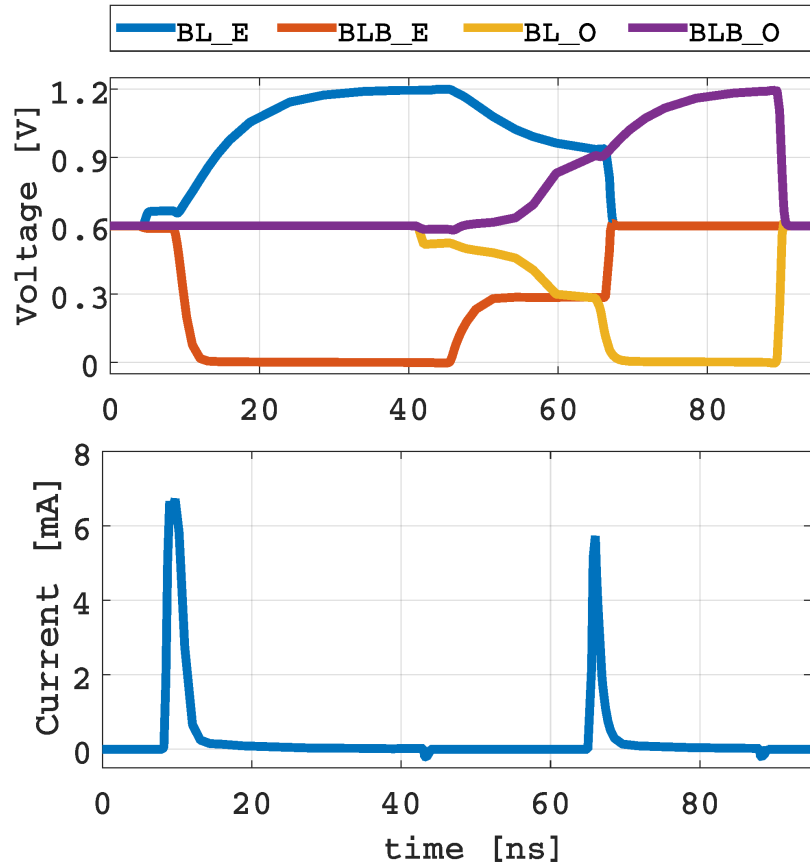


Figure 5.10: SPICE simulation of CRR when charge transfer time is 20ns; bitline development (top) and V_{ss} current (bottom). Narrower and shorter current peak shown during amplification of odd half page bitlines indicates that less charge is supplied by the power supply than before.

energy. Column energy is slightly reduced by using Half Page DRAM because CSL transfers twice as many bitlines in each MAT than conventional, reducing the energy to toggle CSL by half.

5.5.3 Charge Transfer Time of CRR

The SPICE simulation of CRR shown in Figure 5.10 used the Predictive Technology Model [77], which best fits DRAM timing constraints for the transistor size on the 55nm Rambus Power Model [33]. We chose the size of the switch that shorts SAP and SAN of two adjacent MATs to be roughly half the size of the SAP and SAN

Temp.	10ns	15ns	20ns	30ns	35ns
85°C	33.3%	42.1%	46.3%	49.3%	50.0%
55°C	39.0%	45.6%	48.4%	49.8%	50.0%
25°C	44.0%	48.3%	49.5%	49.9%	50.0%

Table 5.1: Percentage of charge recycled as charge transfer time and temperature vary.

drivers. The size was chosen to recycle half of the charge stored on bitlines from one half page to the other at $tRAS_{min}$. As shown in the top half of Figure 5.10, charge is recycled from even half page to odd half page, which can then amplify bitlines of the odd half page based on the cell data sensed by the sense amplifier. It is clear that, by recycling charge, less energy is dissipated from the power supplies, as shown at the bottom of Figure 5.10.

Table 5.1 shows how much charge is recycled from one half page to the other half page as the charge transfer time and temperature are varied. Because the switch shorting SAP and SAN of two adjacent MATs act as a resistor while transferring charge, more charge is transferred as temperature gets colder. As expected, 50% of the charge is recycled starting around $tRAS_{min}$ which is between 30ns to 35ns. We can also see that even at 85°C, more than 40% of the charge transfer time is spent to recycle the last 7% of the charge.

5.5.4 Energy Saved Using CRR

CRR saves refresh energy by reducing the energy to charge and discharge bitlines. This reduces the energy dissipated by the V_{DD} power supply but does not affect the energy dissipated by the V_{PP} power supply. Figure 5.11 shows the effect of CRR on refresh energy using V_{DD} power supply at 85°C. Even with ×1 CRR and 20ns of charge transfer time, CRR effectively reduces 19.7% of the V_{DD} powered refresh energy. More energy is saved as more charge is recycled, which can be achieved by either increasing the charge transfer time or by increasing the number of rows that recycle charge. Expected refresh energy savings shown in Figure 5.11 conservatively

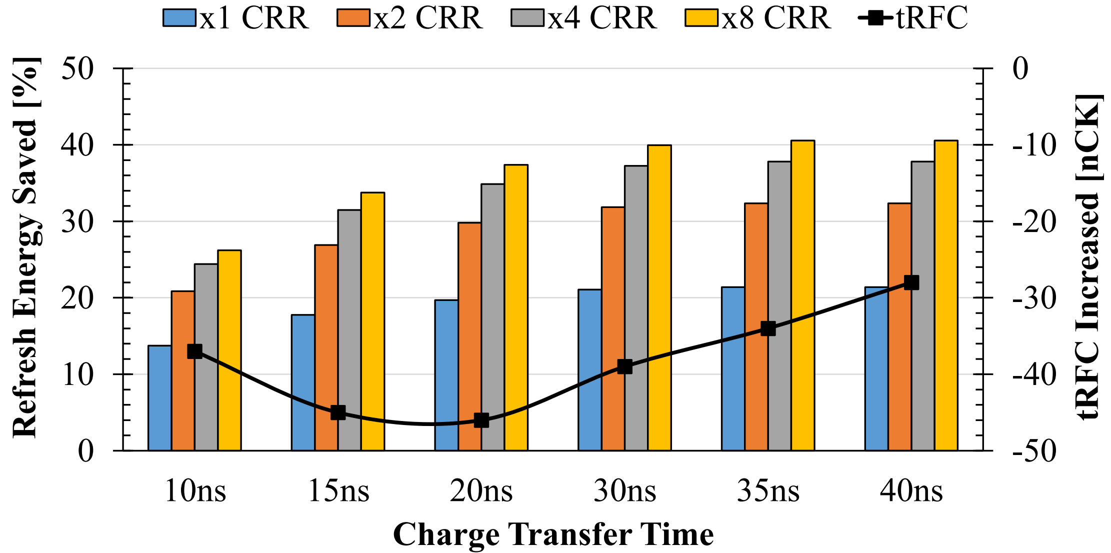


Figure 5.11: Refresh energy saved for V_{DD} power supply with CRR at 85°C. tRFC change as charge transfer time varies is also shown for 8Gb DDR4.

assumed that any repaired row would result in ×1 CRR for all refreshes being pulled-in during multiple-row CRR. We also assumed 1.56% of the total rows were repaired rows [78], hence for ×8 CRR, 87.5% of the refreshes were considered as ×8 CRR and the rest as ×1 CRR.

Refresh cycle time (tRFC), which is the time that memory is blocked from use during refresh, is also a very important parameter for the refresh operation. tRFC is constrained by the power spent to refresh many cells simultaneously. It is set to avoid instantaneous drop in power supply voltage, especially V_{DD}, so that data is fully restored back to the cell by refresh. CRR can potentially reduce tRFC because (i) refresh power is distributed over time by refreshing one half page at a time and (ii) less energy is dissipated to refresh the same amount of cells. Figure 5.11 shows the change in tRFC compared to conventional as the charge transfer time and reduction rate of V_{DD} power supply refresh energy change. tRFC is estimated to be reduced by 45 clock cycles[3] for 15 ns and 20 ns of charge transfer time of 8Gb DDR4 [31]. We use

[3]Additional time needed by CRR is only counted for the last row being refreshed and, with 20 ns of charge transfer time, it increases tRFC by $20ns + (tRAS - tRCD) = 38.3ns$. As the effect of energy saved is applied in addition to the previously calculated tRFC, it changes to $38.3ns - (tRFC \cdot EnergySaved) = -39ns$.

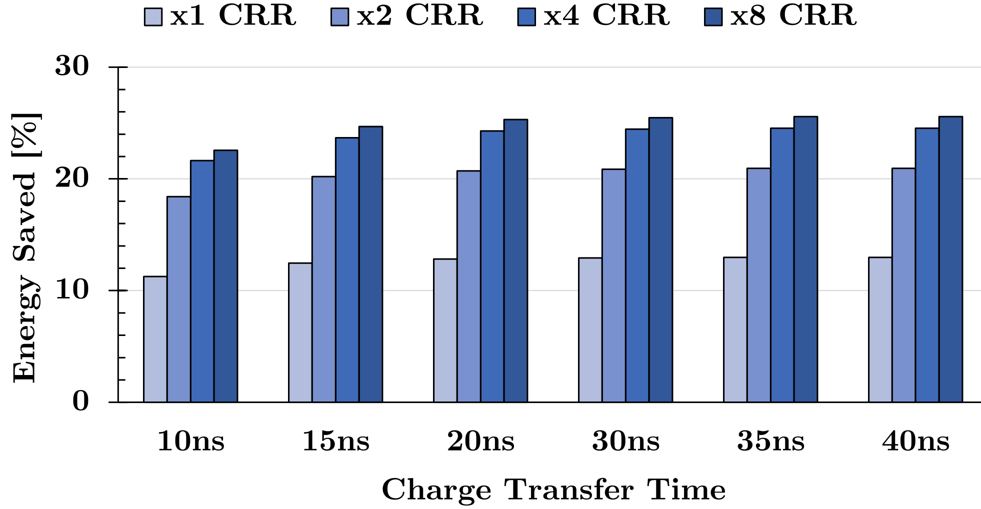


Figure 5.12: DRAM standby energy saved when CRR is used during self-refresh at 25°C. Refresh energy portion during self-refresh mode is assumed to be 80% of DRAM energy [6].

20 ns of charge transfer time for the remaining part of the work, as it transfers 92% of the maximum possible charge to recycle and provides the shortest overall refresh time. This saves 16.9% of V_{DD} and V_{PP} combined refresh energy with 45 clock cycles less tRFC for ×1 CRR.

Refresh energy is a significant portion of total energy at low temperatures because static leakage is suppressed as temperature gets colder. Hence, it is not surprising that a separate Micron product line, DDR3L-RS, reduces its self-refresh rate by half at temperatures below 45°C to reduce overall DRAM energy. DDR3L-RS is reported to reduce 40% of standby power at 25°C [6], which means refresh itself consumes roughly 80% of the standby power during self-refresh. Figure 5.12 shows total DRAM energy saved at 25°C when DRAM is in self-refresh mode. We use the reported energy breakdown for DDR3L-RS self-refresh, which we think is conservative considering the trend in refresh energy. The proposed refresh scheduling can fully utilize the potential of multiple row CRR in self-refresh mode by using ×8 CRR to refresh cells while in self-refresh and ×1 CRR as we exit from self-refresh. This can provide maximum energy savings while reducing the penalty to recover from self-refresh thanks to the short tRFC of ×1 CRR. Assuming the same 20ns charge transfer time, 25.3% of

standby energy is estimated to be saved by using ×8 CRR during self-refresh

Special auto-refresh features such as per-bank refresh [24] and Fine Granularity Refresh (FGR) [22] can also save energy by applying CRR to them. However, tRFC might increase in return, unlike all-bank auto-refresh. This is because additional time spent to recycle charge from one half page to the other are not fully mitigated by the energy being saved, depending on how many cells are being refreshed.

5.5.5 Area Overhead

There is no area overhead in our half page sub-array structure, because we just re-organized the sub-array of DDR4's existing ×4 half page architecture without adding any new components. However, new components added to enable two other designs, Half Page DRAM and CRR, that were built on top of our sub-array structure costs small additional die area.

Half Page DRAM needs additional 4 bit SIO/SIOB wires in each sub-array. Additional SIO/SIOB wires are routed on the sense amplifier region and increase the height of the chip. Sense amplifiers are staggered and shared between two adjacent sub-arrays as described in Section 3.2.2. Hence, 64 additional SIO wires are added instead of 128 wires for each sub-bank because there are 32 sub-arrays in each sub-bank [79]. Area overhead caused by additional SIO wires are estimated by first predicting the pitch of SIO wire using the pitch of the CSL. There are 128 CSLs total within a MAT of 512 columns. If F is feature size then bitline pitch is 2F in a $6F^2$ cell [80], and the wire pitch of CSL and SIO is therefore 8F. For the 55nm Rambus Power Model we used, 64 additional SIO wires plus 64 SIOB increases the height of each bank by $56.3\mu m$, which is estimated to be 1.4% area overhead for the whole chip.

CRR increases the width of the chip by adding switches to short SAP and SAN wires of two adjacent MATs. As described in Section 5.5.3, the size of the switches were chosen to be roughly half of the SAP and SAN drivers. Each switch increases the width by 32F using the size information provided by the 55nm Rambus Power Model [33] and conservative design rules. In each sub-array there are 16 MATs [39],

Processor	4-core, 2.4GHz, Nehalem architecture [65]
LLC	shared 4MB, 16-way LRU, 64B cacheline
DRAM Controller	FR-FCFS, relaxed close-page policy 128-entry read/write queue DDR4 enters fast precharge power-down mode when DRAM is idle LPDDR4 enters self-refresh mode when DRAM is idle
DRAM	2 Channels 1 Rank 8Gb DDR4-2400 17-17-17 [31] 4 Channels 1 Rank 8Gb LPDDR4-3200 [19]

Table 5.2: System Configuration

resulting in 8 additional switches per bank. This increases the width of each bank by $14.1\mu m$ and is estimated to be 1.2% area overhead for the whole chip. Another component that we added to enable CRR includes the latches and corresponding wires added in each bank to store the one bit column address bit that selects either even or odd half page, and the 3 bit row address that represents FX. The area overhead caused by these latches is negligible considering that adding many more latches for each sub-array, instead of each bank, was also estimated to be negligibly small in other work using the same model [45, 40].

5.6 Evaluation Methodology

We use USIMM [56] to evaluate the potential advantage of our proposed Half Page DRAM and CRR in a system configured as shown in Table 5.2. The main difference compared to the system configuration used in the previous chapter that was described in Table 4.5, is the increase in cacheline size from 32 B to 64 B. This is a typical cacheline size for computer systems using commodity DDR4 DRAM DIMMs. An FR-FCFS [81] scheduler with write drain mode is used for the memory controller to maximize DRAM throughput. Relaxed close-page policy is used for the scheduler that closes the row when there are no pending memory request to the opened row. When there are no pending memory requests in the memory controller and all of the DRAM banks are precharged, DDR4 DRAMs enter fast power-down mode to save

Symbol	Benchmark Mix	MPKI
mix0	perlbench-namd-gobmk-povray	Low
mix1	gromacs-claculix-perlbench-gamess	
mix2	povray-gamess-sjeng-namd	
mix3	wrf-bzip2-h264ref-hmmer	Medium
mix4	astar-zeusmp-hmmer-wrf	
mix5	xalancbmk-cactusADM-astar-gcc	
mix6	milc-mcf-sphinx3-omnetpp	High
mix7	bwaves-soplex-leslie3d-mcf	
mix8	lbm-GemsFDTD-libquantum-milc	

Table 5.3: Workload Setup

standby power. Since the overhead of entering and exiting self-refresh mode is lower in LPDDR4 compared to DDR4, LPDDR4 DRAMs enter self-refresh mode instead of fast power-down mode to save even more standby power. A 8GB DRAM module is formed using eight 8Gb DDR4-2400 DRAM chips with CL-nRCD-nRP setting of 17-17-17. The same I/O termination suggested by Micron's DDR4 power calculator spreadsheet [82] is used, hence, the termination powers are extracted directly from the spreadsheet. Similarly, 4GB LPDDR4 package is formed using four 8Gb LPDDR4-3200 DRAM chips where we access both channels on the die simultaneously to transfer 64 B of data for each DRAM accesses. The I/O termination power for LPDDR4 is found by running SPICE simulation assuming 350 mV DQ swing.

Proposed designs are evaluated using multi-program workloads composed of SPEC CPU2006 [66] benchmarks as listed in Table 5.3. Unlike the workloads listed in Table 4.6, there are even mix of low, medium, and high MPKI (Miss Per Kilo-Instruction) workloads. This is to show the potential benefits of our proposed designs in various real use cases instead of just the best scenario. The DRAM footprint of the workloads is collected for a total of 2 billion instructions, 500 million instructions for each of the programs, using PinPoints methodology [68] and Snipersim [83] to select the best representative region of the workload. Energy breakdown of each workloads using the category discussed in Section 2.4 is shown in Figure 5.13 for both DDR4 and

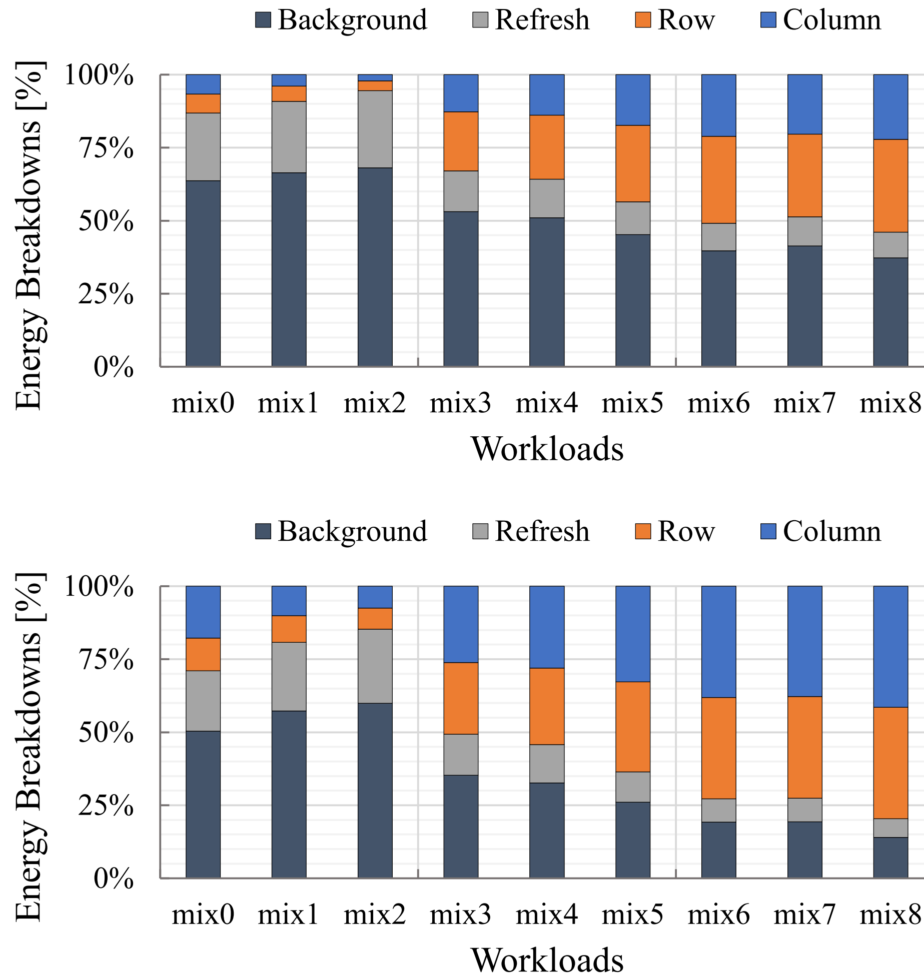


Figure 5.13: Energy breakdown of each workloads for DDR4 (top) and LPDDR4 (bottom).

LPDDR4. The overall trend is similar for both DRAMs, more background and refresh energy is consumed in low MPKI workloads while more row and column energy is consumed in high MPKI workloads. In other words, workload mix0 to mix2 can be categorized as compute intensive workloads whereas mix6 to mix8 are categorized to memory intensive workloads. Our workloads utilize from 1% to 36% of the peak DRAM bandwidth available, which we believe is representative of real use cases [84]. The effectiveness of proposed designs are reported for each workload using using a weighted speedup metric for performance [7] and an energy-per-bit metric.

5.7 Evaluation Results

The performance impact of proposed Half Page DRAM and CRR is shown in Figure 5.14. Half Page DRAM needs one additional clock cycle to activate a row, plus additional cycles spent to activate more rows when both half pages within a row are accessed simultaneously as discussed in Section 5.5.1. However, those penalty are effectively hidden when the memory controller utilizes HPLP improving weighted speed-up by 1.78% on average for DDR4 and 1.15% on average for LPDDR4. Only workload mix1 of LPDDR4 showed performance degradation of 0.17% which is negligible considering that workload is compute intensive.

CRR alone has negligible effect on performance where weighted speed-up improvement of 0.24% and 0.66% on average for DDR4 and LPDDR4 respectively is observed. Same applies for multiple CRR ($\times 8$ CRR) of DDR4 but with less than 1% performance degradation seen on compute intensive workloads. Interestingly, multiple CRR performs even better than Half Page DRAM with HPLP in memory intensive workloads for LPDDR4. This is because more DRAM requests are serviced in between $8 \cdot tREFI$ of $\times 8$ CRR causing less rows to closed and then re-opened due to issue a refresh. The penalty of closing and re-opening a row is greater in LPDDR4 compared to DDR4 making the performance improvement much more noticeable.

When both Half Page DRAM and multiple CRR are combined, the performance improvement reaches more than 5.5% on average for memory intensive workloads for both DDR4 and LPDDR4. The new parallelism introduced by HPLP benefits from the longer refresh interval time of multiple CRR resulting in the larger improvements.

Unlike performance, energy is saved regardless of the combinations of proposed designs, as shown in Figure 5.15. Half Page DRAM reduces 37% of V_{DD} power supply row energy and 4.5% of column energy by accessing only half of the page. Hence, Half Page DRAM is most effective in memory intensive workloads where there are abundant memory requests issued to the DRAMs and saves 12.3% and 13.2% of the energy on average for DDR4 and LPDDR4 respectively.

CRR and multiple CRR on the other hand, saves refresh energy which is consumed periodically every 7.8 μs for DDR4 and 3.9 μs for LPDDR4 regardless of whether the

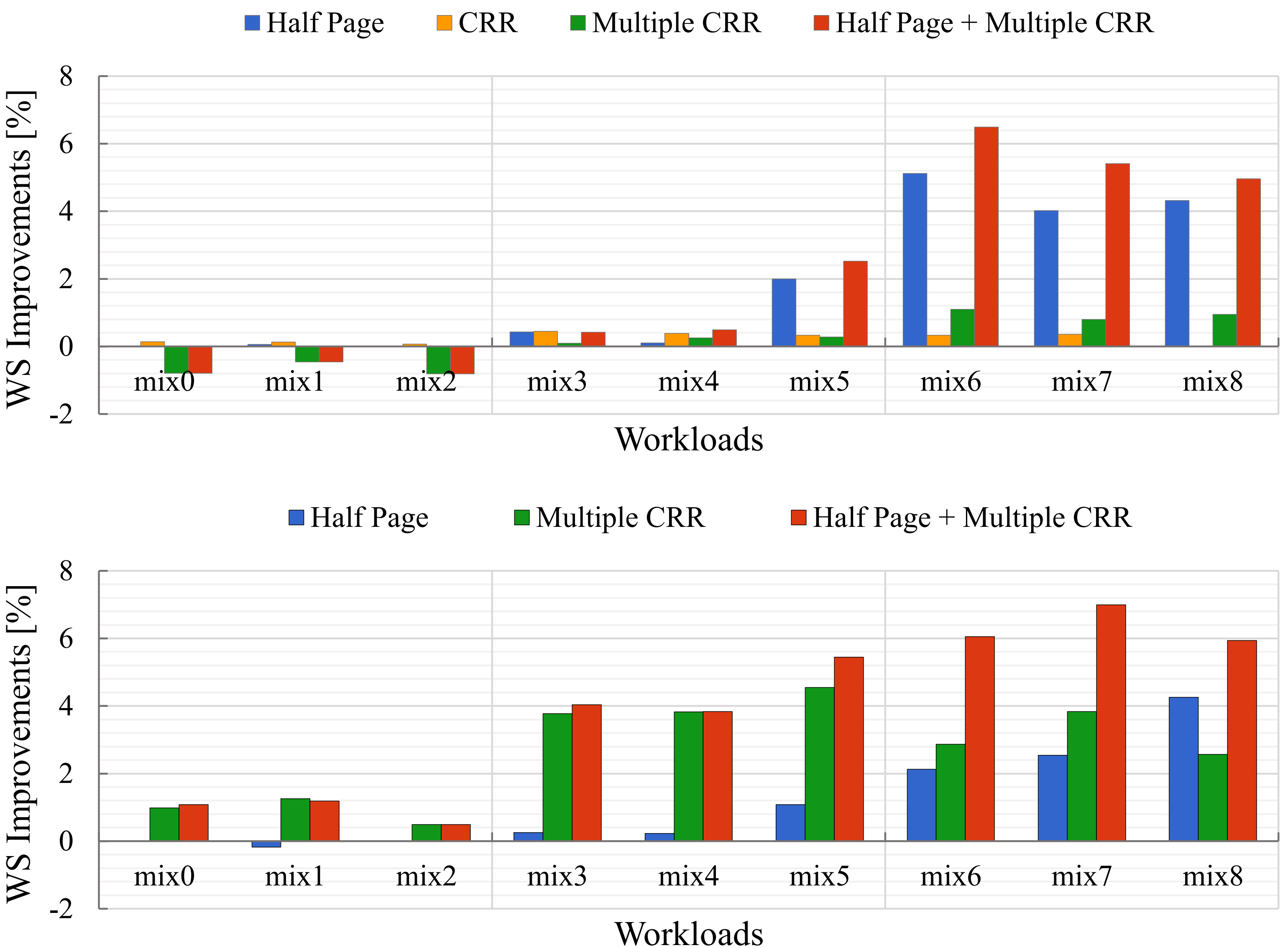


Figure 5.14: Weighted speed-up [7] improvements of proposed Half Page DRAM and CRR as well as when both schemes are combined for DDR4 (top) and LPDDR4 (bottom).

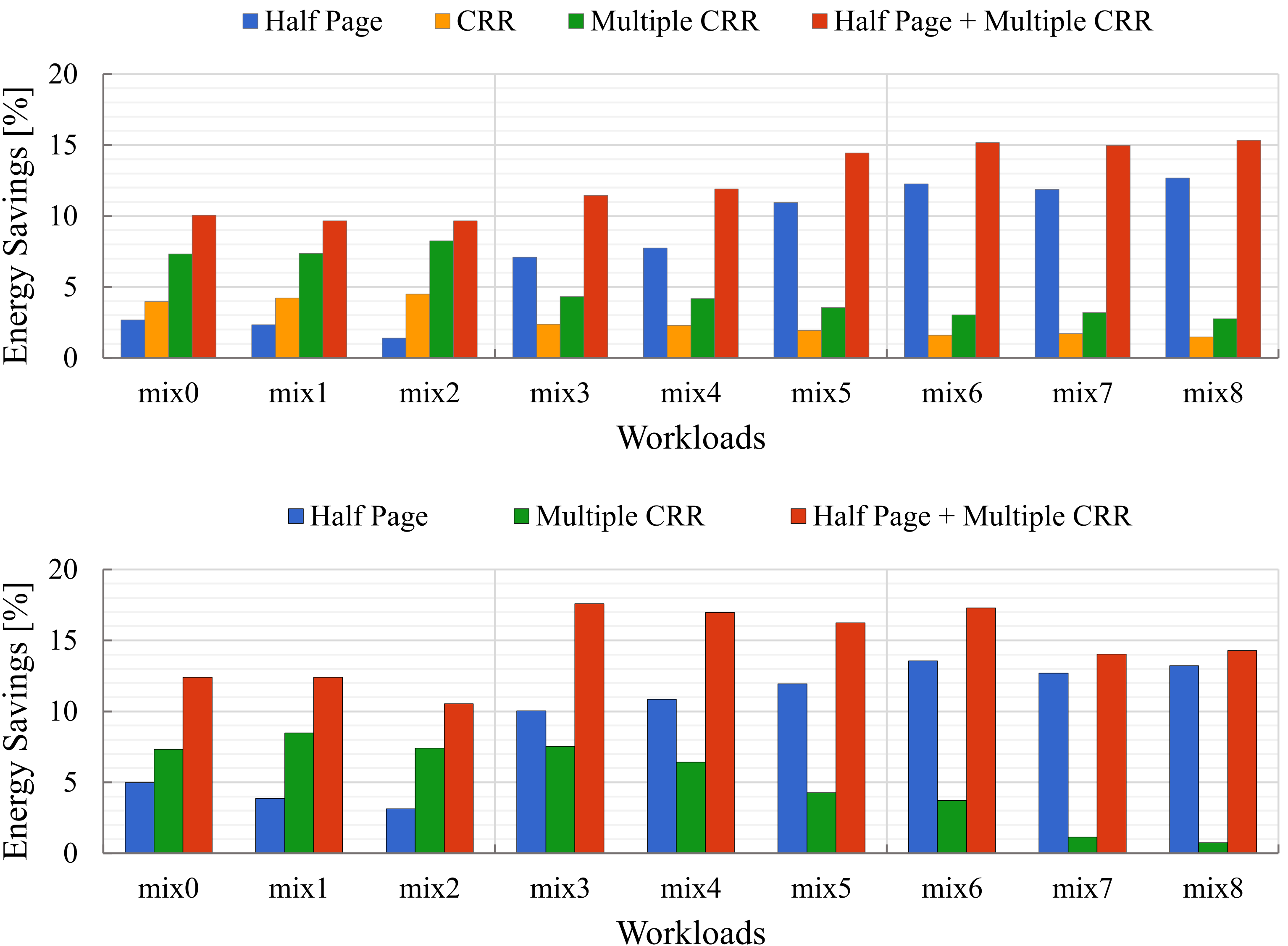


Figure 5.15: Energy/bit saved by proposed Half Page DRAM and CRR as well as when both schemes are combined for DDR4 (top) and LPDDR4 (bottom).

DRAM is being accessed or placed in low power sleep modes including self-refresh mode for LPDDR4. As a result, both CRR and multiple CRR are effective in compute intensive workloads where DRAMs are placed in low power sleep modes for long periods of time. CRR saves on average, 4.2% and 3.8% energy for DDR4 and LPDDR4 while multiple CRR saves 7.7% on average for both DDR4 and LPDDR4 during compute intensive workloads. Surprisingly, the energy saved by CRR and multiple CRR is similar for both DDR and LPDDR4 despite LPDDR4 requires twice as much refreshes to be issued retain data stored on the cells (refreshes issued every 7.8 μs for DDR4 *vs.* 3.9 μs for LPDDR4). This is because the 8GB DDR4 refreshes twice as many cells in each refresh cycle as the density is twice as large than the 4GB LPDDR4. It is notable that we assumed 85°C which is the worst case for CRR and most likely not the temperature for a system having such low bandwidth utilization workloads. As future work, we plan to study the effect of CRR in detail with temperature variation and the refresh scheduling we proposed which dynamically pulls-in refreshes depending on the workload.

Since Half Page DRAM is effective in memory intensive workloads while multiple CRR is effective in compute intensive workloads, when both designs are combined significant energy savings are observed across various workloads. On average, 12.5% and 14.6% of energy is saved for DDR4 and LPDDR4 respectively where 10% or more energy is saved in every workloads for both of the DRAMs. Hence, we can conclude that Half Page DRAM and CRR are orthogonal to each other and complements each other's shortcomings well.

5.8 Related Work

Cooper-Balis *et al.* [44] proposed to add a row division decoder to select segments of the row. Udipi et al. [42] exploited the hierarchical wordline structure and combined a subset of the column address with RX (FX in our terminology) to select a subset of sub-wordlines. But both of the designs degrade memory bandwidth significantly as finer grained row buffers are accessed.

Half-DRAM proposed by Zhang *et al.* [45] re-routed wordlines to select half of the

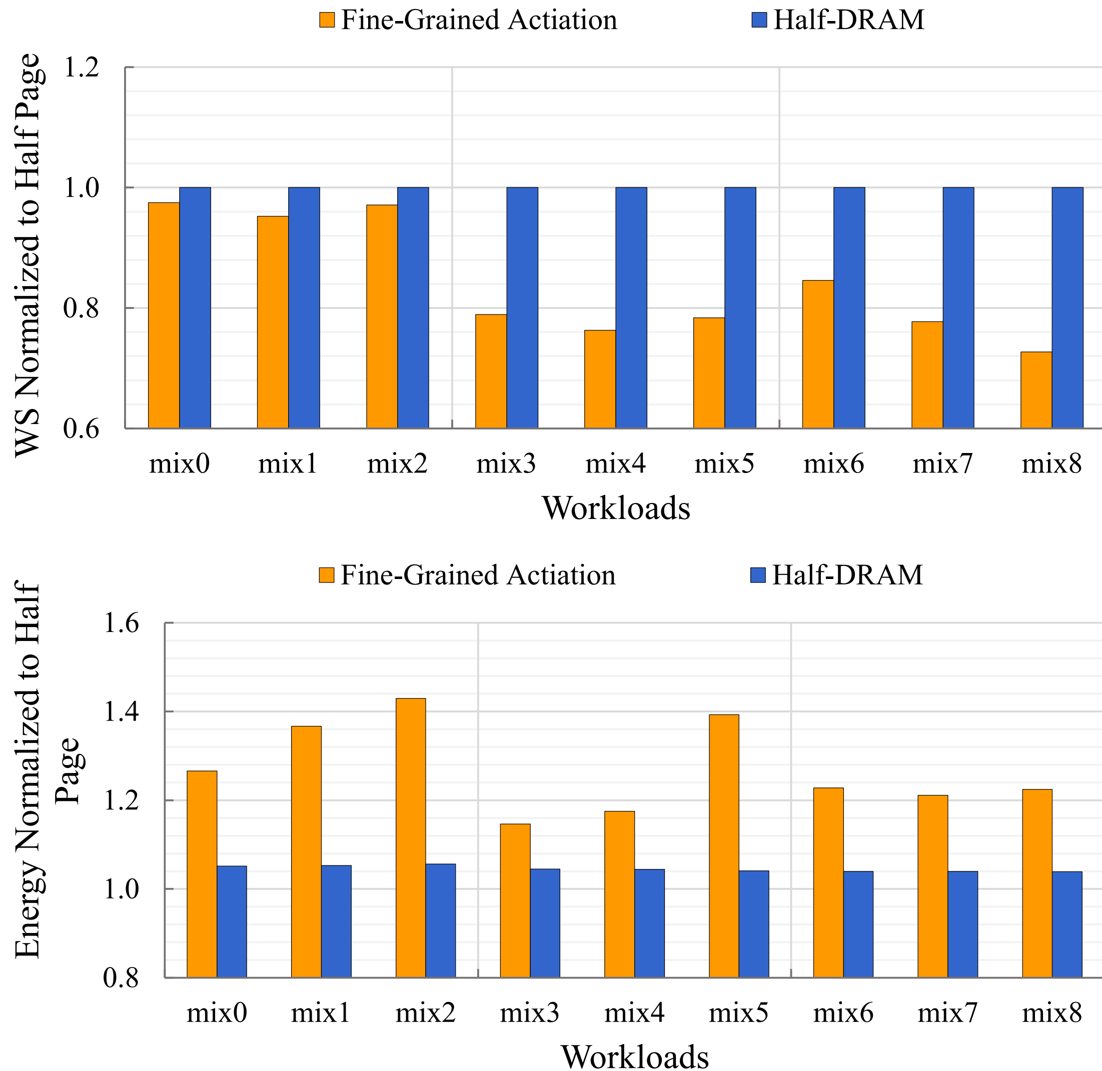


Figure 5.16: Weighted speed-up improvements (top) and energy saved (bottom) by Fine-grained activation and Half-DRAM relative to our Half Page DRAM.

cells located in one MAT and half of the cells located in another MAT, instead of every cell in a single MAT. Full bandwidth is achieved even with half page in this case because every SIO wire within a sub-array can transfer data. However, contrary to the assumptions underlying Half-DRAM, sub-wordlines of modern DRAM are routed in a staggered manner as discussed in Section 3.2.2. This makes Half-DRAM to be possible only with a large area overhead which is estimated to be up to 12% assuming that sub-wordline drivers are doubled. Our design, on the other hand, takes both

staggered sub-wordlines and data wire hierarchy of modern DRAM into account and still achieves 1.4% area overhead.

Figure 5.16 compares the performance and energy improvements of aforementioned prior work to Half Page DRAM we proposed. We assumed every designs access only half of the page on a row for each activate command. In addition, fine-grained activation assumed a 32 B cache line system as only half the data compared to other two designs are fetched in each DRAM accesses. We used DramDSE to estimate the effect of doubling the sub-wordline drivers for Half-DRAM to work and used the IDD values generated from DramDSE to run full system simulation. As can be seen from the top of Figure 5.16, fine-grained activation degrades performance by 16% on average and at most 27% due to the memory bandwidth being cut by half. The energy consumption also increases by 27% on average due to the slowdown in performance. Half-DRAM shows no difference in performance compared to our design as full bandwidth is achievable in Half-DRAM design. However, the energy consumption is increased by 5% on average compared to our design due to the increase in die size. Overall, our Half Page DRAM consumes less energy and occurs less area overhead than the prior state-of-the-art work.

The notion of recycling charge to save energy was explored heavily in the early 1990's, often in the context of adiabatic logic. During this period, Kawahara *et al.* [85] implemented Charge Recycle Refresh scheme in a test chip that consisted of two small MATs. Charges were recycled between two MATs by shorting power supply lines, which is a different approach but shares the same goals as our proposed CRR. This was the first and only proposal we found in our search of this field after creating CRR, and it was published over 20 years ago. Since that time DRAM organization has changed dramatically to deal with the exponentially growing number of bits. As a result, their approach of connecting power supplies of independent memory MATs will no long work. We instead exploit half page access granularity which is already supported by DDR4 and we allow the memory controller to flexibly schedule CRR so that refresh energy can be saved without degrading performance. The next chapter explores how even larger reductions in refresh energy are possible by exploiting the distribution of retention times of DRAM cells.

Chapter 6

Smart Refresh

DRAM technology node scaling has been slowing down recently and the state-of-the-art DRAM is fabricated using the 1y nm process technology. To continue scaling the technology node, DRAM cell area has to shrink along with the storage node capacitance. This leads to smaller read signal which, coupled with increased coupling capacitance in the scaled cells, reduces the overall reliability. DRAM manufacturers are increasing the number of internal rows refreshed during each refresh cycle to improve cell reliability as described in Section 5.1.2. This trend is likely to continue, making the energy consumption and performance impact of refresh ever so important.

Current refresh specification is set based on the small fraction of cells that leaks the most, known as the *weak cells*. The majority of other cells will retain data much longer than the weak cells even when refreshed at a rate much lower than that specified in the datasheet. This observation resulted prior work that selectively refreshes only the weak cells frequently but the rest infrequently as described in Section 5.8, a great solution in reducing the overheads of refresh.

In this chapter, we build upon prior work that exploits the retention time characteristics of DRAM cells and propose a scheme that can work not only in auto-refresh but also in self-refresh where refresh energy is dominant. To the best of our knowledge, this is the first that works in both situations.

6.1 Cell Retention Time Characteristic

This section, first reviews various leakage mechanisms of the DRAM cells and their sensitivity to the temperature. It also discusses how DRAM minimizes the overall leakage to retain data as long as possible based on the temperature sensitivity of each leakage mechanisms. Then, it presents our data on the retention time distribution of DRAM cells operating at various temperature and data patterns using Field Programmable Gate Array (FPGA).

6.1.1 Leakage Mechanisms of DRAM Cell

Retention time of DRAM is determined by the leakage current of DRAM cells. Among many leakage mechanisms, sub-threshold leakage, junction leakage, and Gate Induced Drain Leakage (GIDL) are known to be the dominating leakage mechanisms of DRAM cells.

$$I_{sub} = \mu_{eff} \cdot C_{ox} \cdot \frac{W}{L} \cdot (m-1) \cdot \left(\frac{k_B T}{q}\right)^2 \cdot e^{\frac{q(V_{gs}-V_t)}{mk_B T}} \cdot \left(1 - e^{\frac{qV_{ds}}{k_B T}}\right) \cong I_{sub} \cdot e^{\frac{q(V_{gs}-V_t)}{mk_B T}} \quad (6.1)$$

Sub-threshold leakage is sensitive to $V_{gs} - V_t$ as shown in Equation 6.1 [86]. As temperature gets hot, threshold voltage (V_t) reduces which increases sub-threshold leakage.

$$I_{G-R} = A \cdot J_{G-R} \cdot \left(e^{\frac{qV_{bias}}{2k_B T}} - 1\right) \quad (6.2)$$

$$\textit{where } A = \textit{junction area and } J_{G-R} = \textit{current density}$$

Shockley-Read-Hall (SRH) recombination shown in Equation 6.2 is reported to be the most noticeable junction leakage component in DRAM cells [86, 87], which is sensitive to the area of the source/drain diffusion and the current density that depends on the depletion layer width. Except for the small fraction of cells on the DRAM that have short retention time, also known as *weak cells*, the retention time

of the cells are reported to be determined by the junction leakage [87].

$$I_{GIDL} = a \cdot E_s \cdot e^{\frac{-b}{E_s}} \tag{6.3}$$
$$where\ a = constant,\ b = 21.3\, MV/cm\ and\ E_s = \frac{V_{dg} - 1.2}{3t_{ox}}$$

Intermediate energy states (traps) caused by defects in the gate oxide and substrate interface are known to be the source of Trap-Assisted-Tunneling GIDL shown in Equation 6.3 [86]. Very few DRAM cells have excessive TAT GIDL which are classified as weak cells. The frequency in which refresh should be issued by the memory controller to guarantee error-free operation is determined by the small fraction of the weak cells that have higher GIDL than many other cells.

Temperature sensitivity of the leakage mechanisms described above differs from each other.

$$t_{REF} \propto e^{\frac{E_a}{k_B T}} \tag{6.4}$$
$$where\ t_{REF} : retention\ time\ and\ E_a : activation\ energy$$

Activation energy, which is a measure of temperature sensitivity, can be derived for each leakage mechanisms using the measured retention time of individual cells that have one of the aforementioned leakage mechanisms as dominant and applying Equation 6.4. Activation energy of each leakage mechanisms are reported to be 0.3 eV for GIDL, 0.6 eV for junction leakage, and 0.8 eV for sub-threshold leakage [87]. Thus, GIDL is least sensitive to temperature whereas sub-threshold leakage is most sensitive to temperature. The sum of the leakage currents are minimized in standard commodity DRAM by modulating gate off voltage (V_{BBW}) and bulk bias (V_{BB}) of the access transistor as temperature changes [88]. As temperature increases, V_{BBW} and V_{BB} becomes more negative to suppress sub-threshold leakage, whereas at cold temperatures, V_{BBW} and V_{BB} are smaller in absolute value to suppress junction leakage and GIDL. It is also notable that overall leakage reduce as temperature gets colder.

6.1.2 Pause Refresh Test

This thesis work used the well-known pause refresh test method to measure the retention time of the cells. As shown in Figure 6.1, the pause refresh test routine starts by writing new data to every row and column of the DRAM. Refreshes are issued periodically while the writes are being issued to retain data that has been written to the cells. Once every cell has updated its value, 8,192 refreshes are issued followed by a period of no refreshes, t_{Pause}. Then, 8,192 refreshes are issued once again and the data stored on entire DRAM cells are read and compared with the data originally written to the cells to see if there are any failures. The retention time of the cell in this test routine will be $t_{ret} = 8,192 \cdot t_{REFI} + t_{Pause}$ given that the refresh interval of the refreshes issued after a series of writes is $\mathrm{t_{REFI}}$.

Characterization of DDR3 DRAM cell's retention time was performed using Xilinx Artix-7 [89] and Zynq SoC [90] FPGA at three different temperatures. The pause refresh test routine described earlier was implemented in FPGA as shown in Figure 6.2. The traffic generator issues write and read commands to every row and column with Pseudo Random Bit Sequence (PRBS) data. The time refreshes are paused, t_{Pause}, is controlled by the pause timer that precisely counts the time from the start of the periodic refreshes issued after the write sequence. The refreshes are issued by the memory controller and $\mathrm{t_{REFI}}$ is set to 6.8 μs instead of the specification value, 7.8 μs, to guarantee that only t_{Pause} limits the retention time of the cells under test. The range of t_{Pause} were from 100 ms to 8,192 s with 100 ms interval from 100 ms to 1 s and power of 2's interval from thereafter. The FPGA was put into the chamber with a precision of ±1°C to change the ambient temperature of the design under test (DUT). Cumulative fail bits per t_{Pause} and per temperature were counted and stored on the FIFO. Once the test is complete, the results were read from the FIFO and displayed on the host computer's terminal by transferring the results via Universal Asynchronous Receiver-Transmitter (UART) protocol.

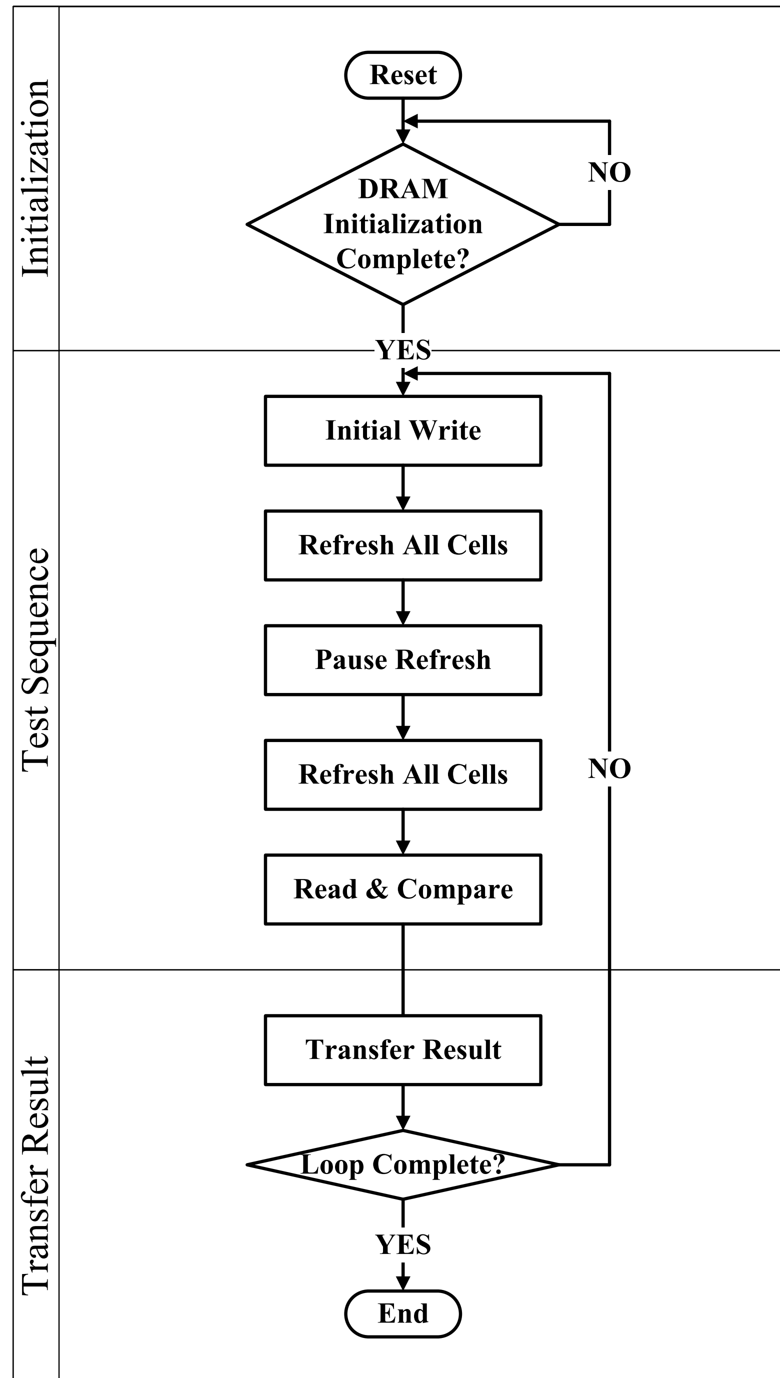


Figure 6.1: Flow chart of the pause refresh test routine used to characterize DRAM cell retention time. The pause refresh time will increment in every test routine and the retention time of the cells is the last pause refresh time without an error.

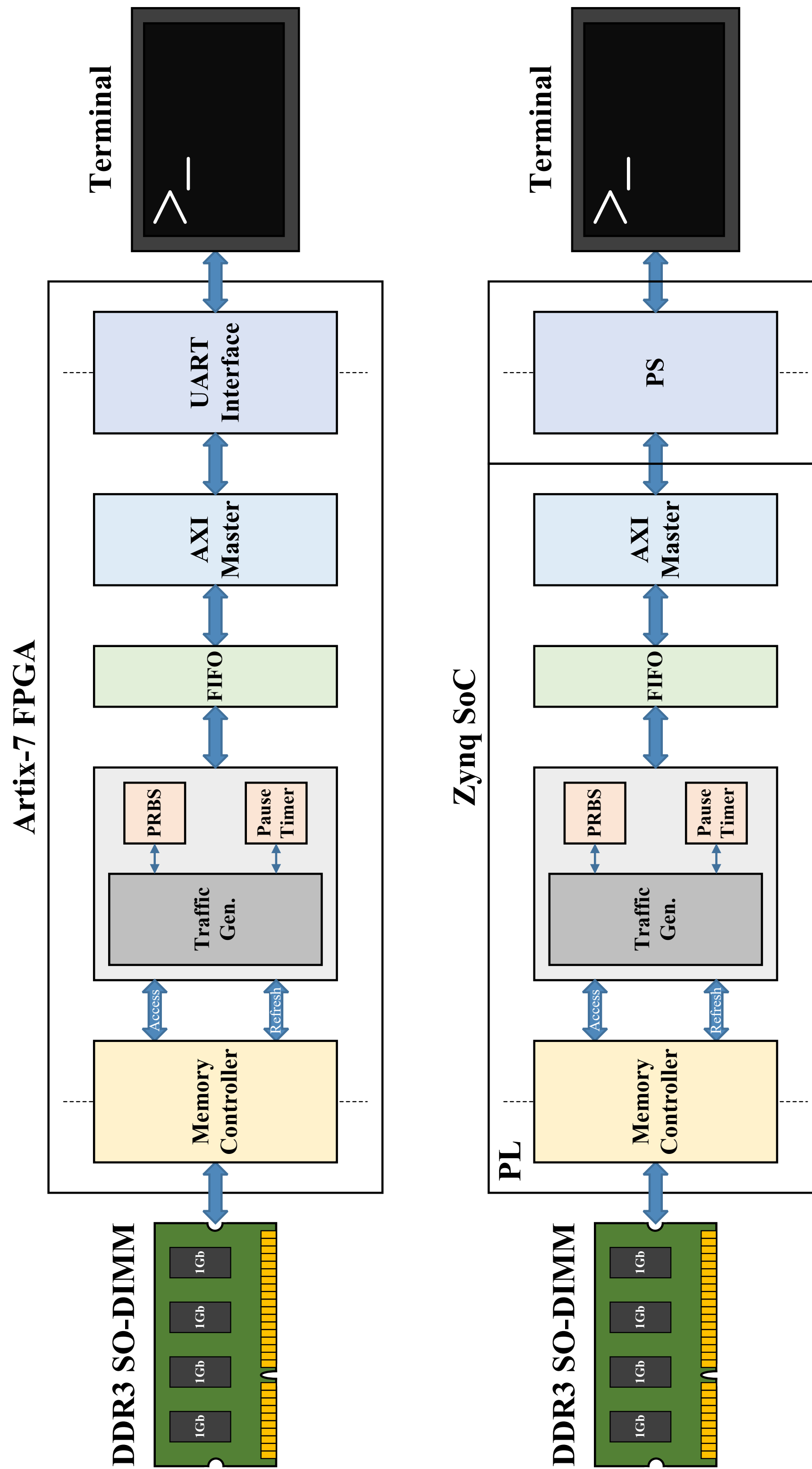


Figure 6.2: Block diagram of the pause refresh test routine shown in Figure 6.1 implemented in Xilinx Artix-7 and Zynq SoC FPGAs. Traffic generator schedules write and read commands based on the test routine and stores the test results to the FIFO. Then, the content of the FIFO is read and transferred to the host computer using UART to be displayed on the terminal.

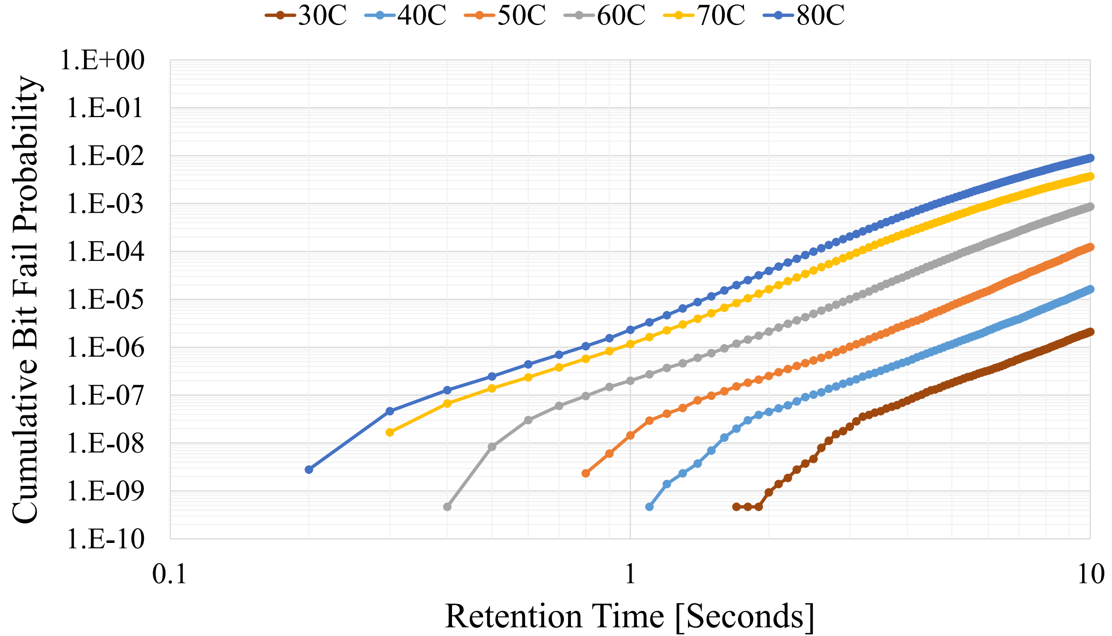


Figure 6.3: Cumulative bit failure probabilities with t_{ret} from 100 ms to 10 s and ambient temperatures from 30°C to 80°C.

6.1.3 Retention Time Distribution

The retention time characterization result gathered using FPGA is shown in Figure 6.3. To better illustrate the weak cell distribution, we only present the cumulative bit fail probabilities with t_{ret} from 100 ms to 10 s. A break point connecting two distinct characteristic lines with different slopes, one being steeper than the other, is observed in Figure 6.3. The two characteristic lines are often called the *tail* and the *main* distribution [91], and are better distinguished in lower temperatures ($\leq$ 60°C) for the t_{ret} range presented in Figure 6.3. The tail distribution is located on the lower bit fail probability quadrant and hence, represents the weak cells. The cumulative bit fail probability of the weak cells or the break point of the tail and main distribution is roughly the same across different temperatures. The main distribution represents other majority of the cells on the DRAM and unlike the weak cells, junction leakage, that has a moderate sensitivity to temperature is the dominant leakage mechanism on these cells.

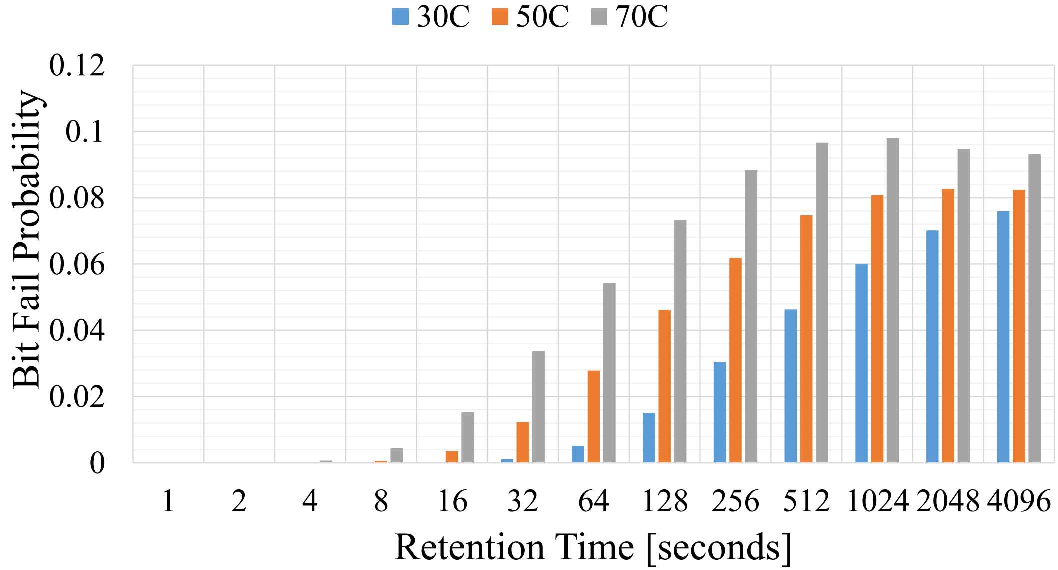


Figure 6.4: Probability distribution of the bit failures with t_{ret} from 1 s to 8,192 s and ambient temperatures of 30°C, 50°C and 70°C.

Figure 6.4 shows the probability distribution of the bit failures which was calculated by finding the difference of measured cumulative distribution results discussed earlier for each of the retention time bins. A Poisson distribution function with log-normal scale best represents the Probability Density Function (PDF) of the distributions shown in Figure 6.4. The mean and variance of the probability distribution or λ of the Poission distribution function gets larger as the temperature gets colder. Less cells fail as the temperature gets colder by having longer retention time and interestingly, majority of the cells retain data longer than 10 seconds which is more than 150$\times$ of the refresh specification, 64 ms, regardless of the temperature.

Refreshes are issued in row granularity not cell by cell. Table 6.1 shows the number of failed rows when refreshes are issued at 2$\times$ the refresh interval specified by the datasheet. We show the result of four different DDR3 SO-DIMM parts from two different DRAM manufacturers in Table 6.1. The column on the far right labeled 'DIMM' shows the total number of rows that failed on the SO-DIMM. There are 128 K rows on each of the parts we tested, resulting in only 0.6% of the rows failing when the refresh frequency is cut by half, even for Part0 that showed the most failed

	Chip0	Chip1	Chip2	Chip3	Chip4	Chip5	Chip6	Chip7	DIMM
Part0	153	153	129	72	78	77	63	45	769
Part1	83	112	95	100	92	76	77	60	694
Part2	3	6	10	2	5	6	6	4	42
Part3	10	3	3	13	9	4	5	9	56

Table 6.1: Number of failed rows in 4 different 1GB DDR3 SO-DIMM parts and each of the chips on the DIMM when refreshes are issued at $2 \cdot t_{REFI}$.

rows. Table 6.1 also shows the failed rows for each of the eight DRAM chips that form the SO-DIMM. Interestingly, the sum of the failed rows on each of the chips of a DIMM is roughly the same as the total number of failed rows on a DIMM. In other words, the location of the weak cell(s) or the failed rows are completely random and are uncorrelated between different chips on the same DIMM. This is to the best of our knowledge the first work to uncover such observation.

	Chip0	Chip1	Chip2	Chip3	Chip4	Chip5	Chip6	Chip7	DIMM
Part0	720	743	714	415	461	423	339	216	4083
Part1	409	578	454	505	519	376	421	351	3565

Table 6.2: Number of failed rows for the first two DDR3 SO-DIMM of Table 6.1. The refreshes are issued at $4 \cdot t_{REFI}$ and only the results of the first two parts are shown.

Since the number of failed rows are small, we took the bad parts and further looked what happens if we refresh every cells one fourth the rate. The trend is not that different when the refreshes are issued at 4× the normal refresh interval which is shown in Table 6.2. Only 1.15% and 1.35% of the failed rows on a DIMM overlap with more than 2 chips. Hence, if each chip on a DIMM can be refreshed independently only total of 750 rows and 578 rows, not 4,083 and 3,565 for Part0 and Part1 respectively, have to be refreshed every 64 ms while the rest of the rows can be refreshed every 256 ms and still guarantee that the data will be retained. This observation is not new. A number of other researchers have proposed extending the refresh rate, and the next section describes this work.

6.2 Prior Work

The prior work on extending the refresh interval can be categorized by how they treat the small fraction of weak cells that can no longer retain data. RAPID proposed by Venkatesan *et al.* [92] and RIO proposed by Baek *et al.* [93] propose to not use the weak cells as they are small in volume. The other main approach, exemplified by RAIDR proposed by Liu *et al.* [94], REFLEX proposed by Bhati *et al.* [95], and scheme proposed by Gong *et al.* [96], propose to refresh the weak cells more frequently than other cells. This work characterize the retention time of the cells every time the system boots up and sometimes even on-the-fly during run-time to compensate for data pattern dependency and variable retention time (VRT) [97, 98, 99] characteristic of the cells.

In both approaches, the rows that contain weak cells are stored on the memory controller hence, these work are not applicable to self-refresh where the refresh energy is most significant. Another huge drawback of the prior work is that the chance of refresh failure still exists even though the memory controller issues refresh to the rows with weak cells more often. This is because the effective retention time of the cells might be shorter than the measured retention time due to more rows being refreshed in each refresh cycle especially in DRAMs fabricated with recent process technology nodes as mentioned in Section 5.1.2. Also, as the previous section showed, the retention time characteristic of the cells for each of the chips on a DIMM differ greatly from each other making DIMM based refresh control inefficient. The next section shows how moving some of the refresh control to the DRAM can avoid these issues.

6.3 Proposed Design

We propose *smart refresh* which extends prior work on reducing number of refresh operations issued based on the retention time characteristics of DRAM. The goal of smart refresh is to simplify the communication protocols between the memory

controller and DRAM. This minimizes the customization needed to the memory controllers and allow computer systems to utilize the smart refresh feature by just plugging in the new revised DIMM. We consider two additional characteristics of modern DRAM that prior work ignored, to improve the practicality of the already excellent idea: 1) more rows are refreshed in each refresh cycle of modern DRAMs, and 2) multiple chips that form a rank have different retention time characteristic hence, rows with weak cells are not the same between different chips that are accessed simultaneously. The novelty of smart refresh comes from DRAM being aware of the weak cell distributions and its location, not the memory controller. In smart refresh, memory controller issues refresh periodically at a given rate without any context of the retention time characteristic distributions, while DRAM decides internally whether to selectively refresh weak cells in addition to the cells to be refreshed periodically. Since DRAM has full control of the refreshes to the weak cells, smart refresh is capable of reducing self-refresh energy unlike prior work. This section describes how to store the weak cell information effectively, especially in terms of cost, inside the DRAM.

6.3.1 Storing Weak Cell Information Inside the DRAM

DRAM uses fuses to store remapping of faulty cells to the functional redundant cells. After a series of test to identify faulty cells and functional redundant cells, the fuses on DRAM are either cut using a laser (metal fuse) or ruptured to break the oxide (anti-fuse or e-fuse) to change its binary value. There are typically 8 redundant wordlines and 4 redundant CSLs (16 redundant bitlines) in a 512 wordlines by 512 bitlines MAT, which is 1.56% and 3.12% additional wordlines and biltines being added for repair purposes. Hence, recent DRAMs use an array of anti-fuses to store both row and column remapping information and achieves a small area overhead of 0.5% for the 18 nm 8Gb DDR4 [100]. The side effect of forming an array is the longer time needed to read the data in the array as the RC loading of the rows and columns in the anti-fuses increases dramatically. As the information stored on the anti-fuses are read-only, meaning the data are not to be changed at any time, DRAM spends a long time during power-up sequence to read the entire contents stored on the array

anti-fuse *once* and stores them in a latch or a small SRAM within a DRAM. This is the reason why t_{INIT3}, one of the timing parameters after reset during power-up sequence, increased 10× from 200 μs of LPDDR3 [23] to 2 ms of LPDDR4 [24].

Similar to how the remapping information is stored, we propose to add another set of anti-fuses to store the weak cell information. More specifically, we propose to store the *refresh counter values*[1] that contains one or more weak cells. DRAM issues 8,196 refreshes within t_{REFW} of either 64 ms or 32 ms depending on the type of the DRAM. Therefore, 13 bit should be sufficient enough to represent a unique refresh counter value and one leading bit will be added to indicate whether this value is valid or not, resulting in a total of 14 bit for an entry of weak cell information. The latch or SRAM that stores the refresh counter value of the weak cells after DRAM has been powered-up, acts like a circular buffer where the read pointer advances sequentially every time a refresh has been issued either by the memory controller or the DRAM itself. Refreshes to weak cells will be performed or skipped depending on the leading valid bit status of the current entry.

Smart refresh also use a global refresh interval of $4 \cdot t_{REFI}$ to retain data stored on majority of the cells. However, unlike prior work, smart refresh scheme does not require anything else to be issued from the memory controller. More details about the memory controller side will be discussed in detail in Section 6.4. Table 6.2 showed the number of rows with weak cells for each of the chip on a DIMM when refreshes are issued at $4 \cdot t_{REFI}$. The number of weak cell rows are found to be 2.7% - 3.1% of the total number of rows on a DIMM which is on par with other retention time characterization results [93]. This is 2× to 3× more than the number of redundancy rows and the area overhead to store all these information will be close to the order of the entire set of anti-fuses currently on DRAM. We can do better by exploiting the fail rate per chip-level. The maximum fail rate measured between the chips on a DIMM is 0.4% - 0.6%, which $\frac{1}{5}$ of the fail rate of a DIMM. Hence, if each DRAM chip stores the weak cell information separately, not only does the area overhead becomes negligible (0.1%), it also reduces the number of refreshes that has to be issued to

[1] DRAM keeps track of which row(s) to refresh by sequentially incrementing a counter value from 0 to 8,195. This counter will increment its value whenever a refresh command is issued explicitly by the memory controller or when DRAM refreshes cells during self-refresh mode.

the weak cells apart from the global refresh by 80%, making our smart refresh both practical and effective DRAM energy reduction scheme. For a simple comparison, one of the prior work REFLEX had to issue 1,966 additional refreshes while smart refresh only needs 327 additional refreshes on top of the 2,048 refreshes every 32 ms for 8Gb LPDDR4. Of course, the numbers might grow larger, when one accounts for the cells that the DRAM are already refreshing using "hidden" refresh cycles.

6.3.2 Refreshes to Weak Cells

The previous section mentioned that the memory controller does not have to explicitly issue refreshes other than the 2,048 refreshes that are issued every 32 ms for LPDDR4. Then, how are 327 additional refreshes that have to be issued to retain data stored on weak cells done in smart refresh? We propose to distribute those refreshes across the 2,048 refreshes and perform the refreshes to weak cells right after the global refresh. Since the number of weak cell rows are well below the refreshes being issued by the memory controller, the refreshes to weak cell rows can fit-in to the main refreshes without any additional refresh commands. Of course, this increases the refresh cycle time (t_{RFC}) or the time spent on refresh but because the gap between the refreshes are longer than before, its impact to performance will be negligible as will be shown in Section 6.5.

We leave the scheduling of the weak cell refreshes completely to the DRAM, allowing the DRAM to decide how frequently to refresh weak cell rows, trading off robustness for energy efficiency. This flexibility is a benefit when DRAM refreshes more rows than the number specified by the datasheet, which is already being done in modern DRAMs, as was discussed in Section 5.1.2. Figure 5.2 showed 2× more rows are being refreshed in each refresh operations, which means a particular row is being refreshed 2× more often. In our smart refresh scheme, DRAMs can schedule those weak cell rows that require 2× more often refreshes by using two of the global refresh slots instead of one for that particular row and evenly spacing those two refreshes within the 2,048 global refreshes. Hence, by leaving the scheduling of the weak cell refreshes to DRAM, our smart refresh scheme can guarantee error free operation of

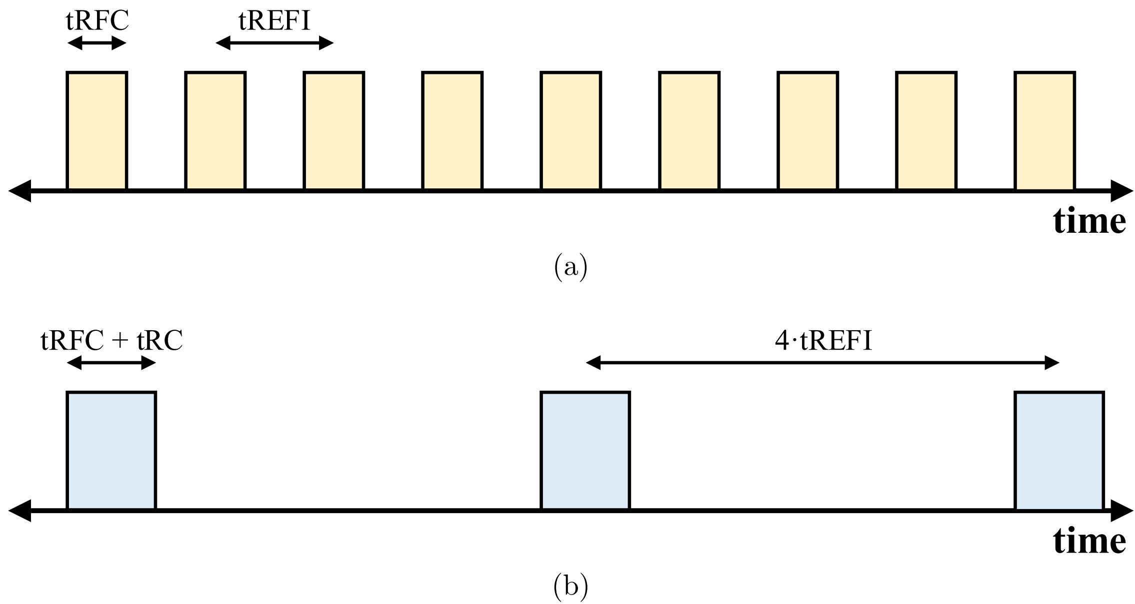


Figure 6.5: Refresh scheduling done by the memory controller for (a) conventional DRAM and (b) DRAM with smart refresh scheme.

DRAM without exposing those confidential information. Moreover, we leave room for the DRAM manufacturer to further improve the yield by allowing some of the dies that were failing due to short retention time to be converted to good dies by issuing refreshes more frequently, if there are left over global refresh slots for those additional refreshes to be slotted in.

6.4 Memory Controller Support

There are no changes done to the memory controller interface to support smart refresh, if the refresh cycle time ($\mathrm{t_{RFC}}$) and refresh interval ($\mathrm{t_{REFI}}$) are adjustable, which is usually the case. These values are usually programmable by software. Figure 6.5 compares the refresh scheduling done by the memory controller for the conventional DRAM and the new DRAM with smart refresh implemented. Since smart refresh uses $4 \cdot t_{REFI}$ as the global refresh rate, only $\frac{1}{4}$ the number of refreshes issued to conventional DRAM are issued. Because many rows are being refreshed in each global refresh operation, the refreshes to the weak cells can be done in conjunction

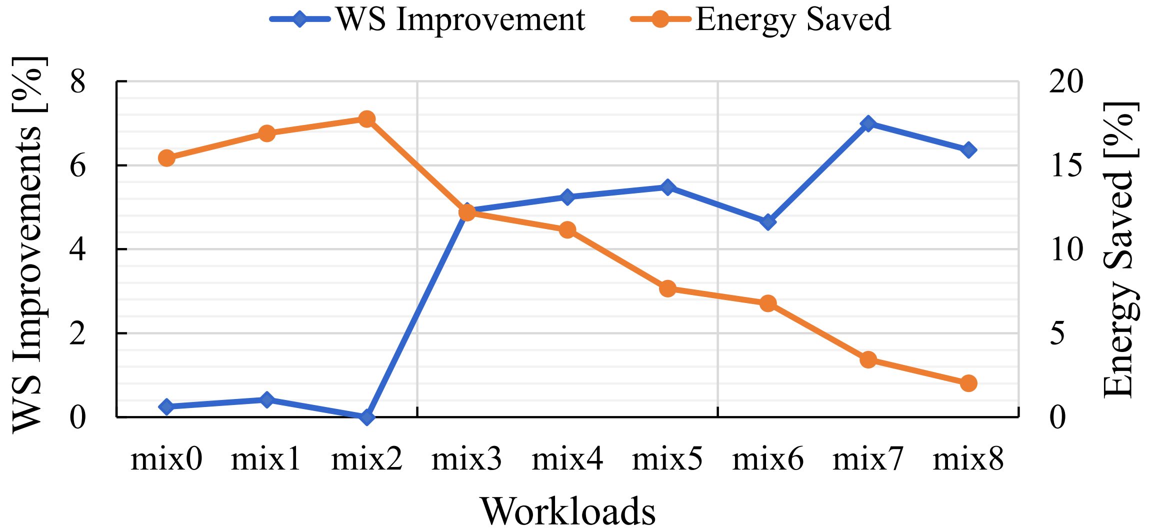


Figure 6.6: Weighted speed-up improvements (left axis) and energy saved (right axis) by proposed smart refresh for various workloads.

with global refresh operation by exploiting the bank parallelism. To be conservative, we penalize ourselves and wait one complete row cycle or t_{RC}, the time needed to refresh a row, to each refresh issued by the memory controller.

6.5 Evaluation Results

This section presents the potential benefits of smart refresh using the same system configuration and workload setup described in Section 5.6. We only use LPDDR4 DRAM that enters self-refresh mode to show the full potential of smart refresh. To be conservative, we assumed that refreshes to weak cell rows are issued whenever a global refresh is issued.

Figure 6.6 shows the performance and energy improvements achieved by using smart refresh. One of the disadvantage of smart refresh was the longer refresh cycle time (t_{RFC}) that is increased to accommodate additional refreshes to weak cell row(s). Data access to the DRAM is completely blocked during t_{RFC} hence, this was potentially a performance degradation factor especially for memory intensive workloads.

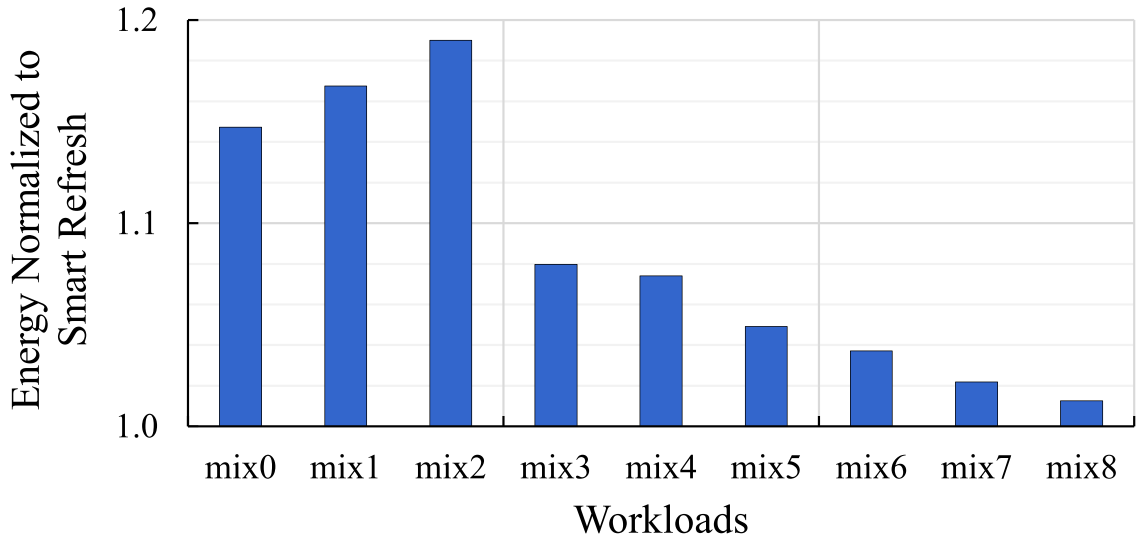


Figure 6.7: Energy saved by REFLEX and Multiple CRR relative to smart refresh.

Contrary to our initial belief, smart refresh improves performance and the most improvements are observed in memory intensive workloads where we see an average of 6% weighted speed-up improvements. This improvement is observed because smart refresh issues 75% less refreshes. The increase in t_{RFC} due to weak cell row refreshes is much less than the t_{RFC} itself, so in net, data access to DRAM is blocked less than before explaining why more performance improvement is shown in memory intensive workloads. The longer intervals between the refreshes also helps in improving performance similar to the multiple CRR case.

Unlike performance improvements, energy consumption saved by smart refresh is largest in compute intensive workloads and the smallest in memory intensive workloads. This was predicted as self-refresh mode that saves the most refresh energy is entered more frequently and for a longer duration when there are less memory accesses. On average, 10.4% energy is saved for the entire workloads we evaluated and saving up to 16.7% on average for compute intensive workloads.

Figure 6.7 compares the energy saved by REFLEX to smart refresh. REFLEX also exploits the retention time distribution of the DRAM cells similar to smart refresh. However, REFLEX scheme cannot be applied during self-refresh resulting in up to 19% more energy consumption in compute intensive workloads. Moreover, REFLEX

characterize the retention time distribution of the cell in the DIMM level not the chip level issuing much more refreshes to the weak cell rows resulting in on average of 9% more energy consumption.

CRR presented in the previous section saves refresh energy regardless of the cell retention time distribution by adding a switch to recycle charge from one half page to the other half page. Hence, CRR and smart refresh is completely orthogonal to each other and more energy can be saved by combining both schemes together. On average, 12% energy is saved for the entire workloads and 18% energy is saved for compute intensive workloads. CRR saves 30% of the refresh energy and when both CRR and smart refresh are taken together, we have reduced the refresh energy by over 7×. However, because smart refresh already reduced the number of refresh operations by 4×, CRR's effect on total energy is modest. The effectiveness of the combined scheme will vary depending on how important refresh energy is. Although combined scheme saved only 18% of energy in compute intensive workloads of current DRAMs, it will be more important in the future as technology node scales and more cells are refreshed.

Chapter 7

Conclusions

This thesis focused on improving the energy efficiency of DRAM by customizing the way DRAM is currently built, based on better understanding of the DRAM itself. The optimization of DRAM process technology and circuit design are centered on minimizing the cost-per-bit-metric, which causes many resources within the DRAM to be shared in non-obvious ways. Thus, proper understanding of how DRAM is designed, is essential for researchers interested in optimizing memory performance. To help researchers better understand and deal with these issues, this thesis built a new DRAM modeling framework with former and current DRAM designers. The modeling framework specifically focused on the design constraints that are present on modern DRAM cell array, which have been misunderstood or neglected in recent memory architecture research. The impact of those constraints to DRAM studies was demonstrated using an executable version of the new modeling framework, named DramDSE. This thesis work also used DramDSE to create public models of LPDDR4 and HBM DRAMs for the first time. The area and power consumption results of those DRAMs generated by DramDSE not only validated the correctness of the model itself but also was used to explore the design space of DRAM. This exploration lead to three approaches to reduce DRAM energy without introducing large area overheads.

The first two designs, Half Page DRAM and CRR, exploited half page row access granularity to reduce the row buffer overfetch cost and refresh energy consumption. Both of the designs were built on top of a new sub-array structure that re-organized

the existing half page architecture of DDR4. Half Page DRAM added SIO wires and re-routed CSL wires to transfer twice as much data from each MAT while CRR added switches that short the power supply of two MATs in different half pages to transfer charges from one half page to the other half page. The changes were done carefully to introduce less than 1.5% area overhead for each Half Page DRAM and CRR. Evaluations performed on a typical multi-core system running various workloads, showed that Half Page DRAM is an effective solution in reducing the row buffer overfetch cost, where both row and column energy are reduced while slightly improving the performance. CRR had negligible impact on performance but saved refresh energy for both auto- and self-refresh. It was also shown that CRR could save even more refresh energy by pulling-in refreshes to recycle charges to more rows. The inefficiency both Half Page DRAM and CRR tries to solve gets worse as the feature size of the DRAM cell gets smaller. Hence, proposed designs should provide a new approach in solving issues posed by todays DRAM, such as scalability and row hammer [101], as both designs provide more benefits as process technology node continues to scale.

The last approach smart refresh, saves even more refresh energy by using the retention time distribution of the DRAM cells to reduce average refresh operation frequency. Our characterization results showed that refreshes issued to DRAM are tailored to retain data stored on small fraction of the cells called the weak cells. Since the data also shows that the distribution of the weak cells are independent between different chips on the same DIMM, protecting weak cells on a per chip basis reduces the number of weak cell rows that need to be refreshed more often by 5× over a DIMM based approach. Smart refresh adds a set of anti-fuses that stores the weak cell information on each DRAM chip, using a structure similar to that used for redundant cells. The area overhead for this structure is very small. The main benefit of smart refresh compared to prior work is its capability to reduce refresh energy even during self-refresh mode where refresh energy is most significant, since DRAM knows the locations of the weak cells. Similar to Half Page DRAM and CRR, the benefits of smart refresh are expected to be greater as technology node continues to scale, since the DRAM manufacturers can improve the reliability and the cost-per-bit by

utilizing the anti-fuses added for smart refresh.

DRAM's low cost-per-bit enabled it to be popular in wide range of computer systems. The unique structure, where every resources within the DRAM is shared, helped in achieving such low cost, but at the same time introduced certain design constraints that resulted in changing even a small fraction of the DRAM internal circuitry difficult without paying a huge overhead. With the breakdown of Denard's scaling, optimizing the energy efficiency of DRAM became one of the key component in computer system designs. This thesis addressed this issue with a focus on reducing the refresh energy, which is not scaling down at a pace with other components in DRAM as technology node shrinks. Among the two proposals to reduce refresh energy, CRR proposed to recycle charge from fully refreshed cells to the cells that are to be refreshed while smart refresh proposed to exploit the characteristic of the cells with the support of additional storage within the DRAM. We were able to reduce the refresh energy by 7× by combining both CRR and smart refresh, while minimizing the area overhead caused by the changes in DRAM's internal circuitry to be less than 2%. Our evaluations showed these schemes saved 18% of the total DRAM energy in compute intensive workloads and that the savings will be greater as DRAM technology node continues to scale and the density-per-die increases. We believe that in the future, solely customizing the internal circuitry of DRAM will not provide enough gains to outperform the overheads, unless how DRAM is built completely changes. This thesis work on DRAM modeling framework is expected to contribute in exploring such design space.

Appendix A

Acronyms and Definitions

Symbol	**Definition**
DRAM	Dynamic Random Access Memory
DDR	Double Data Rate
LPDDR	Low Power Double Data Rate
HBM	High Bandwidth Memory
WL	wordline
BL	bitline
MWL	main-wordline
FX	pre-decoded row address
SWL	sub-wordline selected by MWL and FX
BLSA	bitline sense-amplifier
SWD	sub-wordline decoder
SIO	segmented IO that routes on top of the BLSA
LIO	local IO that transfers data on the SIO to the column decoder
GIO	global IO that transfers data between the cell array and the pad
CSL	Column Select Line that connects a group of bitlines to the SIO
IOSW	a signal that connects SIO on the selected sub-array to the LIO
SAP	BLSA power supply voltage
SAN	BLSA ground voltage
V_{BLP}	bitline precharge voltage
V_{PP}	wordline boost voltage

Table A.1: List of acronyms used throughout this thesis.

Appendix B

DramDSE Configuration Example

```
# Essential Description
[device]
TYPE= LPDDR4
DENSITY= 8Gb
DATA_RATE= 3.2Gbps  # Gbps or Mbps

[external_voltage]
VDD2= 1.1V
VDD1= 1.8V

[internal_voltage]
VPERI= 1.1V
VCORE= 0.95V
VDDY = 1.4V, EXT_VDD= VDD1                    # regulator
VPP  = 3.0V, EXT_VDD= VDD1, EFF= 33.3%  # trippler charge pump

# Micro-architecture Description
[sub-dram]
TRANSPOSE_BANK= TRUE  # transposed: BL are parallel to the PAD arrays
SHARED_ROW_DEC= TRUE  # row decoder shared with another sub-bank?
SHARED_COL_DEC= TRUE  # column decoder shared with another sub-bank?

[bank]
NUM_SUBBANK= 4

SWL_PER_MWL= 8
BL_PER_CSL = 8

[sub-array]
NUM_SWD= 9
NUM_MAT= 16

# SEC: 128bit data and 8bit parity
ECC = SEC              # in-DRAM (either one of NONE, SEC, and SECDED)

[mat]
NUM_SWL= 512
NUM_BL = 512
NUM_REDUNDANT_SWL= 8  # 1 redundant MWL
NUM_REDUNDANT_BL = 48 # 6 redundant CSLs
NUM_DUMMY_SWL= 6      # 3 on each side
NUM_DUMMY_BL = 6      # 3 on each side
```

Listing B.1: Architecture Configuration for 8Gb LPDDR4.

```
[technology]
FEATURE_SIZE    = 26nm
PERI_HEIGHT     = 700um      # PAD periphery height
PAD_TSV_HEIGHT  = 120um      # blocking metal wires to be routed

SUBDRAM_X_SPACING= 15um      # spacing that separates two sub-drams
SUBDRAM_Y_SPACING= 15um      # spacing that separates two sub-drams

ROW_DEC_WIDTH  = 120um
COL_DEC_HEIGHT= 340um

SWD_WIDTH   = 5um
BLSA_HEIGHT= 11um

PITCH_SWL   = 0.064um
PITCH_BL    = 0.064um
PITCH_LOCAL= 0.4um

# load capacitance(wire+driver) per segment
GIO_CAPACITANCE   = 0.35pF   # /1000um
CA_CAPACITANCE    = 0.40pF   # /1000um
LOCAL_CAPACITANCE= 0.50pF   # /1000um

IOSW_CAPACITANCE = 0.15pF   # /MAT(one half)
EQ_CAPACITANCE   = 0.10pF   # /MAT(one half)
SAP1_CAPACITANCE = 0.10pF   # /MAT(one half)
SAP2_CAPACITANCE = 0.10pF   # /MAT(one half)
SAN_CAPACITANCE  = 0.10pF   # /MAT(one half)
MWL_CAPACITANCE  = 0.08pF   # /MAT
FX_CAPACITANCE   = 0.06pF   # /MAT
SWL_CAPACITANCE  = 0.03pF   # /MAT

CSL_CAPACITANCE  = 0.030pF  # /sub-array
SIO_CAPACITANCE  = 0.015pF  # /sub-array
LIO_CAPACITANCE  = 0.015pF  # /sub-array

CELL_CAPACITANCE = 10fF     # /BLSA
BLSA_CAPACITANCE = 40fF     # /BLSA

# static current
IDD2N_VDD1=  1.0  # mA
IDD2N_VDD2=  9.0  # mA
IDD3N_VDD1=  1.5  # mA
IDD3N_VDD2= 10.0  # mA
```

Listing B.2: Technology Configuration for 2y nm LPDDR4. Due to the NDA, parameter values listed here are not the same with those used in Chapter 4.5.

Bibliography

[1] Bernard Marr. How much data do we create every day? the mind-blowing stats everyone should read. https://www.forbes.com/sites/bernardmarr/2018/05/21/how-much-data-do-we-create-every-day-the-mind-blowing-stats-everyone-should-read/#5fb8b82060ba.

[2] Paige Tanner. The smartphone market's growth opportunity for micron in 2018. https://marketrealist.com/2018/01/the-smartphone-markets-growth-opportunity-for-micron-in-2018.

[3] Samsung Newsroom. Exploring the key samsung technologies that enabled 10nm-class dram. https://news.samsung.com/global/exploring-the-key-samsung-technologies-that-enabled-10nm-class-dram.

[4] K. Sohn, W. Yun, R. Oh, C. Oh, S. Seo, M. Park, D. Shin, W. Jung, S. Shin, J. Ryu, H. Yu, J. Jung, K. Nam, S. Choi, J. Lee, U. Kang, Y. Sohn, J. Choi, C. Kim, S. Jang, and G. Jin. 18.2 a 1.2v 20nm 307gb/s hbm dram with at-speed wafer-level i/o test scheme and adaptive refresh considering temperature distribution. In *2016 IEEE International Solid-State Circuits Conference (ISSCC)*, pages 316–317, Jan 2016.

[5] JEDEC Solid State Technology Association. LPDDR2 SDRAM Standard JESD209-2F, April 2011.

[6] Micron. DDR3L-RS: Reducing DRAM Power Consumption in Standby. https://www.micron.com/about/blogs/2013/january/ddr3l-rs-reducing-dram-power-consumption-in-standby.

[7] Allan Snavely and Dean M. Tullsen. Symbiotic jobscheduling for a simultaneous multithreaded processor. In *Proceedings of the Ninth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS IX, pages 234–244, New York, NY, USA, 2000. ACM.

[8] Rambus. Understanding dennard scaling. `https://www.rambus.com/blogs/understanding-dennard-scaling-2`.

[9] W. L. Bircher and L. K. John. Complete system power estimation using processor performance events. *IEEE Transactions on Computers*, 61(4):563–577, April 2012.

[10] T. Hamamoto, S. Sugiura, and S. Sawada. On the retention time distribution of dynamic random access memory (dram). *IEEE Transactions on Electron Devices*, 45(6):1300–1309, Jun 1998.

[11] J. Y. Kim, C. S. Lee, S. E. Kim, I. B. Chung, Y. M. Choi, B. J. Park, J. W. Lee, D. I. Kim, Y. S. Hwang, D. S. Hwang, H. K. Hwang, J. M. Park, D. H. Kim, N. J. Kang, M. H. Cho, M. Y. Jeong, H. J. Kim, J. N. Han, S. Y. Kim, B. Y. Nam, H. S. Park, S. H. Chung, J. H. Lee, J. S. Park, H. S. Kim, Y. J. Park, and K. Kim. The breakthrough in data retention time of dram using recess-channel-array transistor(rcat) for 88 nm feature size and beyond. In *2003 Symposium on VLSI Technology. Digest of Technical Papers (IEEE Cat. No.03CH37407)*, pages 11–12, June 2003.

[12] J. V. Kim, H. J. Oh, D. S. Woo, Y. S. Lee, D. H. Kim, S. E. Kim, G. W. Ha, H. J. Kim, N. J. Kang, J. M. Park, Y. S. Hwang, D. I. Kim, B. J. Park, M. Huh, B. H. Lee, S. B. Kim, M. H. Cho, M. Y. Jung, Y. I. Kim, C. Jin, D. W. Shin, M. S. Shim, C. S. Lee, W. S. Lee, J. C. Park, G. Y. Jin, Y. J. Park, and Kinam Kim. S-rcat (sphere-shaped-recess-channel-array transistor) technology for 70nm dram feature size and beyond. In *Digest of Technical Papers. 2005 Symposium on VLSI Technology, 2005.*, pages 34–35, June 2005.

[13] K. Kim. Future memory technology: challenges and opportunities. In *2008 International Symposium on VLSI Technology, Systems and Applications (VLSI-TSA)*, pages 5–9, April 2008.

[14] J. M. Park, Y. S. Hwang, S. W. Kim, S. Y. Han, J. S. Park, J. Kim, J. W. Seo, B. S. Kim, S. H. Shin, C. H. Cho, S. W. Nam, H. S. Hong, K. P. Lee, G. Y. Jin, and E. S. Jung. 20nm dram: A new beginning of another revolution. In *2015 IEEE International Electron Devices Meeting (IEDM)*, pages 26.5.1–26.5.4, Dec 2015.

[15] Kinam Kim. From the future si technology perspective: Challenges and opportunities. In *2010 International Electron Devices Meeting*, pages 1.1.1–1.1.9, Dec 2010.

[16] S. G. Kim, C. S. Hyun, D. Park, S. J. Kim, T. H. Cho, H. J. Kang, S. H. Lee, J. G. Suk, B. K. Lim, Y. S. Jeon, K. H. Hwang, H. S. Hong, S. G. Jeon, K. Y. Lee, K. S. Oh, and D. G. Park. Improved electrical characteristics and retention time of drams using hsg-merged-aho cylinder capacitor. In *2005 International Semiconductor Device Research Symposium*, pages 113–114, Dec 2005.

[17] MARK LAPEDUS. 1xnm dram challenges. `http://semiengineering.com/1xnm-dram-challenges`.

[18] Kibong Koo, Sunghwa Ok, Yonggu Kang, Seungbong Kim, Choungki Song, Hyeyoung Lee, Hyungsoo Kim, Yongmi Kim, Jeonghun Lee, Seunghan Oak, Y. Lee, Jungyu Lee, Joongho Lee, Hyungyu Lee, Jaemin Jang, Jongho Jung, Byeongchan Choi, Yongju Kim, Youngdo Hur, Yunsaing Kim, Byongtae Chung, and Yongtak Kim. A 1.2V 38nm 2.4Gb/s/pin 2Gb DDR4 SDRAM with bank group and half-page architecture. In *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2012 IEEE International*, pages 40–41, Feb 2012.

[19] K. Song, S. Lee, D. Kim, Y. Shim, S. Park, B. Ko, D. Hong, Y. Joo, W. Lee, Y. Cho, W. Shin, J. Yun, H. Lee, J. Lee, E. Lee, N. Jang, J. Yang, H. k. Jung, J. Cho, H. Kim, and J. Kim. A 1.1 v 2y-nm 4.35 gb/s/pin 8 gb lpddr4 mobile

device with bandwidth improvement techniques. *IEEE Journal of Solid-State Circuits*, 50(8):1945–1959, Aug 2015.

[20] T. Y. Oh, H. Chung, J. Y. Park, K. W. Lee, S. Oh, S. Y. Doo, H. J. Kim, C. Lee, H. R. Kim, J. H. Lee, J. I. Lee, K. S. Ha, Y. Choi, Y. C. Cho, Y. C. Bae, T. Jang, C. Park, K. Park, S. Jang, and J. S. Choi. A 3.2 gbps/pin 8 gbit 1.0 v lpddr4 sdram with integrated ecc engine for sub-1 v dram core operation. *IEEE Journal of Solid-State Circuits*, 50(1):178–190, Jan 2015.

[21] JEDEC Solid State Technology Association. DDR3 SDRAM Standard JESD79-3F, July 2012.

[22] JEDEC Solid State Technology Association. DDR4 SDRAM Standard JESD79-4A, November 2013.

[23] JEDEC Solid State Technology Association. LPDDR3 SDRAM Standard JESD209-3C, August 2013.

[24] JEDEC Solid State Technology Association. LPDDR4 SDRAM Standard JESD209-4A, November 2015.

[25] JEDEC Solid State Technology Association. High Bandwidth Memory Standard JESD235A, November 2015.

[26] T. Y. Oh, Y. S. Sohn, S. J. Bae, M. S. Park, J. H. Lim, Y. K. Cho, D. H. Kim, D. M. Kim, H. R. Kim, H. J. Kim, J. H. Kim, J. K. Kim, Y. S. Kim, B. C. Kim, S. H. Kwak, J. H. Lee, J. Y. Lee, C. H. Shin, Y. Yang, B. S. Cho, S. Y. Bang, H. J. Yang, Y. R. Choi, G. S. Moon, C. G. Park, S. W. Hwang, J. D. Lim, K. I. Park, J. S. Choi, and Y. H. Jun. A 7 gb/s/pin 1 gbit gddr5 sdram with 2.5 ns bank to bank active time and no bank group restriction. *IEEE Journal of Solid-State Circuits*, 46(1):107–118, Jan 2011.

[27] Micron tech. note. TN-ED-01: GDDR5 SGRAM Introduction. `https://www.micron.com/~/media/documents/products/technical-note/dram/tned01_gddr5_sgram_introduction.pdf`.

[28] Micron. TN-41-01: Calculating memory system power for DDR3. Technical report.

[29] Micron. TN-41-08: Design guide for two DDR3-1066 UDIMM systems. Technical report.

[30] Samsung. 4Gb B-die DDR3 SDRAM datasheet. `http://www.samsung.com/global/business/semiconductor/file/2011/product/2011/8/29/923862ds_k4b4gxx46b_rev12.pdf`.

[31] Samsung. 4Gb D-die DDR4 SDRAM datasheet rev. 1.7. `http://www.samsung.com/semiconductor/global/file/product/2016/03/DS_K4A4G085WD-B_Rev17-0.pdf`, Dec. 2015.

[32] Shyamkumar Thoziyoor, Jung Ho Ahn, Matteo Monchiero, Jay B. Brockman, and Norman P. Jouppi. A comprehensive memory modeling tool and its application to the design and analysis of future memory hierarchies. In *Proceedings of the 35th Annual International Symposium on Computer Architecture*, ISCA '08, pages 51–62, Washington, DC, USA, 2008. IEEE Computer Society.

[33] T. Vogelsang. Understanding the energy consumption of dynamic random access memories. In *Microarchitecture (MICRO), 2010 43rd Annual IEEE/ACM Int'l Symposium on*, pages 363–374, Dec 2010.

[34] O. Naji, C. Weis, M. Jung, N. Wehn, and A. Hansson. A high-level dram timing, power and area exploration tool. In *2015 International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS)*, pages 149–156, July 2015.

[35] statista. Total unit shipments global DRAM market from 2009 to 2017 (in billion units). `https://www.statista.com/statistics/298814/unit-shipments-global-dram-market/`.

[36] K. Kim. Future silicon technology. In *2012 Proceedings of the European Solid-State Device Research Conference (ESSDERC)*, pages 1–6, Sept 2012.

[37] Arabinda Das. Hynix DRAM layout, process integration adapt to change. `https://www.edn.com/design/memory-design/4403547/3/DRAM-layout--process-integration-adopt-to-change-`.

[38] chipworks. DRAM Process Report. `https://www.chipworks.com/TOC/DRAM_Process_Report-Sample.pdf`.

[39] Yongsam Moon, Yong-Ho Cho, Hyun-Bae Lee, Byung-Hoon Jeong, Seok-Hun Hyun, Byung-Chul Kim, In-Chul Jeong, Seong-Young Seo, Jun-Ho Shin, Seok-Woo Choi, Ho-Sung Song, Jung-Hwan Choi, Kye-Hyun Kyung, Young-Hyun Jun, and Kinam Kim. 1.2V 1.6Gb/s 56nm 6F2 4Gb DDR3 SDRAM with hybrid-I/O sense amplifier and segmented sub-array architecture. In *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2009 IEEE International*, pages 128–129,129a, Feb 2009.

[40] Yoongu Kim, Vivek Seshadri, Donghyuk Lee, Jamie Liu, and Onur Mutlu. A case for exploiting subarray-level parallelism (salp) in dram. *SIGARCH Comput. Archit. News*, 40(3):368–379, June 2012.

[41] N. Chatterjee, M. OConnor, D. Lee, D. R. Johnson, S. W. Keckler, M. Rhu, and W. J. Dally. Architecting an energy-efficient dram system for gpus. In *2017 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 73–84, Feb 2017.

[42] Aniruddha N Udipi, Naveen Muralimanohar, Niladrish Chatterjee, Rajeev Balasubramonian, Al Davis, and Norman P Jouppi. Rethinking DRAM design and organization for energy-constrained multi-cores. *ACM SIGARCH Computer Arch. News*, 38(3):175–186, 2010.

[43] T. Takahashi, T. Sekiguchi, R. Takemura, S. Narui, H. Fujisawa, S. Miyatake, M. Morino, K. Arai, S. Yamada, S. Shukuri, M. Nakamura, Y. Tadaki, K. Kajigaya, K. Kimura, and K. Itoh. A multigigabit dram technology with 6f2 open-bitline cell, distributed overdriven sensing, and stacked-flash fuse. *IEEE Journal of Solid-State Circuits*, 36(11):1721–1727, Nov 2001.

[44] E. Cooper-Balis and B. Jacob. Fine-grained activation for power reduction in DRAM. *Micro, IEEE*, 30(3):34–47, May 2010.

[45] Tao Zhang, Ke Chen, Cong Xu, Guangyu Sun, Tao Wang, and Yuan Xie. Half-DRAM: a high-bandwidth and low-power DRAM architecture from the rethinking of fine-grained activation. In *Computer Architecture (ISCA), 2014 ACM/IEEE 41st International Symposium on*, pages 349–360. IEEE, 2014.

[46] H. Ha, A. Pedram, S. Richardson, S. Kvatinsky, and M. Horowitz. Improving energy efficiency of dram by exploiting half page row access. In *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 1–12, Oct 2016.

[47] K. N. Lim, W. J. Jang, H. S. Won, K. Y. Lee, H. Kim, D. W. Kim, M. H. Cho, S. L. Kim, J. H. Kang, K. W. Park, and B. T. Jeong. A 1.2v 23nm 6f2 4gb ddr3 sdram with local-bitline sense amplifier, hybrid lio sense amplifier and dummy-less array architecture. In *2012 IEEE International Solid-State Circuits Conference*, pages 42–44, Feb 2012.

[48] J.S. Lee. Semiconductor memory device having row repair circuitry, December 24 2002. US Patent 6,498,756.

[49] S.H. Kang. Row access information transfer device using internal wiring of a memory cell array, August 3 2004. US Patent 6,771,555.

[50] K. K. Chang, P. J. Nair, D. Lee, S. Ghose, M. K. Qureshi, and O. Mutlu. Low-cost inter-linked subarrays (lisa): Enabling fast inter-subarray data movement in dram. In *2016 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 568–580, March 2016.

[51] JEDEC Solid State Technology Association. Annex L: Serial Presence Detect (SPD) for DDR4 SDRAM Modules , September 2015.

[52] D. U. Lee, K. W. Kim, K. W. Kim, K. S. Lee, S. J. Byeon, J. H. Kim, J. H. Cho, J. Lee, and J. H. Chun. A 1.2 v 8 gb 8-channel 128 gb/s high-bandwidth

memory (hbm) stacked dram with effective i/o test circuits. *IEEE Journal of Solid-State Circuits*, 50(1):191–203, Jan 2015.

[53] K. Sohn, W. J. Yun, R. Oh, C. S. Oh, S. Y. Seo, M. S. Park, D. H. Shin, W. C. Jung, S. H. Shin, J. M. Ryu, H. S. Yu, J. H. Jung, H. Lee, S. Y. Kang, Y. S. Sohn, J. H. Choi, Y. C. Bae, S. J. Jang, and G. Jin. A 1.2 v 20 nm 307 gb/s hbm dram with at-speed wafer-level io test scheme and adaptive refresh considering temperature distribution. *IEEE Journal of Solid-State Circuits*, 52(1):250–260, Jan 2017.

[54] S. L. Lu, Y. C. Lin, and C. L. Yang. Improving dram latency with dynamic asymmetric subarray. In *2015 48th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 255–266, Dec 2015.

[55] P. Rosenfeld, E. Cooper-Balis, and B. Jacob. Dramsim2: A cycle accurate memory system simulator. *IEEE Computer Architecture Letters*, 10(1):16–19, Jan 2011.

[56] Niladrish Chatterjee, Rajeev Balasubramonian, Manjunath Shevgoor, S Pugsley, A Udipi, Ali Shafiee, Kshitij Sudan, Manu Awasthi, and Zeshan Chishti. USIMM: The Utah simulated memory module. *University of Utah, Tech. Rep*, 2012.

[57] Naveen Muralimanohar, Rajeev Balasubramonian, and Norman Jouppi. Cacti 6.0: A tool to understand large caches. 01 2007.

[58] Heonjae Ha. Dramdse code repository. `https://bitbucket.org/hunjaeha/dramdse`.

[59] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[60] Jung Ho Ahn, Norman P. Jouppi, Christos Kozyrakis, Jacob Leverich, and Robert S. Schreiber. Improving system energy efficiency with memory rank subsetting. *ACM Trans. Archit. Code Optim.*, 9(1):4:1–4:28, March 2012.

[61] Bong Hwa Jeong, Jongwon Lee, Yin Jae Lee, Tae Jin Kang, Joo Hyeon Lee, Duck Hwa Hong, Jae Hoon Kim, Eun Ryeong Lee, Min Chang Kim, Kyung Ha Lee, S. I. Park, Jong Ho Son, Sang Kwon Lee, Seong Nyuh Yoo, Sung Mook Kim, Tae Woo Kwon, Jin Hong Ahn, and Yong Tak Kim. A 1.35v 4.3gb/s 1gb lpddr2 dram with controllable repeater and on-the-fly power-cut scheme for low-power and high-speed mobile application. In *2009 IEEE International Solid-State Circuits Conference - Digest of Technical Papers*, pages 132–133, Feb 2009.

[62] Y. C. Bae, J. Y. Park, S. J. Rhee, S. B. Ko, Y. Jeong, K. S. Noh, Y. Son, J. Youn, Y. Chu, H. Cho, M. Kim, D. Yim, H. C. Kim, S. H. Jung, H. I. Choi, S. Yim, J. B. Lee, J. S. Choi, and K. Oh. A 1.2v 30nm 1.6gb/s/pin 4gb lpddr3 sdram with input skew calibration and enhanced control scheme. In *2012 IEEE International Solid-State Circuits Conference*, pages 44–46, Feb 2012.

[63] I.H. Lee. Refresh circuit for use in semiconductor memory device and operation method thereof, August 28 2007. US Patent 7,263,021.

[64] Marc A. Viredaz and Deborah A. Wallach. Power evaluation of a handheld computer. *IEEE Micro*, 23(1):66–74, January 2003.

[65] Ronak Singhal. Inside Intel next generation Nehalem microarchitecture. In *Hot Chips*, volume 20, 2008.

[66] John L Henning. SPEC CPU2006 benchmark descriptions. *ACM SIGARCH Computer Architecture News*, 34(4):1–17, 2006.

[67] Wim Heirman, Trevor Carlson, and Lieven Eeckhout. Sniper: scalable and accurate parallel multi-core simulation. In *8th International Summer School on Advanced Computer Architecture and Compilation for High-Performance and Embedded Systems (ACACES-2012)*, pages 91–94. High-Performance and Embedded Architecture and Compilation Network of Excellence (HiPEAC), 2012.

[68] Harish Patil, Robert Cohn, Mark Charney, Rajiv Kapoor, Andrew Sun, and Anand Karunanidhi. Pinpointing representative portions of large intel®itanium®programs with dynamic instrumentation. In *Proceedings of the 37th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 37, pages 81–92, Washington, DC, USA, 2004. IEEE Computer Society.

[69] C. K. Lee, Y. J. Eom, J. H. Park, J. Lee, H. R. Kim, K. Kim, Y. Choi, H. J. Chang, J. Kim, J. M. Bang, S. Shin, H. Park, S. Park, Y. R. Choi, H. Lee, K. H. Jeon, J. Y. Lee, H. J. Ahn, K. H. Kim, J. S. Kim, S. Chang, H. R. Hwang, D. Kim, Y. H. Yoon, S. H. Hyun, J. Y. Park, Y. G. Song, Y. S. Park, H. J. Kwon, S. J. Bae, T. Y. Oh, I. D. Song, Y. C. Bae, J. H. Choi, K. I. Park, S. J. Jang, and G. Y. Jin. 23.2 a 5gb/s/pin 8gb lpddr4x sdram with power-isolated lvstl and split-die architecture with 2-die zq calibration scheme. In *2017 IEEE International Solid-State Circuits Conference (ISSCC)*, pages 390–391, Feb 2017.

[70] Y. Watanabe, N. Nakamura, D. Takashima, T. Hara, and S. Watanabe. Offset compensating bit-line sensing scheme for high density drams. In *1992 Symposium on VLSI Circuits Digest of Technical Papers*, pages 116–117, June 1992.

[71] I.P. LEE. Word line driving signal control circuit, semiconductor memory apparatus having the same, and word line driving method, January 21 2014. US Patent 8,634,262.

[72] Dimitris Kaseridis, Jeffrey Stuecheli, and Lizy Kurian John. Minimalist open-page: A dram page-mode scheduling policy for the many-core era. In *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-44, pages 24–35, New York, NY, USA, 2011. ACM.

[73] Jung Ho Ahn, J. Leverich, R.S. Schreiber, and N.P. Jouppi. Multicore dimm: an energy efficient memory module with independently controlled drams. *Computer Architecture Letters*, 8(1):5–8, Jan 2009.

[74] Micron tech. note. TN-47-10: DDR2 posted CAS# additive latency introduction. `http://application-notes.digchip.com/024/24-19971.pdf`.

[75] S.H. Kang. Semiconductor memory device with sense amplifier driver having multiplied output lines, February 21 2006. US Patent 7,002,862.

[76] Jeffreay Carl Gealow. *Impact of Processing Technology on DRAM Sense Amplifier Design*. PhD thesis, Massachusetts Institute of Technology, June 1990.

[77] Wei Zhao and Yu Cao. New generation of predictive technology model for sub-45 nm early design exploration. *Electron Devices, IEEE Transactions on*, 53(11):2816–2823, 2006.

[78] Young Choi. Under the Hood: DRAM architectures: 8F2 vs. 6F2. http://www.eetimes.com/document.asp?doc_id=1281208.

[79] Kyu-Nam Lim, Woong-Ju Jang, Hyung-Sik Won, Kang-Yeol Lee, Hyungsoo Kim, Dong-Whee Kim, Mi-Hyun Cho, Seung-Lo Kim, Jong-Ho Kang, Keun-Woo Park, and Byung-Tae Jeong. A 1.2V 23nm 6F2 4Gb DDR3 SDRAM with local-bitline sense amplifier, hybrid LIO sense amplifier and dummy-less array architecture. In *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2012 IEEE International*, pages 42–44, Feb 2012.

[80] H.J. Oh, J.-Y. Kim, J.H. Kim, S.G. Park, D.H. Kim, S.E. Kim, D.S. Woo, Y.S. Lee, G.W. Ha, J.M. Park, N.J. Kang, H.J. Kim, Y.S. Hwang, B.Y. Kim, D.I. Kim, Y.S. Cho, J.K. Choi, B.H. Lee, S.B. Kim, M.H. Cho, Y.I. Kim, J. Choi, D.W. Shin, M.S. Shim, W.T. Choi, G.P. Lee, Y.J. Park, W.S. Lee, and B.-I. Ryu. High-density low-power-operating DRAM device adopting 6F2 cell scheme with novel S-RCAT structure on 80nm feature size and beyond. In *Solid-State Device Research Conference, 2005. ESSDERC 2005. Proceedings of 35th European*, pages 177–180, Sept 2005.

[81] S. Rixner, W.J. Dally, U.J. Kapasi, P. Mattson, and J.D. Owens. Memory access scheduling. In *Computer Architecture, 2000. Proceedings of the 27th International Symposium on*, pages 128–138, June 2000.

[82] Micron Technology, Inc. DDR4_Power_Calc.XLSM. `https://www.micron.com/~/media/documents/products/power-calculator/ddr4_power_calc.xlsm`, Version 1, Oct. 1, 2014.

[83] Trevor E. Carlson, Wim Heirman, Stijn Eyerman, Ibrahim Hur, and Lieven Eeckhout. An evaluation of high-level mechanistic core models. *ACM Transactions on Architecture and Code Optimization (TACO)*, 2014.

[84] K.T. Malladi, F.A. Nothaft, K. Periyathambi, B.C. Lee, C. Kozyrakis, and M. Horowitz. Towards energy-proportional datacenter memory with mobile DRAM. In *Computer Architecture (ISCA), 2012 39th Annual International Symposium on*, pages 37–48, June 2012.

[85] T. Kawahara, Yoshiki Kawajiri, M. Horiguchi, T. Akiba, G. Kitsukawa, T. Kure, and M. Aoki. A charge recycle refresh for gb-scale dram's in file applications. *Solid-State Circuits, IEEE Journal of*, 29(6):715–722, Jun 1994.

[86] Yuan Taur and Tak H. Ning. *Fundamentals of Modern VLSI Devices*. Cambridge University Press, New York, NY, USA, 1998.

[87] M. H. Cho, Y. I. Kim, D. S. Woo, S. W. Kim, M. S. Shim, Y. J. Park, W. S. Lee, and B. I. Ryu. Analysis of thermal variation of dram retention time. In *2006 IEEE International Reliability Physics Symposium Proceedings*, pages 433–436, March 2006.

[88] Myungjae Lee, Hyungshin Kwon, Jonghyoung Lim, Hongsun Hwang, SeongJin Jang, and Yonghan Roh. Analysis of dynamic retention characteristics of nwl scheme in high density dram. In *Proceedings of the 20th IEEE International Symposium on the Physical and Failure Analysis of Integrated Circuits (IPFA)*, pages 641–644, July 2013.

[89] Xilinx. Artix-7 fpga ac701 evaluation kit. `https://www.xilinx.com/products/boards-and-kits/ek-a7-ac701-g.html`.

[90] Xilinx. Zynq-7000 soc zc706 evaluation kit. `https://www.xilinx.com/products/boards-and-kits/ek-z7-zc706-g.html`.

[91] K. Kim and J. Lee. A new investigation of data retention time in truly nanoscaled drams. *IEEE Electron Device Letters*, 30(8):846–848, Aug 2009.

[92] R.K. Venkatesan, S. Herr, and E. Rotenberg. Retention-aware placement in dram (rapid): software methods for quasi-non-volatile dram. In *High-Performance Computer Architecture, 2006. The Twelfth International Symposium on*, pages 155–165, Feb 2006.

[93] S. Baek, Sangyeun Cho, and R. Melhem. Refresh now and then. *Computers, IEEE Transactions on*, 63(12):3114–3126, Dec 2014.

[94] J. Liu, B. Jaiyen, R. Veras, and O. Mutlu. Raidr: Retention-aware intelligent dram refresh. In *Computer Architecture (ISCA), 2012 39th Annual International Symposium on*, pages 1–12, June 2012.

[95] I. Bhati, Z. Chishti, Shih-Lien Lu, and B. Jacob. Flexible auto-refresh: Enabling scalable and energy-efficient dram refresh reductions. In *Computer Architecture (ISCA), 2015 ACM/IEEE 42nd Annual International Symposium on*, pages 235–246, June 2015.

[96] Young-Ho Gong and S.W. Chung. Exploiting refresh effect of dram read operations: A practical approach to low-power refresh. *Computers, IEEE Transactions on*, PP(99):1–1, 2015.

[97] D.S. Yaney, C.Y. Lu, R.A. Kohler, M.J. Kelly, and J.T. Nelson. A meta-stable leakage phenomenon in dram charge storage - variable hold time. In *Electron Devices Meeting, 1987 International*, pages 336–339, Dec 1987.

[98] P.J. Restle, J.W. Park, and B.F. Lloyd. Dram variable retention time. In *Electron Devices Meeting, 1992. IEDM '92. Technical Digest., International*, pages 807–810, Dec 1992.

[99] M.K. Qureshi, Dae-Hyun Kim, S. Khan, P.J. Nair, and O. Mutlu. Avatar: A variable-retention-time (vrt) aware refresh for dram systems. In *Dependable Systems and Networks (DSN), 2015 45th Annual IEEE/IFIP International Conference on*, pages 427–437, June 2015.

[100] Jeongdong Choe. Samsungs anti-fuse technology found on 18 nm dram: Kilopass 2t-cell technology used. `http://techinsights.com/about-techinsights/overview/blog/samsungs-anti-fuse-technology-found-on-18-nm-dram`.

[101] Yoongu Kim, Ross Daly, Jeremie Kim, Chris Fallin, Ji Hye Lee, Donghyuk Lee, Chris Wilkerson, Konrad Lai, and Onur Mutlu. Flipping bits in memory without accessing them: An experimental study of dram disturbance errors. In *Proceeding of the 41st annual international symposium on Computer architecture*, pages 361–372. IEEE Press, 2014.

ProQuest Number: 28114980

INFORMATION TO ALL USERS
The quality and completeness of this reproduction is dependent on the quality
and completeness of the copy made available to ProQuest.

Distributed by ProQuest LLC (2020).
Copyright of the Dissertation is held by the Author unless otherwise noted.

This work may be used in accordance with the terms of the Creative Commons license
or other rights statement, as indicated in the copyright statement or in the metadata
associated with this work. Unless otherwise specified in the copyright statement
or the metadata, all rights are reserved by the copyright holder.

This work is protected against unauthorized copying under Title 17,
United States Code and other applicable copyright laws.

Microform Edition where available © ProQuest LLC. No reproduction or digitization
of the Microform Edition is authorized without permission of ProQuest LLC.

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 - 1346 USA