A COMPREHENSIVE MEMORY MODELING TOOL FOR DESIGN AND ANALYSIS OF FUTURE MEMORY HIERARCHIES

A Dissertation

Submitted to the Graduate School of the University of Notre Dame in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

by

Shyamkumar Thoziyoor

Jay B. Brockman, Director

Graduate Program in Computer Science and Engineering Notre Dame, Indiana July 2008
A COMPREHENSIVE MEMORY MODELING TOOL FOR DESIGN AND ANALYSIS OF FUTURE MEMORY HIERARCHIES

Abstract

by

Shyamkumar Thoziyoor

This thesis describes CACTI-D, a memory modeling tool that supports both SRAM and DRAM circuits and technologies. With CACTI-D it is possible to project area, access time, cycle time, dynamic read and write energies per access, and standby leakage power of memories and caches for technology nodes between 90nm and 32nm. CACTI-D supports SRAM, logic process based DRAM (LP-DRAM) and commodity DRAM (COMM-DRAM) technologies. CACTI-D also supports the modeling of main memory DRAM chips. Thus with CACTI-D, modeling of the complete memory hierarchy with consistent models all the way from SRAM based L1 caches through main memory DRAMs on DIMMs becomes possible.

CACTI-D is based on the well-known memory and cache modeling tool CACTI. We borrow key circuit models from CACTI but revamp CACTI-D from the ground up with a complete rewrite of the source code. We incorporate a new technology foundation into CACTI-D with device data based on the ITRS roadmap. We incorporate device data for different ITRS device types such as high performance (HP), low standby power (LSTP) and low operating power (LOP). We also incorporate data for interconnect technology based on well-documented models and data from the literature.

We have validated CACTI-D by comparing its projections against real designs. For
SRAM validation, we compare against two prominent 90nm and 65nm SRAM caches and for DRAM validation, we compare against a 78nm DDR3 DRAM chip. Taking into account the extremely generic nature of CACTI-D, there is good agreement between the projections produced by CACTI-D and the published data.

We illustrate the potential applicability of CACTI-D in the design and analysis of future memory hierarchies by carrying out a last level cache study for a multicore multithreaded architecture at the 32nm technology node. In this study we use CACTI-D to model all components of the memory hierarchy including L1, L2, last level SRAM, LP-DRAM or COMM-DRAM based L3 caches, and main memory DRAM chips. We carry out architectural simulation using benchmarks with large data sets and present results of their execution time, breakdown of power in the memory hierarchy, and system energy-delay product for the different system configurations. We find that COMM-DRAM technology is most attractive for stacked last level caches, with significantly lower energy-delay products.
To mom (Geeta) and dad (Sreedharan)
CONTENTS

FIGURES ................................................................. vi

TABLES ................................................................. x

ACKNOWLEDGMENTS ....................................................... xiii

CHAPTER 1: INTRODUCTION ............................................... 1

CHAPTER 2: OVERVIEW OF CACTI-D ................................... 5
   2.1 CACTI History ....................................................... 5
   2.2 CACTI-D Changes and Contributions .............................. 6
   2.3 Related Work ...................................................... 16

CHAPTER 3: DATA ARRAY ORGANIZATION ............................ 18
   3.1 Mat Organization .................................................... 19
   3.2 Routing to Mats ..................................................... 20
   3.3 Organizational Parameters of a Data Array ..................... 24

CHAPTER 4: CIRCUIT MODELS AND SIZING .......................... 28
   4.1 Wire Modeling ....................................................... 28
   4.2 Mat Circuits ......................................................... 29
      4.2.1 Predecoder and Decoder ....................................... 29
      4.2.2 Memory Cell ................................................... 33
      4.2.3 Bitline Peripheral Circuitry .................................. 34
   4.3 Routing Networks .................................................. 37
      4.3.1 Array Edge to Bank Edge H-tree ............................. 38
      4.3.2 Bank Edge to Mat H-tree ..................................... 39

CHAPTER 5: AREA, DELAY, AND ENERGY MODELING .................. 42
   5.1 Area Modeling ...................................................... 42
      5.1.1 Gate Area Model ............................................... 42
FIGURES

2.1 Row predecoding structure of CACTI in version 3. 7
2.2 Row predecoding structure of CACTI in version 4. 8
2.3 (a) L-model of wire used in version 4, (b) Π RC model of wire used in CACTI-D. 15

3.1 Layout of an example array with 4 banks. In this example each bank has 4 subbanks and each subbank has 4 mats. 19
3.2 High-level composition of a mat. 20
3.3 High-level composition of a subarray. 21
3.4 Peripheral circuitry associated with bitlines. Not shown in this figure, but the outputs of the muxes are assumed to be precharged high. 22
3.5 Layout of edge of array to banks H-tree network. 23
3.6 Layout of the horizontal H-tree within a bank. 24
3.7 Layout of the vertical H-trees within a bank. 25
3.8 Different partitions of a bank. 26

4.1 One-section Π RC model that we have assumed for non-ideal wires. 29
4.2 Capacitance model from [36]. 30
4.3 Structure of the row decoding logic for a subarray with 1024 rows. 31
4.4 Structure of the row predecode block for a subarray with 1024 rows. 32
4.5 Row decoding logic circuit paths for a subarray with 1024 rows. One of the circuit paths contains the NAND2 gate of the 2-4 decode unit while the other contains the NAND3 gate of the 3-8 decode unit. 33
4.6 Structure of the row-decoding logic for a subarray with 8 rows. The row-decoding logic is simply composed of 8 decode gates and drivers. 34
4.7 1-ported 6T SRAM cell 34
8.3 Organization of a data array in a set-associative cache of fast access type. 87

8.4 A 64-bit main memory channel connected to 2 DIMMs with each DIMM composed of 8 DRAM chips 91

8.5 Block diagram of a Micron 256Mb DDR2 main memory DRAM chip with 4 banks and 8 data output pins (Courtsey Micron Technology). 92

9.1 Access time, area and power of the 90nm SPARC L2 cache and of solutions generated by CACTI-D. The CACTI-D solutions assume ‘fast’ access mode and ‘semi-global’ wire type outside mat. The 3 plots correspond to 3 activity factors assumed while computing dynamic power for the CACTI-D-generated solutions. 100

9.2 Access time, area and power of the 90nm SPARC L2 cache and of solutions generated by CACTI-D. The CACTI-D solutions are for assumptions of ‘fast’ access mode and ‘global’ wire type outside mat. The 3 plots correspond to 3 activity factors assumed while computing dynamic power for the CACTI-D solutions. 102

9.3 Access time, area and power of the 90nm SPARC L2 cache and of solutions generated by CACTI-D. The CACTI-D solutions are for assumptions of ‘normal’ access mode and ‘semi-global’ wire type outside mat. The 3 plots correspond to 3 activity factors assumed while computing dynamic power for the CACTI-D solutions. 104

9.4 Access time, area and power of the 90nm SPARC L2 cache and of solutions generated by CACTI-D. The CACTI-D solutions are for assumptions of ‘normal’ access mode and ‘global’ wire type outside mat. The 3 plots correspond to 3 activity factors assumed while computing dynamic power for the CACTI-D solutions. 106

9.5 Access time, area and power of the 65nm Xeon L3 cache and of solutions generated by CACTI-D. The CACTI-D solutions assume ‘semi-global’ wire type outside mat. The 3 plots correspond to 3 activity factors assumed while computing dynamic power for the CACTI-D-generated solutions. 111

9.6 Access time, area and power of the 65nm Xeon L3 cache and of solutions generated by CACTI-D. The CACTI-D solutions assume ‘global’ wire type outside mat. The 3 plots correspond to 3 activity factors assumed while computing dynamic power for the CACTI-D solutions. 113

10.1 Access time, random cycle time, area, dynamic read energy per access and leakage power of a 16MB SRAM as max_area_constraint is varied. 120

10.2 Access time, random cycle time, area, dynamic read energy per access and leakage power of a 16MB SRAM as max_acc_time_constraint is varied. 121
10.3 Access time, random cycle time, area, dynamic read energy per access and leakage power of a 16MB SRAM as max_repeater_delay Constraint is varied. ................................................................. 123
10.4 Access time, random cycle time, area, dynamic read energy per access and leakage power of a 16MB SRAM under different optimization function scenarios. ......................................................... 124
10.5 Access time, random cycle time, area, dynamic read energy per access and leakage power of SRAMs for different 65nm device technology assumptions. ......................................................... 126
10.6 Access time, random cycle time, area, dynamic read energy per access and leakage power of SRAMs under aggressive and conservative interconnect technology assumptions. ......................... 127
10.7 Access time, random cycle time, area, dynamic read energy per access and leakage power of SRAMs when “semi-global” or “global” wire type is used for wires outside a mat. ................................................. 128
10.8 Access time, cycle time, area, dynamic read energy per access and standby leakage power of 65nm SRAM, LP-DRAM and COMM-DRAM technologies. Area, dynamic read energy per access and standby leakage power are split up into two charts based on capacity. ......................... 131
10.9 System architecture for LLC study. ........................................ 133
10.10(a) IPC and average read latency, and (b) Components of execution time of NPB applications on different cache configurations. ................................................................. 137
10.11(a) Memory hierarchy power, (b) system power breakdown and system energy-delay product ......................................................... 139
<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>ORGANIZATIONAL PARAMETERS OF A DATA ARRAY</td>
<td>27</td>
</tr>
<tr>
<td>5.1</td>
<td>PROCESS/TECHNOLOGY INPUT PARAMETERS REQUIRED BY THE GATE AREA MODEL</td>
<td>43</td>
</tr>
<tr>
<td>6.1</td>
<td>TECHNOLOGY CHARACTERISTICS OF TRANSISTORS USED IN THE MODEL</td>
<td>64</td>
</tr>
<tr>
<td>6.2</td>
<td>VALUES OF KEY TECHNOLOGY METRICS OF HP, LSTP AND LOP NMOS TRANSISTORS FOR FOUR TECHNOLOGY-NODES FROM THE 2005 ITRS [77]</td>
<td>65</td>
</tr>
<tr>
<td>6.3</td>
<td>TECHNOLOGY DATA ASSUMED FOR AN SRAM CELL</td>
<td>66</td>
</tr>
<tr>
<td>6.4</td>
<td>AGGRESSIVE AND CONSERVATIVE WIRE PROJECTIONS FROM [36]</td>
<td>68</td>
</tr>
<tr>
<td>7.1</td>
<td>TECHNOLOGY PARAMETERS OF THE DRAM CELL</td>
<td>82</td>
</tr>
<tr>
<td>7.2</td>
<td>COMM-DRAM CELL TECHNOLOGY DATA FOR THE 90/65/45/32NM TECHNOLOGY NODES</td>
<td>83</td>
</tr>
<tr>
<td>7.3</td>
<td>LP-DRAM CELL TECHNOLOGY DATA FOR 90NM AND 65NM FROM [29, 92]</td>
<td>84</td>
</tr>
<tr>
<td>7.4</td>
<td>LP-DRAM CELL TECHNOLOGY DATA FOR THE 90/65/45/32NM TECHNOLOGY NODES</td>
<td>84</td>
</tr>
<tr>
<td>8.1</td>
<td>MAXIMUM PRECHARGE STANDBY CURRENT CONSUMPTION OF MAIN MEMORY DRAM CHIPS OF VARIOUS DDR GENERATIONS WITH DIFFERENT CAPACITIES AND SYSTEM CLOCK PERIODS</td>
<td>95</td>
</tr>
<tr>
<td>9.1</td>
<td>CHARACTERISTICS OF SUN’S SPARC 90NM L2 CACHE</td>
<td>97</td>
</tr>
<tr>
<td>9.2</td>
<td>CACTI-D INPUT PARAMETERS USED FOR MODELING 90NM SPARC L2 CACHE</td>
<td>98</td>
</tr>
</tbody>
</table>
9.3 ERROR PERCENTAGES OF PROMINENT SOLUTIONS GENERATED BY CACTI-D WITH RESPECT TO A 90NM SPARC L2 CACHE. THE CACTI-D SOLUTIONS ASSUME ‘FAST’ ACCESS MODE AND ‘SEMI-GLOBAL’ WIRE TYPE OUTSIDE MAT. AN ACTIVITY FACTOR OF 1 HAS BEEN ASSUMED FOR CALCULATION OF DYNAMIC POWER

9.4 ERROR PERCENTAGES OF PROMINENT SOLUTIONS GENERATED BY CACTI-D WITH RESPECT TO A 90NM SPARC L2 CACHE UNDER ASSUMPTIONS OF ‘FAST’ ACCESS MODE AND ‘GLOBAL’ WIRE TYPE OUTSIDE MAT. AN ACTIVITY FACTOR OF 1 HAS BEEN ASSUMED FOR CALCULATION OF DYNAMIC POWER

9.5 ERROR PERCENTAGES OF PROMINENT SOLUTIONS GENERATED BY CACTI-D WITH RESPECT TO A 90NM SPARC L2 CACHE FOR A NORMAL CACHE UNDER ASSUMPTIONS OF ‘NORMAL’ ACCESS MODE AND ‘SEMI-GLOBAL’ WIRE TYPE OUTSIDE MAT. AN ACTIVITY FACTOR OF 1 HAS BEEN ASSUMED FOR CALCULATION OF DYNAMIC POWER

9.6 ERROR PERCENTAGES OF PROMINENT SOLUTIONS GENERATED BY CACTI-D WITH RESPECT TO A 90NM SPARC L2 CACHE FOR A NORMAL CACHE UNDER ASSUMPTIONS OF ‘NORMAL’ ACCESS MODE AND ‘GLOBAL’ WIRE TYPE OUTSIDE MAT. AN ACTIVITY FACTOR OF 1 HAS BEEN ASSUMED FOR CALCULATION OF DYNAMIC POWER

9.7 CHARACTERISTICS OF INTEL XEON’S 65NM L3 CACHE

9.8 CACTI-D INPUT PARAMETERS USED FOR MODELING 65NM INTEL XEON L3 CACHE

9.9 ERROR PERCENTAGES OF PROMINENT SOLUTIONS GENERATED BY CACTI-D WITH RESPECT TO A 65NM INTEL XEON L3 CACHE WHEN WE ASSUME THAT THE DYNAMIC POWER CONSUMED BY THE CACHE IS 5.4W. THE CACTI-D SOLUTIONS ASSUME ‘SEMI-GLOBAL’ WIRE TYPE OUTSIDE MAT. AN ACTIVITY FACTOR OF 1 HAS BEEN ASSUMED FOR CALCULATION OF DYNAMIC POWER

9.10 ERROR PERCENTAGES OF PROMINENT SOLUTIONS GENERATED BY CACTI-D WITH RESPECT TO A 65NM INTEL XEON L3 CACHE WHEN WE ASSUME THAT THE DYNAMIC POWER CONSUMED BY THE CACHE IS 5.4W. THE CACTI-D SOLUTIONS ASSUME ‘GLOBAL’ WIRE TYPE OUTSIDE MAT. AN ACTIVITY FACTOR OF 1 HAS BEEN ASSUMED FOR CALCULATION OF DYNAMIC POWER

9.11 RESULTS OF CACTI-D DRAM MODEL VALIDATION WITH RESPECT TO A 78NM MICRON 1GB DDR3-1066 X8 DRAM

10.1 DEFAULT CACTI-D INPUT PARAMETERS USED FOR STUDIES IN THIS CHAPTER
10.2 DIFFERENT SOLUTION OPTIMIZATION SCENARIOS TARGETING METRICS OF RANDOM CYCLE TIME, DYNAMIC READ ENERGY PER ACCESS, DYNAMIC POWER AND STANDBY LEAKAGE POWER 125

10.3 PROJECTIONS OF KEY PROPERTIES OF THE VARIOUS CACHES AND MAIN MEMORY CHIP AT THE 32NM NODE . . . . . . . . . . 134
ACKNOWLEDGMENTS

This thesis has been possible thanks to the contributions and support of several people.

My advisor Dr. Jay Brockman has been like a friend to me and has been enormously supportive over the past many years. His open-minded approach towards carrying out research has enabled me to explore a wide variety of areas during my time in graduate school. His optimism has rubbed on me and his ability to see the positive in every venture has taught me to do the same, and has kept me going. I want to thank him for his patience and kindness during all the times when I have found it tough to navigate through the challenges of graduate school.

Dr. Norm Jouppi of HP Labs has been a tremendous source of support and guidance. I have been extremely fortunate that I could work with Norm on the development of CACTI and have greatly benefitted from his expertise in almost every area. I want to express my gratitude to him for being so involved with my thesis work and for being on my thesis committee.

I want to thank Dr. Peter Kogge and Dr. Greg Snider for serving on my thesis committee. Dr. Kogge has served as an inspirational figure to me. I want to thank him for his detailed comments about the thesis. I also want to thank Dr. Christian Poellabauer for allowing me to carry out the groundwork for this thesis as part of a project in his class and for serving on my PhD candidacy committee.

Dr. Jung Ho Ahn and Dr. Matteo Monchiero of HP Labs have been very supportive. Jung Ho and Matteo carried out the architectural simulations for our ISCA paper. I want to thank Jung Ho for his tireless involvement. He uncovered various bugs in the CACTI code and technical report, redrew many figures, and carried out several formatting changes that
made everything look better. His contributions towards our ISCA paper were critical. I have enjoyed my interactions with Matteo very much and I want to thank him for being so cool.

Dr. Arun Lokanathan of Micron Technology helped answer several of my commodity DRAM related questions with help from his colleagues Don Morgan, Wen Li and Yadollah Eslami. Also, Dr. Wolfgang Mueller of Qimonda provided data associated with his 2005 IEDM paper. I want to express my thanks to all of them for their help.

Dr. Wenben-Jone of the University of Cincinnati offered a lot of feedback about the CACTI code. I want to thank him for his encouragement and support.

I have enjoyed the collaborations and discussions with other members of the PIM group. I want to express my thanks to Sheng, Srini, Amit, Dan, Shannon and Arun.

I want to thank the different office-mates I have had over the years – Tim, Ying, Joe, Will, Xiarong, Scott, Ryan, Kiefe and Yingping – they made time spent at the office fun. The prediction pool during football season and all the discussions about ND football were a lot of fun.

I want to thank all the other friends I have made at Notre Dame who have made this journey a fun and memorable one. Krishnan, Sajid, Surajit, and VJ have been with me through most of this journey. Aashish has been very supportive. The discussions I have had with Aashish and Krishnan regarding my research have very helpful. I also want to express my thanks to Jindal, Kranti, Prakash, Subhash, Ravi, Deepak, Raj, Jagadish, Amaresh, Yogesh, Manish, RaviKiran, Shiva, Srinath, Mahesh, Ganti, Sharad, Kameshwar, Ajit, Ajay, Aravind, Chandan, Badri, Vasudha, Anees, Sai, Sundeep, Shyam, Amol, Anubhav, Dinesh, Sunil, and Shashank for their friendship and support. I want to thank Dinesh in particular for being a great source of support.

I also want to thank my longtime friend Srinivasan for his friendship and support.

The CSE department staff has been very helpful. I want to thank MD, Curt, Ginny,
Gina, Dian, Jane and Joyce for all the support they have provided.

I want to thank HP and DARPA for funding the work carried out in this thesis.

I want to express my gratitude to my mom (Geeta), dad (Sreedharan) and sister (Nisha) for their constant love and support throughout my education.

Finally, I want to thank all researchers from whose work I have benefitted greatly.
CHAPTER 1

INTRODUCTION

Over the execution of a program, a processor in a computer system may consume and process large amounts of data. This data may reside in main memory or in the multiple levels of cache that are typical of contemporary memory hierarchies. A major challenge in computer architecture research has been the design of effective memory hierarchies that achieve efficient and cost-effective processing of data. In modern computing systems, components in the memory hierarchy are responsible for a significant portion of the silicon area of the system either in the form of main memory of the system or as caches inside the processor chip. With increase in processor clock rates relative to memory latency, data has moved away from the processor, and the importance of the memory hierarchy has increased. With clock speeds saturating in the multi-core processor era and the advent of parallel processing, the performance of the memory hierarchy becomes even more important. With processors becoming more energy-efficient and applications requiring more and more memory, an increasing percentage of system power consumption is now in the memory subsystem [56], so new techniques are required to achieve efficient and cost-effective processing of data. Future many-core processors [88] would have large memory bandwidth requirements, which is not easily available and may require the use of stacking technology [45], so design and analysis of memory hierarchies of these future many-core processors will continue to be of critical importance.

Typically, computer architects carry out performance evaluation of architectures using
simulation frameworks. Simulation frameworks such as SimpleScalar [13], GEMS [52] and M5 [18] that have been developed in academia and are available in the public domain are heavily used for carrying out computer architecture research. These frameworks capture essential, critical properties of the CPU and memory hierarchy of contemporary computer systems. A user can utilize these frameworks to specify system properties and simulate the execution of application programs and obtain an estimate of performance. These simulation frameworks act as powerful tools for investigating new ideas and have played a key role in bringing out innovations in computer architecture. With respect to the memory hierarchy, these frameworks incorporate properties such as latency of main memory, and hit time, capacity, block width, and associativity of the different levels of cache. The hit time of a cache is a function of its design parameters such as capacity, associativity, and block width. So when an architectural study is carried out in which capacity or associativity or block width of a cache is varied, it is important to make sure that the sensitivity of hit time to these properties is maintained. CACTI [95] is a modeling tool which accomplishes that – given input specifications such as capacity, associativity, and block width of cache, CACTI can project the hit time, area, and energy properties of the cache. CACTI was originally developed in 1994 by Steve Wilton and Norm Jouppi and has been heavily used in computer architecture research either directly for modeling of caches and memories, or indirectly through other tools such as Wattch [20]. CACTI is a circuit-level modeling tool and it incorporates analytical equations for area, delay, and energy. With the help of CACTI, computer architects can make informed tradeoffs in area, delay, and energy of caches while carrying out architectural studies.

One limitation of CACTI is that it supports SRAM (Static RAM) technology only. Traditionally, SRAM has been the memory technology of choice for implementation of caches and other memory structures inside a processor die. This is because processing of SRAM is free in a logic process and SRAM has been the practical memory technol-
ogy that could meet access time and random cycle time requirements. Recently however, DRAM (Dynamic RAM) has also found its way into various applications. The main compute chip inside the Blue Gene/L supercomputer uses embedded DRAM [39]. Embedded DRAM has also been used in the graphics synthesizer unit of Sony’s PlayStation2 [65]. The main reason for inclusion of embedded DRAM is its much smaller 1T1C cell (compared to 6T SRAM cell) which results in lesser area and lower leakage. Also, the trend of worsening wire delay with respect to device delay has meant that the smaller cell size of DRAM has a significant benefit in reducing the propagation delay over interconnect which can be significant in large RAMs [53]. DRAM that has been embedded in ASICs so far has been logic process based (LP-DRAM). A logic process is used to fabricate the processor/logic die. Another flavor of DRAM technology is commodity DRAM (COMM-DRAM) technology which is used for fabrication of main memory chips. Recently computer architects have considered the possibility of stacking COMM-DRAM based caches or memories on top of the processor die [19, 46]. Compared to COMM-DRAM technology, LP-DRAM technology typically has faster transistors and circuitry but worse cell density and retention time [92]. These different technology characteristics of SRAM, LP-DRAM and COMM-DRAM translate into different area, delay, and energy properties for memories and caches built using these technologies. In order to investigate the architectural tradeoffs of caches and memories built using these diverse technologies, it is desirable to have a tool similar to CACTI but which can support LP-DRAM and COMM-DRAM technologies as well. This thesis describes the development of such a tool that we call CACTI-D.

CACTI-D is based on version 4 [82] of CACTI. In order to support the significant technology modeling changes and to enable fair comparisons between the technologies, the CACTI code base had to be completely revamped and extensively rewritten. Various organizational and circuit assumptions have been updated to be more relevant to modern
design practice. At the same time, various limitations of version 4 of CACTI have also been addressed including incorporation of a new technology modeling foundation. The base technology modeling has been changed from simple linear scaling of the original 0.8 micron technology to models based on the ITRS roadmap. Another key contribution of CACTI-D is support for modeling of the organization of a main memory chip. So, with CACTI-D it becomes possible to model organizations and operation models suited both for embedded DRAM, and main memory DRAM on DIMMs (Dual In-line Memory Modules). Thus modeling of the complete memory hierarchy with consistent models all the way from SRAM based L1 caches through main memory DRAMs on DIMMs becomes possible.

The rest of this thesis is organized as follows: First in Chapter 2 we give an overview of the changes and contributions of CACTI-D. Next in Chapter 3, we describe how a memory data array is organized in CACTI-D. In Chapter 4, we describe the circuit modeling in CACTI-D and how different circuits are sized. Next in Chapter 5, we present the equations of the area, delay, and energy models for a data array. In Chapter 6 we describe how device and interconnect technology has been modeled in CACTI-D. In Chapter 7, we describe our modeling of DRAM circuits and technologies. Next in Chapter 8, we describe how caches and main memory chips are modeled in CACTI-D. In Chapter 9, we describe how we have validated CACTI-D against commercial SRAM and DRAM designs. Next in Chapter 10, we present the results of several studies that we have carried out that make use of CACTI-D including an architectural study of last level cache tradeoffs involving SRAM, LP-DRAM, and COMM-DRAM L3 caches for a multicore multithreaded architecture at the 32nm technology node. Finally in Chapter 11 we conclude this thesis with suggestions for future work.
CACTI-D has been developed by modifying and enhancing CACTI from the ground up. In this chapter, we begin by first taking a look at the history and evolution of CACTI. Next we present a summary of the changes made in CACTI-D with respect to version 4 of CACTI. Finally, we present a summary of other prominent memory modeling efforts in the literature.

2.1 CACTI History

CACTI [95] was originally developed in 1994 at the Western Research Laboratory of Digital Equipment Corporation by Steve Wilton and Norm Jouppi. The name CACTI was derived from the phrase “Cache Access and Cycle Timing Information”. The original version 1 of CACTI incorporated analytical models for access time and cycle time of direct-mapped and set-associative caches. A user could specify high level parameters of a cache such as capacity, associativity, block with, and output width, and the model would project access time and random cycle time for the cache. Version 2 of CACTI [70] was developed by Glen Reinman and Norm Jouppi in 2000, and the main enhancement included addition of a dynamic energy model and support for fully associative caches. Version 2 also added support for multiported memories and other improvements. Version 3 of CACTI was developed by Premkishore Shivakumar and Norm Jouppi and its main
enhancement was the addition of an area model. Version 3 also added support for multi-
banked memories and other improvements. Version 4 was developed by David Tarjan,
Shyamkumar Thoziyoor and Norm Jouppi in 2006 and its primary enhancement was the
addition of a leakage power model and a web interface. Version 4 also added support for
modeling sequential and fast cache access modes amongst other enhancements.

CACTI-D, which is the topic of this thesis is being released as version 5 [85] of
CACTI. Version 5 was developed by Shyamkumar Thoziyoor, Naveen Muralimanohar,
Jung Ho Ahn and Norm Jouppi. CACTI version 6 [61] was developed by Naveen Murali-
manohar, Rajeev Balasubramaniom and Norm Jouppi in 2007. Version 6 added support
for modeling of NUCA (Non Uniform Cache Access) architecture and interconnect alter-
natives.

2.2 CACTI-D Changes and Contributions

In this section we describe the changes in CACTI-D with respect to CACTI version 4
and its contributions. Detailed description of these changes and of all modeling aspects
of CACTI-D is presented in the chapters that follow.

DRAM Modeling

The main contribution of CACTI-D is the incorporation of models for DRAM circuits
and technologies. With CACTI-D it becomes possible to model embedded memories and
caches based on LP-DRAM and COMM-DRAM technologies in additional to traditional
SRAM. In our modeling of embedded DRAM, we leverage the similarity that exists in
the global and peripheral circuitry of embedded SRAM and DRAM and model only their
essential differences. We use the same array organization for embedded DRAM that
we used for embedded SRAM. By having a common framework that, in general, places
embedded SRAM and DRAM on an equal footing and emphasizes only their essential
differences, we are able to compare relative tradeoffs between embedded SRAM and DRAM. We describe the modeling of DRAM in Chapter 7.

CACTI-D also adds support for modeling the organization of a main memory chip. A main memory chip is organized and operated differently from an embedded memory. We discuss main memory modeling in Chapter 8.

Organizational Changes

As shown in Figure 2.1, earlier versions of CACTI (up to and including version 3) made use of a single row predecoder at the center of a memory bank with the row predecoded signals being driven to the subarrays for decoding. In version 4.0, this centralized decoding logic was implicitly replaced with distributed decoding logic as shown in Figure 2.2. Using H-tree distribution, the address bits were transmitted to the distributed sinks where the decoding took place. However, because of some inconsistencies in the mod-
eloping, it was not clear at what granularity the distributed decoding took place - whether there was one sink per subarray or 2 or 4 subarrays. There were some other problems with the CACTI code such as the following:

- The area model was not updated after version 3.2, so the impact on area of moving from centralized to distributed decoding was not captured. Also, the leakage model did not account for the multiple distributed sinks. The impact of cache access type (normal/sequential/fast) [82] on area was also not captured;
- Number of address bits routed to the subarrays was being computed incorrectly;
- Gate load seen by NAND gate in the 3-8 line predecode block was being computed incorrectly; and
- There were problems with the logic computing the degree of muxing at the tristate output drivers in a subarray.

In CACTI-D, we resolve these issues, redefine and clarify what the organizational assumptions of memory are and remove ambiguity from the modeling. Details about the organization of memory can be found in Chapter 3.
Circuit and Sizing Changes

Earlier versions of CACTI made use of row decoding logic with two stages - the first stage was composed of 3-8 predecode blocks (composed of NAND3 gates) followed by a NOR decode gate and wordline driver. The number of gates in the row decoding path was kept fixed and the gates were then sized using the method of logical effort [81] for an effective fanout of 3 per stage. In CACTI-D, in addition to the row decoding logic, we also model the bitline mux decoding logic and the sense-amplifier mux decoding logic. We use the same circuit structures to model all decoding logic and we base the modeling on the effort by Amrutur and Horowitz described in [11]. We use the sizing heuristic described in [11] that has been shown to be good from an energy-delay perspective. With the new circuit structures and modeling that we use, the limit on maximum number of signals that can be decoded is increased from 4096 (in version 4.2) to 262144. While we do not expect the number of signals that are decoded to be very high, extending the limit from 4096 helps with exploring area/delay/power tradeoffs in a more thorough manner for large memories, especially for large DRAMs. Details of the circuits modeling of decoding logic circuits are presented in Chapter 4.

There are certain problems with the modeling of the H-tree distribution network in version 4.2. An inverter-driver is placed at branches of the address, datain and dataout H-tree. However, the dataout H-tree does not model tristate drivers. The output data bits may come from a few subarrays and so the address needs to be distributed to a few subarrays, however, dynamic power spent in transmitting address is computed as if all the data comes from a single subarray. The leakage in the drivers of the datain H-tree is not modeled. In CACTI-D, we model the H-tree distribution network more rigorously. For the dataout H-tree we model tristate buffers at each branch. For the address and datain H-trees, instead of assuming inverters at the branches of the H-tree we assume the use of buffers that may be gated to allow or disallow the passage of signals and thereby control the dynamic...
power. We size these drivers based on the methodology described in [11] which takes the resistance and capacitance of intermediate wires into account during sizing. We also model the use of repeaters in the H-tree distribution network which are sized according to equations from [36]. Details of the circuit modeling of the H-tree distribution networks are presented in Chapter 4.

Technology Changes

Earlier versions of CACTI relied on a complicated way of obtaining device data for the input technology-node. Computation of access/cycle time and dynamic power were based on device data of a 0.8-micron process that was scaled to the given technology-node using simple linear scaling principles. Leakage power calculation, however, made use of Ioff (subthreshold leakage current) values that were based on device data obtained through BSIM3 parameter extractions. In version 4.2, BSIM3 extraction was carried out for a few select technology nodes (130/100/70nm); as a result leakage power estimation was available only for these select technology nodes.

There are several problems with the above approach of obtaining device data. Using two sets of parameters, one for computation of access/cycle time/dynamic power and another for leakage power is a convoluted approach and is hard to maintain. Also, the approach of basing device parameter values on a 0.8-micron process is not a good one because of several reasons. Device scaling has become quite non-linear in the deep-submicron era. Device performance targets can no longer be achieved through simple linear scaling of device parameters. Moreover, it is well-known that physical gate-lengths (according to the ITRS, physical gate-length is the final, as-etched length of the bottom of the gate electrode) have scaled much more aggressively [14, 77] than what would be projected by simple linear scaling from the 0.8 micron process.

In CACTI-D, we adopt a simpler, more evolvable approach of obtaining device data.
We use device data that the ITRS [77] uses to make its projections. The ITRS makes use of the MASTAR software tool (Model for Assessment of CMOS Technologies and Roadmaps) [76] for computation of device characteristics of current and future technology nodes. Using MASTAR, device parameters may be obtained for different technologies such as planar bulk, double gate and Silicon-On-Insulator. MASTAR includes device profile and result files of each year/technology-node for which the ITRS makes projections and we incorporate the data from these files into CACTI-D. These device profiles are based on published industry process data and industry-consensus targets set by historical trends and system drivers. While it is not necessary that these device numbers match or would match process numbers of various vendors in an exact manner, they do come within the same ball-park as can be seen by looking at the Ion-Ioff cloud graphic within the MASTAR software which shows a scatter plot of various published vendor Ion-Ioff numbers and corresponding ITRS projections. With this approach of using device data from the ITRS, it also becomes possible to incorporate device data corresponding to different device types that the ITRS defines such as high performance (HP), LSTP (Low Standby Power) and Low Operating Power (LOP). More details about the device data used in CACTI-D can be found in Chapter 6.

There are some problems with interconnect modeling of version 4 also. Version 4 utilizes 2 types of wires in the delay model, ‘local’ and ‘global’. The local type is used for wordlines and bitlines, while the global type is used for all other wires. The resistance per unit length and capacitance per unit length for these two wire types are also calculated in a convoluted manner. For a given technology, the resistance per unit length of the local wire is calculated by assuming ideal scaling in all dimensions and using base data of a 0.8-micron process. The base resistance per unit length for the 0.8-micron process is itself calculated by assuming copper wires in the base 0.8-micron process and readjusting the sheet resistance value of version 3 which assumed aluminium wires. As the resistivity of
copper is about 2/3rd that of aluminium, the sheet resistance of copper was computed to be 2/3rd that of aluminium. However, this implies that the thickness of metal assumed in versions 3 and 4 are the same which turns out to be not true. When we compute sheet resistance for the 0.8-micron process with the thickness of local wire assumed in version 4 and assuming a resistivity of 2.2 \(\mu\)ohm-cm for copper, the value comes out to be a factor of 3.4 smaller than that used in version 3. In version 4, resistance per unit length for the global wire type is calculated to be smaller than that of local wire type by a factor of 2.04. This factor of 2.04 is calculated based on RC delays and wire sizes of different wire types in the 2004 ITRS but the underlying assumptions are not known. Another problem is that even though the delay model makes use of two types of wires, local and global, the area model makes use of just the local wire type and the pitch calculation of all wires (local type and global type) are based on the assumed width and spacing for the local wire type; this results in an underestimation of pitch (and area) occupied by the global wires.

Capacitance per unit length calculation of version 4 also suffers from certain problems. The capacitance per unit length values for local and global wire types are assumed to remain constant across technology nodes. The capacitance per unit length value for local wire type was calculated for a 65nm process as \((2.9/3.6) \times 230 = 185\)fF/m where 230fF/m is the published capacitance per unit length value for an Intel 130nm process [83], 3.6 is the dielectric constant of the 130nm process and 2.9 is the dielectric constant of an Intel 65nm process [14]. Computing the value of capacitance per unit length in this manner for a 65nm process ignores the fact that the fringing component of capacitance remains almost constant across technology-nodes and scales very slowly [36, 73]. Also, assuming that the dielectric constant remains fixed at 2.9 for future technology nodes ignores the possibility of use of lower-k dielectrics. Capacitance per unit length of the global type wire of version 4 is calculated to be smaller than that of local type wires by a factor of 1.4. This factor of 1.4 is again calculated based on RC delays and wire
sizes of different wire types in the 2004 ITRS but the underlying assumptions again are not known.

In CACTI-D, we remove the ambiguity from the interconnect modeling. We use the interconnect projections made by Ho et al in [36,38] which are based on well-documented simple models of resistance and capacitance. Because of the difficulty in projecting the values of interconnect properties in an exact manner at future technology nodes the approach employed in [36, 38] was to come up with two sets of projections based on aggressive and conservative assumptions. The aggressive projections assume aggressive use of low-k dielectrics, insignificant resistance degradation due to dishing and scattering, and tall wire aspect ratios. The conservative projections assume limited use of low-k dielectrics, significant resistance degradation due to dishing and scattering, and smaller wire aspect ratios. We incorporate both sets of projections into CACTI-D. We also model 2 types of wires inside CACTI-D - semi-global and global with properties identical to that described in [36, 38]. More details of the interconnect modeling are described in Chapter 6.

Miscellaneous Changes

Optimization Function Change In CACTI-D, we follow a different approach compared to version 4 in finding the optimal solution. Our new approach allows users to exercise more control on area, delay and power of the final solution. The optimization is carried out in the following steps: first, we find all solutions with area efficiency that is within a certain percentage (user-supplied value) of the area efficiency of the solution with best area efficiency. We refer to this area constraint as `max_area_constraint`. Next, from this reduced set of solutions that satisfy the `max_areaconstraint`, we find all solutions with access time that is within a certain percentage of the best access time solution. We refer to this access time constraint as `max_acc_time_constraint`. To the subset of solutions that
results after the application of max_acc_time constraint, we apply the following optimization function:

\[
\text{optimization-func} = \frac{\text{dynamic-energy}}{\text{min-dynamic-energy}} \cdot \text{flag-opt-for-dynamic-energy} + \frac{\text{dynamic-power}}{\text{min-dynamic-power}} \cdot \text{flag-opt-for-dynamic-power} + \frac{\text{leak-power}}{\text{min-leak-power}} \cdot \text{flag-opt-for-leak-power} + \frac{\text{rand-cycle-time}}{\text{min-rand-cycle-time}} \cdot \text{flag-opt-for-rand-cycle-time}
\]  

(2.1)

where dynamic-energy, dynamic-power, leak-power and rand-cycle-time are the dynamic energy, dynamic power, leakage power, and random cycle time of a solution respectively, and min-dynamic-energy, min-dynamic-power, min-leak-power and min-rand-cycle-time are their minimum (best) values in the subset of solutions being considered. flag-opt-for-dynamic-energy, flag-opt-for-dynamic-power, flag-opt-for-leak-power and flag-opt-for-rand-cycle-time are user-specified boolean variables. The new optimization process allows exploration of the solution space in a controlled manner to arrive at a solution with user-desired characteristics.

New Gate Area Model In CACTI-D, we introduce a new analytical gate area model described by Yoshida et al in [97]. With the new gate area model it becomes possible to make the areas of gates sensitive to transistor sizing so that when transistor sizing changes, the areas also change. With the new gate area model, transistors may get folded when they are subject to pitch-matching constraints and the area is calculated accordingly. This feature is useful in capturing differences in area caused due to different pitch-matching constraints that may have to be satisfied, particularly between SRAM and DRAM.
Wire Model  Version 4 models wires using the equivalent circuit model shown in Figure 2.3(a). The Elmore delay of this model is $RC/2$, however this model underestimates the wire-to-gate component ($R_{\text{wire}}C_{\text{gate}}$) of delay. In CACTI-D, we replace this model with the $\Pi$ RC model, shown in Figure 2.3(b), which has been used in more recent SRAM modeling efforts [10].

![Diagram of L-model and $\Pi$ RC model of wire](image)

Figure 2.3. (a) L-model of wire used in version 4, (b) $\Pi$ RC model of wire used in CACTI-D.

Leakage Control Mechanism  Modern memories and caches are equipped with mechanisms that reduce leakage. In CACTI-D, for the SRAM memories and caches, we provide support for modeling a leakage control mechanism similar to that used in the 16MB L3 cache of 65nm Intel Xeon processor [21]. In order to control leakage in the cache, the Intel cache implements n and p sleep transistors at the level of ‘blocks’ within subarrays (each subarray within the Intel cache is composed of multiple ‘blocks’ with one block within a subarray activated per access. The ‘subarray’ of the Intel cache is not the same as that of CACTI-D). The impact of these sleep transistors is that leakage power of all blocks that are not activated during an access is cut down by half.

In CACTI-D, we model the impact of a similar mechanism by introducing a variable $\text{mat}_{\text{leak reduction due to sleep transistors factor}}$. If $\text{mat}_{\text{leak reduction due to sleep transistors factor}}$ is set to 2, then the leakage of all mats that are not activated during an
access is cut down by half.

ECC and Redundancy  In order to be able to check and correct soft errors, most memories of today have support for ECC (Error Correction Code). In CACTI-D, we capture the impact of ECC by incorporating a model that captures the ECC overhead in memory cell and data bus (datain and dataout) area. We incorporate a variable that specifies the number of data bits per ECC bit. By default, we fix the value of this variable to 8.

In order to improve yield, many memories of today incorporate redundant entities even at the subarray level. For example, the data array of the 16MB Intel Xeon L3 cache [21] which has 256 subarrays also incorporates 32 redundant subarrays. In CACTI-D, we incorporate a variable that specifies the number of mats per redundant mat. By default, we fix the value of this variable to 8.

2.3 Related Work

Apart from CACTI, there are a number of SRAM and cache modeling efforts in the literature. An early effort was the development of an area model for embedded memories by Mulder, Quach and Flynn [60]. Another early effort was the development of an access time model for embedded caches by Wada, Rajan, and Przybylski [90]. Evans and Franzon [30] developed an energy model for SRAMs and used their model for predicting an optimum organization for SRAMs. The effort by Amrutur and Horowitz [10–12] is a prominent one that has influenced subsequent SRAM modeling efforts including CACTI-D. CACTI-D follows [11] in the use of the method of logical effort for sizing decoders. A major contribution of eCACTI [51] was the incorporation of a leakage power model into CACTI; CACTI version 4 and CACTI-D both employ the leakage power modeling methodology of eCACTI. Bhavnagarwala, Kosonocky and Meindl [17] modeled an interconnect-centric SRAM architecture with hierarchical array division that is suitable
for large RAM capacities. With CACTI-D also we are able to model very large RAM capacities through the use of an interconnect-centric organization composed of mats and request/reply H-tree networks. Rodriguez and Jacob [72] incorporated SRAM technology data for 90/65/45/32nm based on predictive models, and modeled the impact of pipelining overhead in caches. Zeng, Rose, and Guttman developed PRACTICS [98] which also added a DRAM delay model whose access time was validated against a 250nm NEC embedded DRAM design. Do et al [25] made use of analytical models in order to model dynamic energy and circuit simulation of basic memory structures in order to estimate leakage power. Liang, Turgay, and Brooks [50] made use of a mixture of analytical and empirical methods in order to improve the accuracy of SRAM power models. In their models, they also consider the impact of leakage variability on power which CACTI-D does not support currently. CACTI-D adds to these existing efforts by providing a new technology modeling foundation as well as support for DRAM technologies and operational models, making it possible to model the complete memory hierarchy with consistent models all the way from SRAM based L1 caches through main memory DRAMs on DIMMs.
CHAPTER 3

DATA ARRAY ORGANIZATION

In this chapter we describe how a data array is organized in CACTI-D. We incorporate a hierarchical scalable organization for the array that is suitable for modeling of large RAMs. At the highest level, a data array is composed of multiple identical banks ($N_{\text{banks}}$). These banks may either be accessed concurrently to serve as multiported memory or in case of a main memory DRAM chip they share the address and data buses. Each bank is composed of multiple identical subbanks ($N_{\text{subbanks}}$) with one subbank being activated per access. Each subbank is composed of multiple identical mats ($N_{\text{mats-in-subbank}}$). All mats in a subbank are activated during an access with each mat holding part of the accessed word in the bank. Each mat itself is a self-contained memory structure composed of 4 identical subarrays and associated predecoding logic. Each subarray is a 2D matrix of memory cells and associated peripheral circuitry. Figure 3.1 shows the layout of an array with 4 banks. In this example each bank is shown to have 4 subbanks and each subbank is shown to have 4 mats. Not shown in Figure 3.1, address and data are assumed to be distributed to the mats on H-tree distribution networks.

The rest of this chapter further describes details of the array organization assumed in CACTI-D. Section 3.1 describes the organization of a mat. Section 3.2 describes the organization of the H-tree distribution networks. Section 3.3 presents the different organizational parameters associated with a data array.
Figure 3.1. Layout of an example array with 4 banks. In this example each bank has 4 subbanks and each subbank has 4 mats.

3.1 Mat Organization

Figure 3.2 shows the high-level composition of a mat. A mat is always composed of 4 subarrays and associated predecoding/decoding logic which is located at the center of the mat. The predecoding/decoding logic is shared by all 4 subarrays. The bottom subarrays are mirror images of the top subarrays and the left hand side subarrays are mirror images of the right hand side ones. Not shown in this figure, by default, address/datain/dataout signals are assumed to enter the mat in the middle through its sides; alternatively, under user-control, it may also be specified to assume that they traverse over the memory cells.

Figure 3.3 shows the high-level composition of a subarray. The subarray consists of a 2D matrix of the memory cells and associated peripheral circuitry. Figure 3.4 shows the peripheral circuitry associated with bitlines of a subarray. After a wordline gets activated, memory cell data get transferred to bitlines. The bitline data may go through a level of bitline multiplexing before it is sensed by the sense amplifiers. Depending on the degree of bitline multiplexing, a single sense amplifier may be shared by multiple bitlines. The data is sensed by the sense amplifiers and then passed to tristate output drivers which drive the dataout vertical H-tree (described later in this section). An additional level of multiplexing may be required at the outputs of the sense amplifiers in organizations
Figure 3.2. High-level composition of a mat.

in which the bitline multiplexing is not sufficient to cull out the output data or in set-associative caches in which the output word from the correct way needs to be selected. The select signals that control the multiplexing of the bitline mux and the sense amp mux are generated by the bitline mux select signals decoder and the sense amp mux select signals decoder respectively. When the degree of multiplexing after the outputs of the sense amplifiers is simply equal to the associativity of the cache, the sense amp mux select signal decoder does not have to decode any address bits and instead simply buffers the input way-select signals that arrive from the tag array.

3.2 Routing to Mats

Address and data are routed to and from the mats on H-tree distribution networks. H-tree distribution networks are used to route address and data and provide uniform access to all the mats in a large memory. Such a memory organization is interconnect-centric and is well-suited for coping with the trend of worsening wire delay with respect to device delay. Rather than shipping a bunch of predecoded address signals to the mats, it makes sense to ship the address bits and decode them at the sinks (mats) [75]. Contemporary
divided wordline architectures which make use of broadcast of global signals suffer from increased wire delay as memory capacities get larger [10]. Details of a memory organization similar to what we have assumed may also be found in [8]. For ease of pipelining multiple accesses in the array, separate request and reply networks are assumed. The request network carries address and datain from the edge of the array to the mats while the reply network carries dataout from the mats to the edge of the array. The structure of the request and reply networks is similar; here we discuss the high-level organization of the request network.

The request H-tree network is divided into two networks:

1. The H-tree network from the edge of the array to the edge of a bank; and,
2. The H-tree network from the edge of the bank to the mats.

Figure 3.5 shows the layout of the request H-tree network between the array edge and the banks. Address and datain are routed to each bank on this H-tree network and enter
Figure 3.4. Peripheral circuitry associated with bitlines. Not shown in this figure, but the outputs of the muxes are assumed to be precharged high.

each bank at the middle from one of its sides. The H-tree network from the edge of the bank to the mats is further divided into two 1-dimensional horizontal and vertical H-tree networks. Figure 3.6 shows the layout of the horizontal H-tree within a bank which is located at the middle of the bank while Figure 3.7 shows the layout of the vertical H-trees within a bank. The leaves of the horizontal H-tree act as the parent nodes (marked as V0) of the vertical H-trees. In order to understand the routing of signals on the H-tree networks within a bank, we use an illustrative example. Consider a bank with the following parameters: 1MB capacity, 256-bit output word, 4 subbanks, 4 mats in each subbank. Looked at together, Figures 3.6 and 3.7 can be considered to be the horizontal and vertical H-trees within such a bank. The number of address bits required to address a
Figure 3.5. Layout of edge of array to banks H-tree network.

word in this bank is 15 (since the size of a bank is 1MB and size of a word is 32 bytes). As there are 4 subbanks and because each mat in a subbank is activated during an access, the number of address bits that need to be distributed to each mat is 13. Because each mat in a subbank produces 64 out of the 256 output bits, the number of datain signals that need to be distributed to each mat is 64. Thus 15 bits of address and 256 bits of datain enter the bank from the left side driven by the H0 node. At the H1 node, the 15 address signals are redriven such that each of the two nodes H1 receive the 15 address signals. The datain signals split at node H1 and 128 datain signals go to the left H2 node and the other 128 go to the right H2 node. At each H2 node, the address signals are again redriven such that all of the 4 V0 nodes end up receiving the 15 address bits. The datain signals again split at each H2 node so that each V0 node ends up receiving 64 datain bits. These 15 address bits and 64 datain bits then traverse to each mat along the 4 vertical H-trees. In the vertical H-trees, address and datain may either be assumed to be broadcast to all mats or alternatively, it may be assumed that these signals are appropriately gated so that
they are routed to just the correct subbank that contains the data; by default, we assume the latter scenario.

The reply network H-trees are similar in principle to the request network H-trees. In case of the reply network vertical H-trees, dataout bits from each mat of a subbank travel on the vertical H-trees to the middle of the bank where they sink into the reply network horizontal H-tree, and are carried to the edge of the bank.

3.3 Organizational Parameters of a Data Array

In order to calculate the optimal organization based on a given objective function, like earlier versions of CACTI [70, 79, 82, 95], each bank is associated with partitioning
parameters $N_{dw1}$, $N_{dbl}$ and $N_{spd}$, where $N_{dw1}$ = number of times a row (or wordline) in a bank has been cut with vertical cut lines to create shorter wordlines in a subarray, $N_{dbl}$ = number of times a bitline in a bank has been cut with horizontal cut lines to create shorter bitlines in a subarray, and $N_{spd}$ = number of sets of a cache that are mapped to each row of a bank. Even for a pure scratchpad memory (not a cache), $N_{spd}$ is varied in order to vary the aspect ratio of the bank.

$N_{subbanks}$ and $N_{mats-in-subbank}$ are related to $N_{dw1}$ and $N_{dbl}$ as follows:
Figure 3.8. Different partitions of a bank.

\[ N_{\text{subbanks}} = \frac{N_{\text{dbl}}}{2} \quad (3.1) \]
\[ N_{\text{mats-in-subbank}} = \frac{N_{\text{dwl}}}{2} \quad (3.2) \]

Figure 3.8 shows different partitions of the same bank. The partitioning parameters are labeled alongside. Table 3.1 lists the various organizational parameters associated with a data array. For each parameter, we have also indicated whether it’s specified by the user or whether it’s a degree of freedom that is varied internally to explore different partitions or whether it’s calculated internally based on values of other parameters.
### TABLE 3.1

ORGANIZATIONAL PARAMETERS OF A DATA ARRAY

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Meaning</th>
<th>Parameter Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_{\text{banks}}$</td>
<td>Number of banks</td>
<td>User input</td>
</tr>
<tr>
<td>$N_{\text{dwl}}$</td>
<td>Number of vertical divisions of a wordline in a bank</td>
<td>Degree of freedom</td>
</tr>
<tr>
<td>$N_{\text{dbl}}$</td>
<td>Number of horizontal divisions of a bitline in a bank</td>
<td>Degree of freedom</td>
</tr>
<tr>
<td>$N_{\text{spd}}$</td>
<td>Number of sets mapped to a bank wordline (for data array of cache)</td>
<td>Degree of freedom</td>
</tr>
<tr>
<td>$D_{\text{bitline-mux}}$</td>
<td>Degree of muxing at bitlines</td>
<td>Degree of freedom</td>
</tr>
<tr>
<td>$D_{\text{senseamp-mux}}$</td>
<td>Degree of muxing at sense amp outputs</td>
<td>Degree of freedom</td>
</tr>
<tr>
<td>$N_{\text{subbanks}}$</td>
<td>Number of subbanks</td>
<td>Calculated</td>
</tr>
<tr>
<td>$N_{\text{mats-in-subbank}}$</td>
<td>Number of mats in a subbank</td>
<td>Calculated</td>
</tr>
<tr>
<td>$N_{\text{subarr-rows}}$</td>
<td>Number of rows in a subarray</td>
<td>Calculated</td>
</tr>
<tr>
<td>$N_{\text{subarr-cols}}$</td>
<td>Number of columns in a subarray</td>
<td>Calculated</td>
</tr>
<tr>
<td>$N_{\text{subarr-senseamps}}$</td>
<td>Number of sense amplifiers in a subarray</td>
<td>Calculated</td>
</tr>
<tr>
<td>$N_{\text{subarr-out-drivers}}$</td>
<td>Number of output drivers in a subarray</td>
<td>Calculated</td>
</tr>
<tr>
<td>$N_{\text{bank-addr-bits}}$</td>
<td>Number of address bits to a bank</td>
<td>Calculated</td>
</tr>
<tr>
<td>$N_{\text{bank-datain-bits}}$</td>
<td>Number of datain bits to a mat</td>
<td>Calculated</td>
</tr>
<tr>
<td>$N_{\text{bank-dataout-bits}}$</td>
<td>Number of dataout bits from a mat</td>
<td>Calculated</td>
</tr>
<tr>
<td>$N_{\text{mat-addr-bits}}$</td>
<td>Number of address bits to a mat</td>
<td>Calculated</td>
</tr>
<tr>
<td>$N_{\text{mat-datain-bits}}$</td>
<td>Number of datain bits to a mat</td>
<td>Calculated</td>
</tr>
<tr>
<td>$N_{\text{mat-dataout-bits}}$</td>
<td>Number of dataout bits from a mat</td>
<td>Calculated</td>
</tr>
<tr>
<td>$N_{\text{mat-way-select}}$</td>
<td>Number of way-select bits to a mat (for data array of cache)</td>
<td>Calculated</td>
</tr>
</tbody>
</table>
CHAPTER 4

CIRCUIT MODELS AND SIZING

In Chapter 3, the high-level organization of an array was described. In this chapter, we delve deeper into logic and circuit design of the different entities. In general the sizing of circuits depends on various optimization goals: circuits may be sized for minimum delay, minimum energy-delay product, etc. CACTI-D’s goal is to model simple representative circuit sizing applicable to a broad range of common applications. As in earlier SRAM modeling efforts [10,11,51], we have made extensive use of the method of logical effort [81] in sizing different circuit blocks. Explanation of the method of logical effort may be found in [81]. The rest of this chapter is organized as follows: First, in Section 4.1, we describe the circuit model that we have assumed for wires. Next in Section 4.2, we describe the circuit models and sizing techniques for the different circuits within a mat, and in Section 4.3, we describe them for the circuits used in the different H-tree networks.

4.1 Wire Modeling

Wires are considered to belong to one of two types: ideal or non-ideal. Ideal wires are assumed to have zero resistance and capacitance. Non-ideal wires are assumed to have finite resistance and capacitance and are modeled using a one-section Π RC model shown in Figure 4.1. In this figure, \( R_{\text{wire}} \) and \( C_{\text{wire}} \) for a wire of length \( L_{\text{wire}} \) are given by the following equations:
For computation of $R_{\text{unit-length-wire}}$ and $C_{\text{unit-length-wire}}$, we use the equations presented in [36, 38] which are reproduced below. Figure 4.2 shows the accompanying picture for the capacitance model from [36].

\begin{align*}
R_{\text{wire}} &= L_{\text{wire}} R_{\text{unit-length-wire}} \\
C_{\text{wire}} &= L_{\text{wire}} C_{\text{unit-length-wire}}
\end{align*}

\begin{align*}
R_{\text{unit-length-wire}} &= \alpha_{\text{scatter}} \frac{\rho}{(\text{thickness} - \text{barrier} - \text{dishing})(\text{width} - 2 \times \text{barrier})} \quad (4.3) \\
C_{\text{unit-length-wire}} &= \varepsilon_0 \left(2M \varepsilon_{\text{horiz}} \frac{\text{thickness}}{\text{spacing}} + 2 \varepsilon_{\text{vert}} \frac{\text{width}}{\text{ILD}_{\text{thick}}} \right) + \text{fringe}(\varepsilon_{\text{horiz}}, \varepsilon_{\text{vert}}) \quad (4.4)
\end{align*}

4.2 Mat Circuits

As described earlier in Section 3.1, a mat is composed of entities such as the pre-decoding/decoding logic, memory cell array and bitline peripheral circuitry. We present circuits, models and sizing techniques for these entities.

4.2.1 Predecoder and Decoder

As mentioned in Chapter 2, new circuit structures have been adopted for the decoding logic. The same decoding logic circuit structures are utilized for producing the row-
decode signals and the select signals of the bitline and sense amplifier muxes. In the discussion here, we focus on the row-decoding logic. In order to describe the circuit structures assumed within the different entities of the row-decoding logic, we use an illustrative example. Figure 4.3 shows the structure of the row-decoding logic for a subarray with 1024 rows. The row-decoding logic is composed of two row-predecode blocks and the row-decode gates and drivers. The row-predecode blocks are responsible for predecoding the address bits and generating predecoded signals. The row-decode gates and drivers are responsible for decoding the predecoded outputs and driving the wordline load. Each row-predecode block can predecode a maximum of 9 bits and has a 2-level logic structure. With 1024 rows, the number of address bits required for row-decoding is 10. Figure 4.4 shows the structure of each row predecode block for a subarray with 1024 rows. Each row predecode block is responsible for predecoding 5 address bits and each of them generates 32 predecoded output bits. Each predecode block has two levels. The first level is composed of one 2-4 decode unit and one 3-8 decode unit. At the second level, the 4 outputs from the 2-4 decode unit and the 8 outputs from the 3-8 decode unit are combined together using 32 NAND2 gates in order to produce the 32 predecoded outputs. The 32 predecoded outputs from each predecode block are combined together using the 1024 NAND2 gates to generate the row decode signals.
Figure 4.3. Structure of the row decoding logic for a subarray with 1024 rows.

Figure 4.5 shows the circuit paths in the decoding logic for the subarray with 1024 rows. One of the paths contains the NAND2 of the 2-4 decode unit and the other contains the NAND3 gate of the 3-8 decode unit. Each path has 3 stages in its path. The branching efforts at the outputs of the first two stages are also shown in the figure. The predecode output wire is treated as a non-ideal wire with its $R_{\text{predec-out-wire}}$ and $C_{\text{predec-out-wire}}$ computed using the following equations:

\[
R_{\text{predec-output-wire}} = L_{\text{predec-output-wire}} R_{\text{unit-length-wire}} \tag{4.5}
\]
\[
C_{\text{predec-output-wire}} = L_{\text{predec-output-wire}} C_{\text{unit-length-wire}} \tag{4.6}
\]

where $L_{\text{predec-output-wire}}$ is the maximum length amongst lengths of predecode output wires.
Figure 4.4. Structure of the row predecode block for a subarray with 1024 rows.

The sizing of gates in each circuit path is calculated using the method of logical effort. In each of the 3 stages of each circuit path, minimum-size transistors are assumed at the input of the stage and each stage is sized independent of each other using the method of logical effort. While this is not optimal from a delay point of view, it is simpler to model and has been found to be a good sizing heuristic from an energy-delay point of view [11].

In this example that we considered for decoding logic of a subarray with 1024 rows, there were two different circuit paths, one involving the NAND2 gate and another involving the NAND3 gate. In the general case, when each predecode block decodes different number of address bits, a maximum of four circuit paths may exist. When the degree of decoding is low, some of the circuit blocks shown in Figure 4.3 may not be required. For example, Figure 4.6 shows the decoding logic for a subarray with 8 rows. In this case, the decoding logic simply involves a 3-8 decode unit as shown.
Figure 4.5. Row decoding logic circuit paths for a subarray with 1024 rows. One of the circuit paths contains the NAND2 gate of the 2-4 decode unit while the other contains the NAND3 gate of the 3-8 decode unit.

As mentioned before, the same circuit structures used within the row-decoding logic are also used for generating the select signals of the bitline and sense amplifier muxes. However, unlike the row-decoding logic in which the NAND2 decode gates and drivers are assumed to be placed on the side of subarray, the NAND2 decode gates and drivers are assumed to be placed at the center of the mat near their corresponding predecode blocks. Also, the resistance/capacitance of the wires between the predecode blocks and the decode gates are not modeled and are assumed to be zero.

4.2.2 Memory Cell

Figure 4.7 shows the circuit assumed for a 1-ported SRAM cell. The transistors of the SRAM cell are sized based on the widths specified in [39] and are presented in Chapter
3-8 decoder

Figure 4.6. Structure of the row-decoding logic for a subarray with 8 rows. The row-decoding logic is simply composed of 8 decode gates and drivers.

Figure 4.7. 1-port 6T SRAM cell

4.2.3 Bitline Peripheral Circuitry

Sense Amplifier

Figure 4.8 shows the circuit assumed for a sense amplifier - it’s a clocked latch-based sense amplifier. When the ENABLE signal is not activated, there is no flow of current through the transistors of the latch. When the ENABLE signal is activated the sensing begins. The isolation transistors are responsible for isolating the high capacitance of the bitlines from the sense amplifier nodes during the sensing operation.
Figure 4.8. Clocked latch-based sense amplifier

Figure 4.9. Small-signal model of the latch-based sense amplifier [35].

[35] presents analysis of the sense amplifier circuit of 4.8 and equations for sensing delay under different assumptions. Figure 4.9 shows one of the small-signal models presented in [35]. Use of this small-signal model is based on two assumptions:

1. Current has been flowing in the circuit for a sufficiently long time; and
2. The equilibrating device can be modeled as an ideal switch.

For the small-signal model of Figure 4.9, it has been shown that the delay of the sensing operation is given by the following equation:
\[ T_{\text{sense}} = \frac{C_{\text{sense}}}{G_m} \ln \left( \frac{V_{DD}}{V_{\text{sense}}} \right) \] (4.7)

\[ G_m = g_{mn} + g_{mp} \] (4.8)

Use of Equation 4.7 for calculation of sense amplifier delay requires that the values of \( g_{mn} \) (NMOS transconductance) and \( g_{mp} \) (PMOS transconductance) be known. We assume that the transistors in the sense amplifier latch exhibit short-channel effects. For a transistor that exhibits short-channel effect, we use the following typical current equation [69] for computation of saturation current:

\[ I_{\text{dsat}} = \frac{\mu_{\text{eff}}}{2} C_{\text{ox}} \frac{W}{L} (V_{GS} - V_{\text{TH}}) V_{\text{dsat}} \] (4.9)

Differentiating the above equation with respect to \( V_{GS} \) gives the equation for \( g_m \) of the transistor. It can be seen that because of short-channel effect, \( g_m \) comes out to be independent of \( V_{GS} \).

\[ g_m = \frac{\mu_{\text{eff}}}{2} C_{\text{ox}} \frac{W}{L} V_{\text{dsat}} \] (4.10)

Bitline and Sense Amplifier Muxes

Figure 4.10 shows the circuit assumed for the bitline and sense amplifier muxes. We assume that the mux is implemented using NMOS pass transistors. The use of NMOS transistors implies that the output of the mux needs to be precharged high in order to avoid degraded ones. We do not attempt to size the transistors in the muxes and instead assume (as in [10]) fixed widths for the NMOS transistors across all partitions of the array.
Figure 4.10. NMOS-based mux. The output is assumed to be precharged high.

Precharge and Equalization Circuitry

Figure 4.11 shows the circuit assumed for precharging and equalizing the bitlines. The bitlines are assumed to be precharged to \( V_{DD} \) through the PMOS transistors. Just like the transistors in the bitline and sense amp muxes, we do not attempt to size the precharge and equalization transistors and instead assume fixed-width transistors across different partitions of the array.

Bitlines Read Path Circuit Model Figure 4.12 shows the circuit model for the bitline read path between the memory cell and the sense amplifier mux.

4.3 Routing Networks

As described earlier in Section 3.2, address and data are routed to and from the mats on H-tree distribution networks. First, address/data are routed on an H-tree from array
edge to bank edge and then on another H-tree from bank edge to the mats.

4.3.1 Array Edge to Bank Edge H-tree

Figure 3.5 showed the layout of H-tree distribution of address and data between the array edge and the banks. This H-tree network is assumed to be composed of inverter-based repeaters. The sizing of the repeaters and the separation distance between them is determined based on the formulae given in [36]. In order to allow for energy-delay tradeoffs in the repeater design, we introduce an user-controlled variable max_repeater_delay_constraint. A max_repeater_delay_constraint of zero results in the best delay repeater solution. For a
Figure 4.13. Circuit path of address/datain H-trees within a bank.

max_repeater_delay_constraint of 10%, the delay of the path is allowed to get worse by a maximum of 10% with respect to the best delay repeater solution by reducing the sizing and increasing the separation distance. Thus, with the max_repeater_delay_constraint, limited energy savings are possible at the expense of delay.

4.3.2 Bank Edge to Mat H-tree

Figures 3.6 and 3.7 showed layout examples of horizontal and vertical H-trees within a bank, each with 3 nodes. We assume that drivers are placed at each of the nodes of these H-trees. Figure 4.13 shows the circuit path and driver circuit structure of the address/datain H-trees, and Figure 4.14 shows the circuit path and driver circuit structure of the vertical dataout H-tree. In order to allow for signal-gating in the address/datain H-trees we consider multi-stage buffers with a 2-input NAND gate as the input stage. The sizing and number of gates at each node of the H-trees is computed using the methodology described in [11] which takes into account the resistance and capacitance of the intermediate wires in the H-tree.

One problem with the circuit paths of Figures 4.13 and 4.14 is that they start experiencing increased wire delays as the wire lengths between the drivers start to get long. This also limits the maximum random cycle time that can be achieved for the array. So, as an alternative to modeling drivers only at H-tree branching nodes, we also consider an alternative model in which the H-tree circuit paths within a bank are composed of buffers
at regular intervals (i.e. repeaters). With repeaters, the delay through the H-tree paths within a bank can be reduced at the expense of increased power consumption. Figure 4.15 shows the different types of buffer circuits that have been modeled in the H-tree path. At the branches of the H-tree, we again assume buffers with a NAND gate in the input stage in order to allow for signal-gating whereas in the H-tree segments between two nodes, we model inverter-based buffers. We again size these buffers according to the buffer sizing formulae given in [36]. The max_repeater_delay_constraint that was described in Section 4.3.1 is also used here to decide the sizing of the buffers and their separation distance so that delay in these H-trees also may be traded off for potential energy savings.
Figure 4.15. Different types of buffer circuit stages that have been modeled in the H-trees within a bank.
In this chapter, we present the analytical equations used in CACTI-D to compute area, delay, and energy metrics associated with an array. The equations presented in this chapter are suitable for embedded SRAM arrays. In Chapter 7, we describe how some of the equations presented here are modified for embedded DRAM arrays, and in Chapter 8, we describe how they are extended for cache and main memory chip organizations.

5.1 Area Modeling

5.1.1 Gate Area Model

A new area model has been used to estimate the areas of transistors and gates such as inverter, NAND and NOR gates. This area model is based on a layout model presented in [97] which describes a fast technique to estimate standard cell characteristics before the cells are actually laid out. Figure 5.1 illustrates the layout model that has been used in [97]. Table 5.1 shows the process/technology input parameters required by this gate area model. For a thorough description of the technique, please refer to [97]. Gates with stacked transistors are assumed to have a layout similar to that described in [95]. When a transistor width exceeds a certain maximum value ($H_{n\text{-diff}}$ for NMOS and $H_{p\text{-diff}}$ for PMOS in Table 5.1), the transistor is assumed to be folded. This maximum value can either be process-specific or context-specific. An example of when a context-specific
Figure 5.1. Layout model assumed for gates [97].

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$H_{n\text{-diff}}$</td>
<td>Maximum height of n diffusion of a transistor</td>
</tr>
<tr>
<td>$H_{p\text{-diff}}$</td>
<td>Maximum height of p diffusion for a transistor</td>
</tr>
<tr>
<td>$H_{\text{gap-bet-same-diffs}}$</td>
<td>Minimum gap between diffusions of the same type</td>
</tr>
<tr>
<td>$H_{\text{gap-bet-opp-diffs}}$</td>
<td>Minimum gap between n and p diffusions</td>
</tr>
<tr>
<td>$H_{\text{power-rail}}$</td>
<td>Height of $V_{DD}$ (GND) power rail</td>
</tr>
<tr>
<td>$W_p$</td>
<td>Minimum width of poly (poly half-pitch or process feature size)</td>
</tr>
<tr>
<td>$S_{p-p}$</td>
<td>Minimum poly-to-poly spacing</td>
</tr>
<tr>
<td>$W_c$</td>
<td>Contact width</td>
</tr>
<tr>
<td>$S_{p-c}$</td>
<td>Minimum poly-to-contact spacing</td>
</tr>
</tbody>
</table>

width would be used is in case of memory sense amplifiers which typically have to be laid out at a certain pitch.

Given the width of an NMOS transistor, $W_{\text{before-folding}}$, the number of folded transistors may be calculated as follows:

$$N_{\text{folded-transistors}} = \left\lceil \frac{W_{\text{before-folding}}}{H_{n\text{-diff}}} \right\rceil$$  (5.1)
The equation for total diffusion width of $N_{\text{stacked}}$ transistors when they are not folded is given by the following equation:

$$\text{total-diff-width} = 2(W_c + 2S_{p-c}) + N_{\text{stacked}}W_p + (N_{\text{stacked}} - 1)S_{p-p} \quad (5.2)$$

The equation for total diffusion width of $N_{\text{stacked}}$ transistors when they are folded is given by the following equation:

$$\text{total-diff-width} = N_{\text{folded-transistors}}(2(W_c + 2S_{p-c}) + N_{\text{stacked}}W_p + (N_{\text{stacked}} - 1)S_{p-p}) \quad (5.3)$$

Note that Equation 5.3 is a generalized form of the equations used for calculating diffusion width (for computation of drain capacitance) in the original CACTI report [95]. Earlier versions of CACTI assumed at most two folded transistors; in CACTI-D, we allow the degree of folding to be greater than 2 and make the associated layout and area models more general. Note that drain capacitance calculation in CACTI-D makes use of equations similar to 5.2 and 5.3 for computation of diffusion width.

The height of a gate is calculated using the following equation:

$$H_{\text{gate}} = H_{\text{n-diff}} + H_{\text{p-diff}} + H_{\text{gap-bet-opp-diffs}} + 2H_{\text{power-rail}} \quad (5.4)$$

5.1.2 Area Model Equations

The area of the data array is estimated based on the area occupied by a single bank and the area spent in routing address and data to the banks. It is assumed that the area spent in routing address and data to the bank is decided by the pitch of the routed wires. Figures 5.2 and 5.3 show two example arrays with 8 and 16 banks respectively; we present equations for the calculation of the areas of these arrays.
Figure 5.2. Supporting figure for example area calculation of array with 8 banks.

Figure 5.3. Supporting figure for example area calculation of array with 16 banks.

\[ A_{\text{data-arr}} = H_{\text{data-arr}}W_{\text{data-arr}} \]  \hspace{1cm} (5.5)

The pitch of wires routed to the banks is given by the following equation:

\[ P_{\text{all-wires}} = P_{\text{wire}}N_{\text{wires-routed-to-banks}} \]  \hspace{1cm} (5.6)
For the data array of Figure 5.2 with 8 banks, the relevant equations are as follows:

\[
W_{\text{data-arr}} = 4W_{\text{bank}} + P_{\text{all-wires}} + \frac{2P_{\text{all-wires}}}{4} \quad (5.7)
\]

\[
H_{\text{data-arr}} = 2H_{\text{bank}} + \frac{P_{\text{all-wires}}}{2} \quad (5.8)
\]

\[
N_{\text{wires-routed-to-banks}} = 8(N_{\text{bank-addr-bits}} + N_{\text{bank-datain-bits}} + N_{\text{bank-dataout-bits}} + N_{\text{way-select-signals}}) \quad (5.9)
\]

For the data array of Figure 5.3 with 16 banks, the relevant equations are as follows:

\[
W_{\text{data-arr}} = 4W_{\text{bank}} + \frac{P_{\text{all-wires}}}{2} + \frac{2P_{\text{all-wires}}}{8} \quad (5.10)
\]

\[
H_{\text{data-arr}} = 4H_{\text{bank}} + P_{\text{all-wires}} + \frac{2P_{\text{all-wires}}}{4} \quad (5.11)
\]

\[
N_{\text{wires-routed-to-banks}} = 16(N_{\text{bank-addr-bits}} + N_{\text{bank-datain-bits}} + N_{\text{bank-dataout-bits}} + N_{\text{way-select-signals}}) \quad (5.12)
\]

The banks in a data array are assumed to be placed in such a way that the number of banks in the horizontal direction is always either equal to or twice the number of banks in the vertical direction. The height and width of a bank is calculated by computing the area occupied by the mats and the area occupied by the routing resources of the horizontal and vertical H-tree networks within a bank. We again use an example to illustrate the calculations. Figures 3.6 and 3.7 showed the layouts of horizontal and vertical H-trees within a bank. The horizontal and vertical H-trees were each shown to have three branching nodes (H0, H1 and H2; V0, V1 and V2). Combined together, these horizontal and vertical H-trees may be considered as H-trees within a bank with 4 subbanks and 4 mats in each subbank. We present area model equations for such a bank.

\[
A_{\text{bank}} = H_{\text{bank}}W_{\text{bank}} \quad (5.13)
\]
In CACTI-D, as described in Section 4.3, for the H-trees within a bank we assume that drivers are placed either only at the branching nodes of the H-trees or that there are buffers at regular intervals in the H-tree segments. When drivers are present only at the branching nodes of the vertical H-trees within a bank, we consider two alternative models in accounting for area overhead of the vertical H-trees. In the first model, we consider that wires of the vertical H-trees may traverse over memory cell area; in this case, the area overhead caused by the vertical H-trees is in terms of area occupied by drivers which are placed between the mats. In the second model, we do not assume that the wires traverse over the memory cell area and instead assume that they occupy area besides the mats. The second model is also applicable when there are buffers at regular intervals in the H-tree segments. The equations that we present next for area calculation of a bank assume the second model i.e. the wires of the vertical H-trees are assumed to not pass over the memory cell area. The equations for area calculation under the assumption that the vertical H-tree wires go over the memory cell area are quite similar. For our example bank with 4 subbanks and 4 mats in each subbank, the height of the bank is calculated to be equal to the sum of heights of all subbanks plus the height of the routing resources of the horizontal H-tree.

\[ H_{\text{bank}} = 4H_{\text{mat}} + H_{\text{hor-htree}} \]  

(5.14)

The width of the bank is calculated to be equal to the sum of widths of all mats in a subbank plus the width of the routing resources of the vertical H-trees.

\[ W_{\text{bank}} = 4(W_{\text{mat}} + W_{\text{ver-htree}}) \]  

(5.15)

The height of the horizontal H-tree is calculated as the height of the area occupied by the wires in the H-tree. These wires include the address, way-select, datain, and dataout
Figure 5.4. Layout assumed for wires of the horizontal H-tree within a bank.

signals. Figure 5.4 illustrates the layout that we assume for the wires of the horizontal H-tree. We assume that the wires are laid out using a single layer of metal. The height of the area occupied by the wires can be calculated simply by finding the total pitch of all wires in the horizontal H-tree. Figure 5.5 illustrates the layout style assumed for the vertical H-tree wires, and is similar to that assumed for the horizontal H-tree wires. Again the width of the area occupied by a vertical H-tree can be calculated by finding the total pitch of all wires in the vertical H-tree.

\[
H_{\text{hor-tree}} = P_{\text{hor-tree-wires}} \tag{5.16}
\]

\[
W_{\text{ver-tree}} = P_{\text{ver-tree-wires}} \tag{5.17}
\]

The height and width of a mat are estimated using the following equations. Figure 5.6 shows the layout of a mat and illustrates the assumptions made in the following equations. We assume that half of the address, way-select, datain and dataout signals enter the mat from its left and the other half enter from the right.

\[
W_{\text{mat}} = \frac{H_{\text{mat}}W_{\text{initial-mat}} + A_{\text{mat-center-circuitry}}}{W_{\text{initial-mat}}} \tag{5.18}
\]

\[
H_{\text{mat}} = 2H_{\text{subarr-mem-cell-area}} + H_{\text{mat-non-cell-area}} \tag{5.19}
\]

\[
W_{\text{initial-mat}} = 2W_{\text{subarr-mem-cell-area}} + W_{\text{mat-non-cell-area}} \tag{5.20}
\]
Figure 5.5. Layout assumed for wires of the vertical H-tree within a bank.

Figure 5.6. Layout of a mat.

\[
A_{\text{mat-center-circuitry}} = A_{\text{row-predec-block-1}} + A_{\text{row-predec-block-2}} \\
+ A_{\text{bit-mux-predec-block-1}} + A_{\text{bit-mux-predec-block-2}} \\
+ A_{\text{senseamp-mux-predec-block-1}} + A_{\text{senseamp-mux-predec-block-2}} + \\
A_{\text{bit-mux-dec-drivers}} + A_{\text{senseamp-mux-dec-drivers}} \tag{5.21}
\]
\[ H_{\text{subarr-mem-cell-area}} = N_{\text{subarr-rows}}H_{\text{mem-cell}} \]  
\[ W_{\text{subarr-mem-cell-area}} = N_{\text{subarr-cols}}W_{\text{mem-cell}} + \frac{N_{\text{subarr-cols}}}{N_{\text{mem-cells-per-wordline-stitch}}} W_{\text{wordline-stitch}} + \left\lfloor \frac{N_{\text{subarr-cols}}}{N_{\text{bits-per-ecc-bit}}} \right\rfloor W_{\text{mem-cell}} \]  
\[ H_{\text{mat-non-cell-area}} = 2H_{\text{subarr-bitline-peri-circ}} + H_{\text{hor-wires-within-mat}} \]  
\[ H_{\text{hor-wires-within-mat}} = H_{\text{bit-mux-sel-wires}} + H_{\text{senseamp-mux-sel-wires}} + H_{\text{write-mux-sel-wires}} + \frac{H_{\text{number-mat-addr-bits}}}{2} + \frac{H_{\text{number-way-select-signals}}}{2} + \frac{H_{\text{number-mat-datain-bits}}}{2} + \frac{H_{\text{number-mat-dataout-bits}}}{2} \]  
\[ W_{\text{mat-non-cell-area}} = \max(2W_{\text{subarr-row-decoder}}, W_{\text{row-predec-out-wires}}) \]  
\[ H_{\text{subarr-bitline-peri-cir}} = H_{\text{bit-mux}} + H_{\text{senseamp-mux}} + H_{\text{bitline-pre-eq}} + H_{\text{write-driver}} \]

Note that the width of the mat is computed as in Equation 5.18 because we optimistically assume that the circuitry laid out at the center of the mat does not lead to white space in the mat. The areas of lower-level circuit blocks such as the bitline and sense amplifier muxes and write drivers are calculated using the area model that was described in Section 5.1.1 while taking into account pitch-matching constraints.

When redundancy in mats is also considered, the following area contribution due to redundant mats is added to the area of the data array computed in Equation 5.5.

\[ A_{\text{redundant-mats}} = N_{\text{redundant-mats}}A_{\text{mat}} \]  
\[ N_{\text{redundant-mats}} = \left\lfloor \frac{N_{\text{banks}}}{N_{\text{mats}}}N_{\text{mats-per-redundant-mat}} \right\rfloor \]

where \( N_{\text{mats-per-redundant-mat}} \) is the number of mats per redundant mat that and is set to 8 by default. The final height of the data array is readjusted under the optimistic assumption that the redundant mats do not cause any white space in the data array.
\[ H_{\text{data-arr}} = \frac{A_{\text{data-arr}}}{W_{\text{data-arr}}} \] (5.30)

5.2 Delay Modeling

In this section we present equations used in CACTI-D to calculate access time and random cycle time of a memory array.

5.2.1 Access Time Equations

\[ T_{\text{access}} = T_{\text{request-network}} + T_{\text{mat}} + T_{\text{reply-network}} \] (5.31)

\[ T_{\text{request-network}} = T_{\text{arr-edge-to-bank-edge-htree}} + T_{\text{bank-addr-din-hor-htree}} + T_{\text{bank-addr-din-ver-htree}} \] (5.32)

\[ T_{\text{mat}} = \max(T_{\text{row-decoder-path}}, T_{\text{bit-mux-decoder-path}}, T_{\text{senseamp-mux-decoder-path}}) \] (5.33)

\[ T_{\text{reply-network}} = T_{\text{bank-dout-ver-htree}} + T_{\text{bank-dout-hor-htree}} + T_{\text{bank-edge-to-arr-edge}} \] (5.34)

The critical path in the mat usually involves the wordline and bitline access. However, Equation 5.33 also must include a max with the delays of the bitline mux decoder and sense amp mux decoder paths as these circuits operate in parallel with the row decoding logic, and in general may act as the critical path for certain partitions of the data array. Usually when that happens, the number of rows in the subarray would be too few and the partitions would not get selected.
\[ T_{\text{row-decoder-path}} = T_{\text{row-predec}} + T_{\text{row-dec-driver}} + T_{\text{bitline}} + T_{\text{bitline-mux}} + T_{\text{senseamp}} \] (5.35)

\[ T_{\text{bit-mux-decoder-path}} = T_{\text{bit-mux-predec}} + T_{\text{bit-mux-dec-driver}} + T_{\text{senseamp}} \] (5.36)

\[ T_{\text{senseamp-mux-decoder-path}} = T_{\text{senseamp-mux-predec}} + T_{\text{senseamp-mux-dec-driver}} \] (5.37)

\[ T_{\text{row-predec}} = \max(T_{\text{row-predec-blk-1-nand2-path}}, T_{\text{row-predec-blk-1-nand3-path}}, T_{\text{row-predec-blk-2-nand2-path}}, T_{\text{row-predec-blk-2-nand3-path}}) \] (5.38)

\[ T_{\text{bit-mux-sel-predec}} = \max(T_{\text{bit-mux-sel-predec-blk-1-nand2-path}}, T_{\text{bit-mux-sel-predec-blk-1-nand3-path}}, T_{\text{bit-mux-sel-predec-blk-2-nand2-path}}, T_{\text{bit-mux-sel-predec-blk-2-nand3-path}}) \] (5.39)

\[ T_{\text{senseamp-mux-sel-predec}} = \max(T_{\text{senseamp-mux-sel-predec-blk-1-nand2-path}}, T_{\text{senseamp-mux-sel-predec-blk-1-nand3-path}}, T_{\text{senseamp-mux-sel-predec-blk-2-nand2-path}}, T_{\text{senseamp-mux-sel-predec-blk-2-nand3-path}}) \] (5.40)

The calculation for bitline delay is based on the model described in [94]. The model considers the effect of the wordline rise time by considering the slope (m) of the wordline signal.

\[ T_{\text{bitline}} = \begin{cases} \sqrt{2T_{\text{step}} \frac{V_{\text{DD}} - V_{\text{th}}}{m}} & \text{if } T_{\text{step}} \leq 0.5\frac{V_{\text{DD}} - V_{\text{th}}}{m} \\ T_{\text{step}} + \frac{V_{\text{DD}} - V_{\text{th}}}{2m} & \text{if } T_{\text{step}} > 0.5\frac{V_{\text{DD}} - V_{\text{th}}}{m} \end{cases} \] (5.41)
\[ T_{\text{step}} = \tau \ln \left( \frac{V_{\text{DD}}}{V_{\text{DD}} - V_{\text{bitline-swing}}} \right) \quad (5.42) \]

\[ \tau = \left( R_{\text{cell-pull-down}} + R_{\text{cell-acc}} \right) \left( C_{\text{bitline}} + 2C_{\text{drain-bit-mux}} + 2C_{\text{drain-iso}} + C_{\text{sense}} + 
C_{\text{drain-senseamp-mux}} \right) + R_{\text{bitline}} \left( \frac{C_{\text{bitline}}}{2} + 2C_{\text{drain-bit-mux}} + 2C_{\text{drain-iso}} + C_{\text{sense}} + 
C_{\text{drain-senseamp-mux}} \right) + R_{\text{bit-mux}} \left( C_{\text{drain-bit-mux}} + 2C_{\text{drain-iso}} + C_{\text{sense}} + 
C_{\text{drain-senseamp-mux}} \right) + R_{\text{iso}} \left( C_{\text{drain-iso}} + C_{\text{sense}} + C_{\text{drain-senseamp-mux}} \right) \quad (5.43) \]

The calculation of sense amplifier delay makes use of the model described in [35].

\[ T_{\text{senseamp}} = \tau \ln \left( \frac{V_{\text{DD}}}{V_{\text{senseamp}}} \right) \quad (5.44) \]
\[ \tau = \frac{C_{\text{sense}}}{G_m} \quad (5.45) \]

### 5.2.2 Random Cycle Time Equations

Typically, the random cycle time of an SRAM would be limited by wordline and bitline delays. In order to come up with an equation for lower bound on random cycle time, we consider that the SRAM is potentially pipelineable with placement of latches at appropriate locations.

\[ T_{\text{random-cycle}} = \max \left( T_{\text{row-dec-driver}} + T_{\text{bitline}} + T_{\text{senseamp}} + T_{\text{wordline-reset}} + 
\max \left( T_{\text{bitline-precharge}}, T_{\text{bit-mux-out-precharge}}, T_{\text{senseamp-mux-out-precharge}} \right), 
T_{\text{between-buffers-bank-hor-htree}}, T_{\text{between-buffers-bank-ver-dataout-htree}}, T_{\text{row-predec-blk}}, 
T_{\text{bit-mux-predec}} + T_{\text{bit-mux-dec-driver}}, 
T_{\text{senseamp-mux-predec}} + T_{\text{senseamp-mux-dec-driver}} \right) \quad (5.46) \]

We come up with an estimate for the wordline reset delay by assuming that the wordline discharges through the NMOS transistor of the final inverter in the wordline driver.
\[ T_{\text{wordline-reset}} = \ln\left( \frac{V_{DD} - 0.1V_{DD}}{V_{DD}} \right) \left( R_{\text{final-inv-wordline-driver}} C_{\text{wordline}} + \frac{R_{\text{final-inv-wordline-driver}} C_{\text{wordline}}}{2} \right) \] (5.47)

\[ T_{\text{bitline-precharge}} = \ln\left( \frac{V_{DD} - 0.1V_{\text{bitline-swing}}}{V_{DD} - V_{\text{bitline-swing}}} \right) \left( R_{\text{bit-pre}} C_{\text{bitline}} + \frac{R_{\text{bitline}} C_{\text{bitline}}}{2} \right) \] (5.48)

\[ T_{\text{bit-mux-out-precharge}} = \ln\left( \frac{V_{DD} - 0.1V_{\text{bitline-swing}}}{V_{DD} - V_{\text{bitline-swing}}} \right) \left( R_{\text{bit-mux-pre}} C_{\text{bit-mux-out}} + \frac{R_{\text{bit-mux-out}} C_{\text{bit-mux-out}}}{2} \right) \] (5.49)

\[ T_{\text{senseamp-mux-out-precharge}} = \ln\left( \frac{V_{DD} - 0.1V_{\text{bitline-swing}}}{V_{DD} - V_{\text{bitline-swing}}} \right) \left( R_{\text{senseamp-mux-pre}} C_{\text{senseamp-mux-out}} + \frac{R_{\text{senseamp-mux-out}} C_{\text{senseamp-mux-out}}}{2} \right) \] (5.50)

### 5.3 Energy Modeling

In this section, we present the equations used in CACTI-D to calculate dynamic energy and leakage power of a data array. We present equations for dynamic read energy; the equations for dynamic write energy are similar.

#### 5.3.1 Calculation of Dynamic Energy

**Dynamic Energy Calculation Example for a CMOS Gate Stage**

We present a representative example to illustrate how we calculate the dynamic energy for a CMOS gate stage. Figure 5.7 shows a CMOS gate stage composed of a NAND2 gate followed by an inverter which drives the load. The energy consumption of this circuit is given by:
Figure 5.7. A simple CMOS gate stage composed of a NAND2 followed by an inverter which is driving a load.

\[
E_{\text{dyn}} = E_{\text{dyn-nand2}} + E_{\text{dyn-inv}} \quad (5.51)
\]

\[
E_{\text{dyn-nand2}} = 0.5(C_{\text{intrinsic-nand2}} + C_{\text{gate-inv}})V_{\text{DD}}^2 \quad (5.52)
\]

\[
E_{\text{dyn-inv}} = 0.5(C_{\text{intrinsic-inv}} + C_{\text{gate-load-next-stage}} + C_{\text{wire-load}})V_{\text{DD}}^2 \quad (5.53)
\]

\[
C_{\text{intrinsic-nand2}} = \text{draincap}(\text{nand2}, W_{\text{nand-pmos}}, W_{\text{nand-nmos}}) \quad (5.54)
\]

\[
C_{\text{gate-inv}} = \text{gatecap}(\text{inv}, W_{\text{inv-pmos}}, W_{\text{inv-nmos}}) \quad (5.55)
\]

\[
C_{\text{drain-inv}} = \text{draincap}(\text{inv}, W_{\text{inv-pmos}}, W_{\text{inv-nmos}}) \quad (5.56)
\]

The multiplicative factor of 0.5 in the equations of \( E_{\text{dyn-nand2}} \) and \( E_{\text{dyn-inv}} \) assumes consecutive charging and discharging cycles for each gate. Energy is consumed only during the charging cycle of a gate when its output goes from low to high.

Dynamic Energy Equations

The dynamic energy per read access consumed in the data array is the sum of the dynamic energy consumed in the mats and that consumed in the request and reply networks during a read access.
\[ E_{\text{dyn-read}} = E_{\text{dyn-read-request-network}} + E_{\text{dyn-read-mats}} + E_{\text{dyn-read-reply-network}} \]  

\[ E_{\text{dyn-read-mats}} = (E_{\text{dyn-predec-blks}} + E_{\text{dyn-decoder-drivers}} + E_{\text{dyn-read-bitlines}} + E_{\text{senseamps}})N_{\text{mats-in-subbank}} \]  

\[ E_{\text{dyn-predec-blks}} = E_{\text{dyn-row-predec-blks}} + E_{\text{dyn-bit-mux-predec-blks}} \]  

\[ E_{\text{dyn-row-predec-blks}} = E_{\text{dyn-row-predec-blk-1-nand2-path}} + E_{\text{dyn-row-predec-blk-1-nand3-path}} + E_{\text{dyn-row-predec-blk-2-nand2-path}} + E_{\text{dyn-row-predec-blk-2-nand3-path}} \]  

\[ E_{\text{dyn-bit-mux-predec-blks}} = E_{\text{dyn-bit-mux-predec-blk-1-nand2-path}} + E_{\text{dyn-bit-mux-predec-blk-1-nand3-path}} + E_{\text{dyn-bit-mux-predec-blk-2-nand2-path}} + E_{\text{dyn-bit-mux-predec-blk-2-nand3-path}} \]  

\[ E_{\text{dyn-senseamp-mux-predec-blks}} = E_{\text{dyn-senseamp-mux-predec-blk-1-nand2-path}} + E_{\text{dyn-senseamp-mux-predec-blk-1-nand3-path}} + E_{\text{dyn-senseamp-mux-predec-blk-2-nand2-path}} + E_{\text{dyn-senseamp-mux-predec-blk-2-nand3-path}} \]  

\[ E_{\text{dyn-decoder-drivers}} = E_{\text{dyn-row-decoder-drivers}} + E_{\text{dyn-bitmux-decoder-driver}} + E_{\text{dyn-senseamp-mux-decoder-driver}} \]  

\[ E_{\text{dyn-row-decoder-drivers}} = 4E_{\text{dyn-mat-row-decoder-driver}} \]  

\[ E_{\text{dyn-read-bitlines}} = N_{\text{subarr-cols}}E_{\text{dyn-read-bitline}} \]  

\[ E_{\text{dyn-read-bitline}} = C_{\text{bitline}}V_{\text{bitline-swing}}V_{\text{DD}} \]  

\[ V_{\text{bitline-swing}} = 2V_{\text{senseamp}} \]  

56
\[ E_{\text{dyn-read-request-network}} = E_{\text{dyn-read-arr-edge-to-bank-edge-request-htree}} + E_{\text{dyn-read-bank-hor-request-htree}} + E_{\text{dyn-read-bank-ver-request-htree}} \]  

\[ (5.68) \]

\[ E_{\text{dyn-read-reply-network}} = E_{\text{dyn-read-bank-ver-reply-htree}} + E_{\text{dyn-read-bank-hor-reply-htree}} + E_{\text{dyn-read-bank-edge-to-arr-edge-reply-htree}} \]  

\[ (5.69) \]

Equation 5.67 assumes that the swing in the bitlines rises up to twice the signal that can be detected by the sense amplifier [10]. \( E_{\text{dyn-read-request-network}} \) and \( E_{\text{dyn-read-reply-network}} \) are calculated by determining the energy consumed in the wires/drivers/repeaters of the H-trees. The energy consumption in the horizontal and vertical H-trees of the request network within a bank for the example 1MB bank discussed in Section 4.3 with 4 subbanks and 4 mats in each subbank may be written as follows (referring to Figures 3.6 and 3.7 in Section 3.2):

\[ E_{\text{dyn-read-bank-hor-request-htree}} = E_{\text{dyn-read-req-network-H0-H1}} + E_{\text{dyn-read-req-network-H1-H2}} + E_{\text{dyn-read-req-network-read-H2-V0}} \]  

\[ (5.70) \]

\[ E_{\text{dyn-read-bank-ver-request-htree}} = E_{\text{dyn-read-req-network-V0-V1}} + E_{\text{dyn-read-req-network-V1-V2}} \]  

\[ (5.71) \]

The energy consumed in the H-tree segments depends on the location of the segment in the H-tree and the number of signals that are transmitted in each segment. In the request network, during a read access, between nodes H0 and H1, a total of 15 (address) signals are transmitted; between node H1 and both H2 nodes, a total of 30 (address) signals are transmitted; between all H2 and V0 nodes, a total of 60 (address) signals are transmitted. In the vertical H-tree, we assume signal-gating so that the address bits are transmitted...
to the mats of a single subbank only; thus, between all V0 and V1 nodes, a total of 56 (address) signals are transmitted; between all V1 and V2 nodes, a total of 52 (address) signals are transmitted.

\[ E_{\text{dyn-read-req-network-H0-H1}} = (15)E_{\text{H0-H1-1-bit}} \] (5.72)
\[ E_{\text{dyn-read-req-network-H1-H2}} = (30)E_{\text{H1-H2-1-bit}} \] (5.73)
\[ E_{\text{dyn-read-req-network-H2-V0}} = (60)E_{\text{H2-V0-1-bit}} \] (5.74)
\[ E_{\text{dyn-read-req-network-V0-V1}} = (56)E_{\text{V0-V1-1-bit}} \] (5.75)
\[ E_{\text{dyn-read-req-network-V1-V2}} = (52)E_{\text{V1-V2-1-bit}} \] (5.76)

The equations for energy consumed in the H-trees of the reply network are similar in form to the above equations. Also, the equations for dynamic energy per write access are similar to the ones that have been presented here for read access. In case of write access, the datain bits are written into the memory cells at full swing of the bitlines.

5.3.2 Calculation of Leakage Power

We estimate the standby leakage power consumed in the array. Our leakage power estimation does not consider the use of any leakage control mechanism in the array. We make use of the methodology presented in [24, 51] to simply provide an estimate of the drain-to-source subthreshold leakage current for all transistors that are off with \( V_{\text{DD}} \) applied across their drain and source.

Leakage Power Calculation for CMOS gates

We illustrate our methodology of calculation of leakage power for the CMOS gates that are used in our modeling. Figure 5.8 illustrates the leakage power calculation for an inverter. When the input is low and the output is high, there is subthreshold leakage through the NMOS transistor whereas when the input is high and the output is low, there
is subthreshold leakage current through the PMOS transistor. In order to simplify our modeling, we come up with a single average leakage power number for each gate. Thus for the inverter, we calculate leakage as follows:

\[ P_{\text{leak-inv}} = \frac{W_{\text{inv-pmos}}I_{\text{off-pmos}} + W_{\text{inv-nmos}}I_{\text{off-nmos}}}{2} V_{\text{DD}} \]  \hspace{1cm} (5.77)

where \( I_{\text{off-pmos}} \) is the subthreshold current per unit width for the PMOS transistor and \( I_{\text{off-nmos}} \) is the subthreshold current per unit width for the NMOS transistor.

Figure 5.9 illustrates the leakage power calculation for a NAND2 gate. When both inputs are high, the output is low and for this condition there is leakage through the PMOS transistors as shown. When either of the inputs is low, the output is high and there is leakage through the NMOS transistors. Because of the stacked NMOS transistors [24,51], this leakage depends on which input(s) is low. The leakage is least when both inputs are low. Under standby operating conditions, for NAND2 and NAND3 gates in the decoding logic within the mats, we assume that the output of each NAND is high (deactivated) with both of its inputs low. Thus we attribute a leakage number to the NAND gate based on the leakage through its stacked NMOS transistors when both inputs are low. We consider the reduction in leakage due to the effect of stacked transistors and calculate leakage for the NAND2 gate as follows:

\[ P_{\text{leak-nand2}} = W_{\text{inv-nmos}}I_{\text{off-nmos}}SF_{\text{nand2}} \]  \hspace{1cm} (5.78)

where \( SF_{\text{nand2}} \) is the stacking fraction for reduction in leakage due to stacking.

Leakage Power Equations

Most of the leakage power equations are similar to the dynamic energy equations in form.
\[ P_{\text{leak}} = P_{\text{leak-request-network}} + P_{\text{leak-mats}} + P_{\text{leak-reply-network}} \]  \hspace{0.5cm} (5.79)

\[ P_{\text{leak-mats}} = (P_{\text{leak-mem-cells}} + P_{\text{leak-predec-blks}} + P_{\text{leak-decoder-drivers}} + \]
\[ P_{\text{leak-senseamps}} ) N_{\text{banks}} N_{\text{subbanks}} N_{\text{mats-in-subbank}} \]  \hspace{0.5cm} (5.80)

\[ P_{\text{leak-mem-cells}} = N_{\text{subarr-rows}} N_{\text{subarr-cols}} P_{\text{mem-cell}} \]  \hspace{0.5cm} (5.81)

\[ P_{\text{leak-decoder-drivers}} = P_{\text{leak-row-decoder-drivers}} + P_{\text{leak-bitmux-decoder-driver}} + \]
\[ P_{\text{leak-senseamp-mux-decoder-driver}} \]  \hspace{0.5cm} (5.82)

\[ P_{\text{leak-row-decoder-drivers}} = 4N_{\text{subarr-rows}} P_{\text{leak-row-decoder-driver}} \]  \hspace{0.5cm} (5.83)

\[ P_{\text{leak-request-network}} = P_{\text{leak-arr-edge-to-bank-edge-request-htree}} + P_{\text{leak-bank-hor-request-htree}} + \]
\[ P_{\text{leak-bank-ver-request-htree}} \]  \hspace{0.5cm} (5.84)
Figure 5.10. Leakage paths in a memory cell in idle state. BIT and BITB are precharged to $V_{DD}$.

\[
P_{\text{leak-reply-network}} = P_{\text{dyn-ver-reply-htree}} + P_{\text{dyn-hor-reply-htree}} + P_{\text{dyn-bank-edge-to-arr-edge-reply-htree}}
\]

Figure 5.10 shows the subthreshold leakage paths in an SRAM cell when it is in idle/standby state [24, 51]. The leakage power contributed by a single memory cell may be given by:

\[
P_{\text{mem-cell}} = V_{DD} I_{\text{mem-cell}}
\]

\[
I_{\text{mem-cell}} = I_{p1} + I_{n2} + I_{n3}
\]

\[
I_{p1} = W_{p1} I_{\text{off-pmos}}
\]

\[
I_{n2} = W_{n2} I_{\text{off-nmos}}
\]

\[
I_{n3} = W_{n2} I_{\text{off-nmos}}
\]

Figure 5.11 shows the subthreshold leakage paths in a sense amplifier during an idle/standby cycle [24, 51].
Figure 5.11. Leakage paths in a sense amplifier in idle state.
CHAPTER 6

TECHNOLOGY MODELING

In CACTI-D we incorporate technology data for four ITRS technology nodes (we use MPU/ASIC metal 1 half-pitch to define the technology node) – 90, 65, 45 and 32 nm – which cover years 2004 to 2013 in the ITRS. Section 6.1 gives more details about the device data and modeling and Section 6.2 gives more details about the interconnect data and modeling.

6.1 Devices

Table 6.1 shows the characteristics of transistors modeled by the ITRS that are incorporated within CACTI-D. We include data for the three device types that the ITRS defines - High Performance (HP), Low Standby Power (LSTP) and Low Operating Power (LOP). The HP transistors are state-of-the-art fast transistors with short gate lengths, thin gate oxides, low $V_{th}$ and low $V_{DD}$ whose CV/I is targeted to improve by 17% every year. As a consequence of their high on-currents, these transistors tend to be very leaky. The LSTP transistors on the other hand are transistors with longer gate lengths, thicker gate oxides, higher $V_{th}$ and higher $V_{DD}$. The gate-lengths of the LSTP transistors lag the HP transistors by 4 years. The LSTP transistors trade off high on-currents for maintenance of an almost constant low leakage of 10pA across the technology nodes. The LOP transistors have performance that lie in between the HP and LSTP transistors. They use the lowest $V_{DD}$ to
TABLE 6.1

TECHNOLOGY CHARACTERISTICS OF TRANSISTORS USED IN THE MODEL

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Meaning</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_{DD}$</td>
<td>Voltage applied between drain and source, gate and source</td>
<td>V</td>
</tr>
<tr>
<td>$L_{gate}$</td>
<td>Physical length of the gate</td>
<td>micron</td>
</tr>
<tr>
<td>$V_{th}$</td>
<td>Saturation threshold voltage</td>
<td>V</td>
</tr>
<tr>
<td>$M_{eff}$</td>
<td>Effective mobility</td>
<td>cm²/V·s</td>
</tr>
<tr>
<td>$V_{dsat}$</td>
<td>Drain saturation voltage</td>
<td>V</td>
</tr>
<tr>
<td>$C_{ox-elec}$</td>
<td>Capacitance of gate-oxide in inversion</td>
<td>F/µ²</td>
</tr>
<tr>
<td>$C_{gd-overlap}$</td>
<td>Gate to drain overlap capacitance</td>
<td>F/µ</td>
</tr>
<tr>
<td>$C_{gd-fringe}$</td>
<td>Gate to drain fringing capacitance</td>
<td>F/µ</td>
</tr>
<tr>
<td>$C_{j-bottom}$</td>
<td>Bottom junction capacitance</td>
<td>F/µ²</td>
</tr>
<tr>
<td>$C_{j-sidewall}$</td>
<td>Sidewall junction capacitance</td>
<td>F/µ²</td>
</tr>
<tr>
<td>$I_{on}$</td>
<td>On-current (saturation)</td>
<td>A/µ</td>
</tr>
<tr>
<td>$I_{off}$</td>
<td>Channel leakage current (for $V_{gate} = 0$ and $V_{drain} = V_{DD}$)</td>
<td>A/µ</td>
</tr>
</tbody>
</table>

control the operating power and their gate-lengths lag those of HP transistors by 2 years.

The CV/I of the LSTP and LOP transistors improves by about 14% every year.

Table 6.2 shows values of key technology metrics of the HP, LSTP and LOP NMOS transistors for four technology nodes. The data is obtained from MASTAR [76] files. According to the 2003 ITRS, the years 2004, 2007, 2010 and 2013 correspond to 90, 65, 45 and 32 nm technology-nodes. Because the 2005 ITRS does not include device data for the 90nm technology-node (year 2004), we obtain this data using MASTAR and targeting the appropriate CV/I. Note that all values shown are for planar bulk devices. The ITRS actually makes the assumption that planar high-performance bulk devices reach their limits of practical scaling in 2012 and therefore includes multiple parallel paths of scaling for SOI (Silicon On Insulator) and multiple-gate MOS transistors such as FinFETs starting from the year 2008 which run in parallel with conventional bulk CMOS scaling; for

---

1Because of ambiguity associated with the “technology-node” term, the 2005 ITRS has discontinued the practice of using the term, however, for the sake of convenience, we continue to use it in CACTI-D.
Table 6.2

VALUES OF KEY TECHNOLOGY METRICS OF HP, LSTP AND LOP NMOS TRANSISTORS FOR FOUR TECHNOLOGY-NODES FROM THE 2005 ITRS [77]

<table>
<thead>
<tr>
<th>Technology-node</th>
<th>90nm</th>
<th>65nm</th>
<th>45nm</th>
<th>32nm</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L_{gate}$ (nm)</td>
<td>37/75/53</td>
<td>25/45/32</td>
<td>18/28/22</td>
<td>13/20/16</td>
</tr>
<tr>
<td>EOT (nm)</td>
<td>1.2/2.2/1.5</td>
<td>1.1/1.9/1.2</td>
<td>0.65/1.4/0.9</td>
<td>0.5/1.1/0.8</td>
</tr>
<tr>
<td>$V_{DD}$ (V)</td>
<td>1.2/1.2/0.9</td>
<td>1.1/1.2/0.8</td>
<td>1/1.1/0.7</td>
<td>0.9/1/0.7</td>
</tr>
<tr>
<td>$V_{th}$ (mV)</td>
<td>237/525/318</td>
<td>195/554/315</td>
<td>181/532/256</td>
<td>218/513/242</td>
</tr>
<tr>
<td>$I_{on}$ ($\mu$A/$\mu$)</td>
<td>1077/465/550</td>
<td>1197/519/573</td>
<td>2047/666/749</td>
<td>2212/684/890</td>
</tr>
<tr>
<td>$I_{off}$ (nA/$\mu$)</td>
<td>32.4/0.008/2.0</td>
<td>196/0.009/4.9</td>
<td>280/0.01/4.0</td>
<td>152/0.021/65</td>
</tr>
<tr>
<td>$C_{ox-elec}$ (fF/$\mu^2$)</td>
<td>17.9/12.2/16.0</td>
<td>18.8/13.6/18.7</td>
<td>37.7/20.1/28.2</td>
<td>41.1/22.9/31.2</td>
</tr>
<tr>
<td>$\tau$ (ps)</td>
<td>1.01/2.98/1.78</td>
<td>0.64/1.97/1.17</td>
<td>0.4/1.33/0.79</td>
<td>0.27/0.9/0.53</td>
</tr>
<tr>
<td>FOI delay (ps)</td>
<td>8.46/29.1/23.7</td>
<td>5.33/19.1/10.8</td>
<td>3.16/12.2/7.1</td>
<td>1.76/8.32/3.66</td>
</tr>
</tbody>
</table>

32nm HP devices we have assumed the use of SOI. For all technology nodes, the overlap capacitance value has been assumed to be 20% of ideal (no overlap) gate capacitance. The bottom junction capacitance value for the planar bulk CMOS transistors has been assumed to be 1fF/$\mu^2$, which is the value that MASTAR assumes. As MASTAR does not model sidewall capacitance, we compute values for sidewall capacitance in the following manner: we use process data provided at the MOSIS website [78] for TSMC and IBM 130/180/250nm processes and compute average of the ratios of sidewall-to-bottom junction capacitances for these processes. We observe that average error in using this average value for projecting sidewall capacitance given bottom junction capacitance is less than 10%. We use this average value in projecting sidewall capacitances for the ITRS processes.
We calculate the drive resistance of a transistor during switching as follows:

\[ R_{\text{on}} = \frac{V_{\text{DD}}}{I_{\text{eff}}} \]  \hspace{1cm} (6.1)

The effective drive current is calculated using the following formula described in [62, 96]:

\[ I_{\text{eff}} = \frac{I_H + I_L}{2} \]  \hspace{1cm} (6.2)

where \( I_H = I_{DS} (V_{GS} = V_{DD}, V_{DS} = \frac{V_{DD}}{2}) \) and \( I_L = I_{DS} (V_{GS} = \frac{V_{DD}}{2}, V_{DS} = V_{DD}) \).

For PMOS transistors, we find the width of the transistor that produces the same \( I_{\text{off}} \) as a unit-width NMOS transistor. Using this width, we compute the PMOS effective drive current \( I_{\text{eff-pmos}} \) and the PMOS-to-NMOS sizing ratio that is used during the application of the method of logical effort:

\[ S_{\text{pmos-to-nmos-logical-effort}} = \frac{I_{\text{eff-nmos}}}{I_{\text{eff-pmos}}} \]  \hspace{1cm} (6.3)

Table 6.3 shows technology data that we have assumed for an SRAM cell.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A_{\text{sram-cell}} ) (Area of an SRAM cell) (( \mu^2 ))</td>
<td>146( F^2 )</td>
<td>[39]</td>
</tr>
<tr>
<td>( W_{\text{sram-cell-acc}} ) (Width of SRAM cell access transistor) (( \mu ))</td>
<td>1.31( F )</td>
<td>[39]</td>
</tr>
<tr>
<td>( W_{\text{sram-cell-pd}} ) (Width of SRAM cell pull-down transistor) (( \mu ))</td>
<td>1.23( F )</td>
<td>[39]</td>
</tr>
<tr>
<td>( W_{\text{sram-cell-pu}} ) (Width of SRAM cell pull-up transistor) (( \mu ))</td>
<td>2.08( F )</td>
<td>[39]</td>
</tr>
<tr>
<td>( A R_{\text{sram-cell}} ) (Aspect ratio of the cell)</td>
<td>1.46</td>
<td>[39]</td>
</tr>
</tbody>
</table>
It may be useful to know that while currently we provide device data for just the three ITRS device types, it is not difficult to incorporate device data from other sources into CACTI-D. Thus, published data of various industrial fabrication processes or data from sources such as [99] may also be utilized. Also, by making use of MASTAR, it is possible to obtain device data for scaling models and assumptions that are different from those of the ITRS. As an example, while the ITRS device data for its High Performance device type is based on an improvement in device CV/I of 17% every year, one may obtain alternative device data by targeting a different CV/I improvement and/or $I_{\text{off}}$. Another example is to start off with the ITRS High Performance device type and use MASTAR to come up with higher Vt or longer channel variations of the base device.

6.2 Wires

Wire characteristics in CACTI-D are based on the projections made in [36, 38]. The approach followed in [36, 38] is to consider both aggressive (optimistic) and conservative (pessimistic) assumptions regarding interconnect technology. The aggressive projections assume aggressive use of low-k dielectrics, insignificant resistance degradation due to dishing and scattering, and tall wire aspect ratios. The conservative projections assume limited use of low-k dielectrics, significant resistance degradation due to dishing and scattering, and smaller wire aspect ratios. For these assumptions, [36, 38] looks at two types of wires, semi-global and global. Wires of semi-global type have a pitch of $4F$ ($F =$ Feature size) whereas wires of global type have a pitch of $8F$. We incorporate the properties of both these wire types into CACTI-D. The values of the semi-global and global wire characteristics under aggressive and conservative assumptions are presented in Table 6.4 for 90/65/45/32 nm technology nodes. The resistance per unit length and capacitance per unit length values are calculated based off Equations 4.3 and 4.4 respectively. For the capacitance per unit micron calculation, we assume a Miller factor of 1.5 as a “realistic
TABLE 6.4

AGGRESSIVE AND CONSERVATIVE WIRE PROJECTIONS FROM [36]

<table>
<thead>
<tr>
<th>Technology-node</th>
<th>90nm</th>
<th>65nm</th>
<th>45nm</th>
<th>32nm</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Common wire characteristics (aggressive/conservative)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(\rho (m\Omega \cdot \mu))</td>
<td>0.022/0.022</td>
<td>0.018/0.022</td>
<td>0.018/0.022</td>
<td>0.018/0.022</td>
</tr>
<tr>
<td>(\varepsilon_f orC_c)</td>
<td>2.709/3.038</td>
<td>2.303/2.734</td>
<td>1.958/2.46</td>
<td>1.664/2.214</td>
</tr>
<tr>
<td><strong>Semi-global wire properties (aggressive/conservative)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pitch (nm)</td>
<td>360</td>
<td>280</td>
<td>180</td>
<td>128</td>
</tr>
<tr>
<td>Aspect ratio</td>
<td>2.4/2.0</td>
<td>2.7/2.0</td>
<td>3.0/2.0</td>
<td>3.0/2.0</td>
</tr>
<tr>
<td>Thickness (nm)</td>
<td>432/400</td>
<td>351/280</td>
<td>270/200</td>
<td>192/140</td>
</tr>
<tr>
<td>ILD (nm)</td>
<td>480/480</td>
<td>405/405</td>
<td>315/315</td>
<td>210/210</td>
</tr>
<tr>
<td>Miller factor</td>
<td>1.5/1.5</td>
<td>1.5/1.5</td>
<td>1.5/1.5</td>
<td>1.5/1.5</td>
</tr>
<tr>
<td>Barrier (nm)</td>
<td>10/8</td>
<td>0/6</td>
<td>0/4</td>
<td>0/3</td>
</tr>
<tr>
<td>Dishing (%)</td>
<td>0/0</td>
<td>0/0</td>
<td>0/0</td>
<td>0/0</td>
</tr>
<tr>
<td>(\alpha_{scatter})</td>
<td>1/1</td>
<td>1/1</td>
<td>1/1</td>
<td>1/1</td>
</tr>
<tr>
<td>Resistance per unit length ((\Omega/\mu))</td>
<td>0.33/0.38</td>
<td>0.34/0.73</td>
<td>0.74/1.52</td>
<td>1.46/3.03</td>
</tr>
<tr>
<td>Capacitance per unit length ((fF/\mu))</td>
<td>0.314/0.302</td>
<td>0.302/0.282</td>
<td>0.291/0.265</td>
<td>0.269/0.254</td>
</tr>
<tr>
<td><strong>Global wire properties (aggressive/conservative)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pitch (nm)</td>
<td>800</td>
<td>560</td>
<td>400</td>
<td>280</td>
</tr>
<tr>
<td>Aspect ratio</td>
<td>2.7/2.2</td>
<td>2.8/2.2</td>
<td>3.0/2.2</td>
<td>3.0/2.2</td>
</tr>
<tr>
<td>Thickness (nm)</td>
<td>1080/880</td>
<td>784/616</td>
<td>600/440</td>
<td>420/308</td>
</tr>
<tr>
<td>ILD (nm)</td>
<td>960/1100</td>
<td>810/770</td>
<td>630/550</td>
<td>420/385</td>
</tr>
<tr>
<td>Miller factor</td>
<td>1.5</td>
<td>1.5</td>
<td>1.5</td>
<td>1.5</td>
</tr>
<tr>
<td>Barrier (nm)</td>
<td>10/8</td>
<td>0/6</td>
<td>0/4</td>
<td>0/3</td>
</tr>
<tr>
<td>Dishing (%)</td>
<td>0/10</td>
<td>0/10</td>
<td>0/10</td>
<td>0/10</td>
</tr>
<tr>
<td>(\alpha_{scatter})</td>
<td>1/1</td>
<td>1/1</td>
<td>1/1</td>
<td>1/1</td>
</tr>
<tr>
<td>Resistance per unit length ((\Omega/\mu))</td>
<td>0.067/0.09</td>
<td>0.095/0.17</td>
<td>0.19/0.36</td>
<td>0.37/0.72</td>
</tr>
<tr>
<td>Capacitance per unit length ((fF/\mu))</td>
<td>0.335/0.315</td>
<td>0.308/0.298</td>
<td>0.291/0.281</td>
<td>0.269/0.267</td>
</tr>
</tbody>
</table>

worst-case” value [73]. For material strength, we assume that low-k dielectrics are not utilized between wire layers as suggested in [73].
6.3 Technology Exploration

As an additional feature in CACTI-D, we allow the user to map different device and wire types to different parts of the array. We divide the devices in the array into two parts: one, devices used in the memory cells and wordline drivers, and two, the rest of the peripheral and global circuitry. Different device types such as the ITRS HP, LSTP, LOP or other user-added device types may be mapped to the devices in the two parts of the array\(^2\). We divide the wires in the array also into two parts, wires inside mats and wires outside mats. Different wire types such as the semi-global or global wire types or other user-defined wire types may be mapped to the wires inside and outside mats.

\(^2\)It is important to note that in reality, SRAM cell functionality and design does depend on device type [23, 26, 28], however, we do not model different SRAM cell designs for the different device types.
In this chapter, we describe the modeling of embedded DRAM in CACTI-D. CACTI-D supports both logic process based DRAM (LP-DRAM) and commodity DRAM (COMM-DRAM) technologies. An LP-DRAM process [47, 54, 64] typically means that DRAM has been embedded into the logic process without affecting the characteristics of the original process much [53]. In our modeling of DRAM, we leverage the similarity that exists in the global and peripheral circuitry of SRAM and DRAM and model only their essential differences. We also use the same array organization for embedded DRAM that we have used for embedded SRAM. By having a common framework that in general places embedded SRAM and DRAM on an equal footing and emphasizes only their essential differences, we would be able to compare relative tradeoffs involving embedded SRAM and DRAM.

7.1 Differences between SRAM and DRAM

We capture the following essential differences between SRAM and DRAM in our area, delay and power models:

7.1.1 Cell

The most essential difference between SRAM and DRAM is in their storage cell. While SRAM typically makes use of the 6T cell that was shown in Figure 4.7 and the
principle of positive feedback to store data, DRAM typically makes use of the 1T-1C cell shown in Figure 7.1 and relies on the charge-storing capability of the capacitor. Because it makes use of only one transistor, a DRAM cell is usually laid out in a much smaller area compared to an SRAM cell. Logic process based DRAM (LP-DRAM) cells presented in [92] for four different technology nodes – 180/130/90/65 nm – have areas in the range of 19–26$F^2$. In contrast, a typical SRAM cell would have an area of about 120–150$F^2$. Commodity DRAM (COMM-DRAM) process based cells occupy the least area with typical values in the range of 6–8$F^2$.

7.1.2 Destructive Readout and Writeback

When data is read out from a DRAM cell, the charge stored in the cell gets destroyed because of charge redistribution between the cell and its capacitive bitline, and there is a need for data to be written back into the cell. Also, after a writeback the bitlines need to be restored to their precharged value. These writeback and restore operations take time and increase the random cycle time of a DRAM array. In an SRAM there is no need for writeback because the data is not destroyed during a read.

7.1.3 Sense Amplifier Input Signal

In a DRAM, the maximum differential signal developed on the bitlines is limited by the amount of charge transferred between the DRAM cell and the bitline which in turn
depends on the capacitance of the DRAM cell and the bitline. The lower the differential signal, the greater the sense amplifier delay. In an SRAM, there is no charge-based limit on the differential signal developed on the bitlines. In any case, in modern technologies the sense amplifiers of SRAMs or DRAMs are operating at signal level inputs of more or less the same amplitude [53], so the delay of the sense amplifier in either SRAM or DRAM can come out to have similar values.

7.1.4 Refresh

In a DRAM cell, charge cannot be stored for an infinite time in the capacitor and the charge leaks out because of various leakage components. If charge from a DRAM cell is allowed to leak out for a sufficient period of time, the differential voltage developed on the bitline pair becomes so small that the data stored in the cell can no longer be detected by the sense amplifier. Thus there is an upper bound on the time for which data may be retained in a DRAM cell without it being refreshed, and this time is known as the retention time. Because of a finite retention time, the DRAM cell needs to be refreshed periodically. The LP-DRAM technology that we have based our LP-DRAM modeling on has much lower refresh period (64 µS for macro in [29]) than COMM-DRAMs (64ms), which means that the LP-DRAM cells have to be refreshed more often than the COMM-DRAM cells. This is because the LP-DRAM cells have employed transistors with oxides that are much thinner than those of the COMM-DRAM cells.

7.1.5 Wordline Boosting

In a DRAM cell, because the access takes place through an NMOS pass transistor, there is a $V_{\text{th}}$ drop during the write/writeback of a 1 into the cell. In order to prevent this $V_{\text{th}}$ drop, DRAM wordlines are usually boosted to a voltage at least equal to $V_{\text{PP}} = V_{\text{DD}} + V_{\text{th}}$. In commodity DRAMs, $V_{\text{th}}$ is relatively high (typically around 1V) in order to maintain the high refresh period (64ms) that requires extremely low leakage. This means
that $V_{PP}$ is also high (typically around 3.3V for 65nm) and forces the use of high voltage (thicker gate-oxide) slower transistors in the wordline driver. For the LP-DRAM DRAMs that we have modeled, however, $V_{th}$ is not very high (0.44V for 65nm), consequently $V_{PP}$ is also not very high (1.6V for 65nm).

7.2 Array Organization and Layout

For DRAM, we assume a folded array architecture [44] in the subarray, shown in Figure 7.2. In the folded array architecture, the bitline being read (true bitline) and its complement are laid out next to each other, similar to the dual bitlines of an SRAM cell. The difference here is that the true and complement bitlines connect to alternate rows of the array and not to the same row as in SRAM. This has an impact on bitline capacitance calculation. Assuming drain contacts are shared, the bitline capacitance for DRAM may be given by the following equation:

$$C_{bitline} = \frac{N_{subarr-rows}}{2}C_{\text{drain-cap-acc-transistor}} + N_{\text{subarr-rows}}C_{\text{bit-metal}} \quad (7.1)$$
7.2.1 Bitline Multiplexing

In DRAM, the read access is destructive. This means that during a read access after data is read from a DRAM cell, it needs to be written back into the cell. This writeback is typically accomplished by using the sense amplifier which detects the data stored in the cell during a read. During a read access, because each cell connected to a wordline is read out through its associated bitline, this means that there needs to be a sense-amplifier associated with each bitline. Hence bitline multiplexing, which is common in SRAMs to connect multiple bitlines to a single sense amplifier, is not feasible in DRAMs. Thus in DRAMs, there needs to be a sense amplifier associated with every bitline that can carry out the writeback. With respect to the bitline peripheral circuitry shown in Figure 3.4 this means that DRAM arrays do not have a bitline mux between the bitlines and sense amplifiers.

7.2.2 Reference Cells for \( V_{DD} \) Precharge

We assume that the bitlines are precharged to \( V_{DD} \) (GND) just like the DRAM described in [15, 64]. As in [15], we assume the use of reference cells that store \( V_{DD}/2 \) and connect to the complement bitline during a read. Figure 7.3 shows the bitline peripheral circuitry with the reference cells. For each subarray, we assume an extra two rows of reference cells that store \( V_{DD}/2 \). One of the rows with reference cells is activated during read of even-numbered rows in the subarray and the other row is activated during read of odd-numbered rows in the subarray.

7.3 Timing Model

7.3.1 Bitline Model

In DRAM the differential voltage swing developed on a bitline pair that acts as input to the sense amplifier is limited by the ratio of charge transferred between the bitline and
Figure 7.3. DRAM bitline circuitry showing reference cells for $V_{DD}$ precharge.

A DRAM cell, and is given by the following equation:

$$V_{\text{sense-max}} = \frac{V_{DD}}{2} \frac{C_{\text{dram}}}{C_{\text{dram}} + C_{\text{bitline}}}$$  \hspace{1cm} (7.2)

The delay for the above differential signal to develop may be given by the following equation [16] (ignoring the effect of wordline rise time):

$$T_{\text{step}} = 2.3R_{\text{dev}} \frac{C_{\text{dram}}C_{\text{bitline}}}{C_{\text{dram}} + C_{\text{bitline}}}$$  \hspace{1cm} (7.3)

where $R_{\text{dev}}$ is the resistance in series with the storage capacitor of the DRAM cell and may be given by the following equation:

$$R_{\text{dev}} = \frac{V_{DD}}{I_{\text{cell-on}}}$$  \hspace{1cm} (7.4)
It is important to note that use of Equations 7.3 and 7.4 assumes that the impact of bitline resistance on signal development time is negligible. This approximation works well for contemporary logic-based embedded DRAM processes. When bitline resistance becomes significant, as in the case of commodity DRAM processes that do not make use of copper bitlines, more sophisticated models need to be used.

Equation 7.3 assumes that 90% of the data stored in the cell is read out and corresponds to the development of approximately $V_{\text{sense-max}}$ (given by Equation 7.2) on the bitline pair. In order to improve the random cycle time of a DRAM macro further, nowadays less than 90% of the data stored in a cell is read out [40], just enough to generate the required input signal of the sense amplifier ($V_{\text{senseamp-input}}$). To accommodate this case, Equation 7.3 may be generalized as follows:

$$T_{\text{step-generalized}} = 2.3R_{\text{dev}}\frac{C_{\text{dram}}C_{\text{bitline}}}{C_{\text{dram}} + C_{\text{bitline}}} \frac{V_{\text{senseamp-input}}}{V_{\text{sense-max}}}$$  \hspace{1cm} (7.5)

When $V_{\text{senseamp-input}}$ is equal to $V_{\text{sense-max}}$, Equation 7.5 reduces to Equation 7.3. In CACTI-D, we assume a certain value for $V_{\text{senseamp-input}}$ (such as 80mV) and use Equation 7.5 to compute the signal development delay.

When rise time of the wordline is also considered, the bitline delay ($T_{\text{bitline}}$) of DRAM may be calculated using the same methodology that was used for SRAM (Equation 5.41 in Section 5.2).

The time taken to write data back into a DRAM cell after a read depends on the time taken for the charge transfer to take place between the bitline and the DRAM and may be given by the following equation:

$$T_{\text{writeback}} = T_{\text{step}}$$  \hspace{1cm} (7.6)
7.3.2 Multisubbank Interleave Cycle Time

For a DRAM array, we consider three timing characteristics: random access time, random cycle time and multibank interleave cycle time.

Calculation of random access time makes use of the same equations that were used for calculation of random access time of an SRAM array (in Section 5.2).

For a DRAM array, typically there are two kinds of cycle time: random cycle time and multibank interleave cycle time. Random cycle time has the same meaning as the random cycle time of an SRAM viz. it is the time interval between two successive random accesses. This time interval is typically limited by the time it takes to activate a wordline, sense the data, write back the data and then precharge the bitlines. Random cycle time can thus be calculated using the following equation:

\[ T_{\text{random-cycle}} = T_{\text{row-dec-driver}} + T_{\text{bitline}} + T_{\text{senseamp}} + T_{\text{writeback}} + T_{\text{wordline-reset}} + \max(T_{\text{bitline-precharge}}, T_{\text{bit-mux-out-precharge}}, T_{\text{senseamp-mux-out-precharge}}) \]  

(7.7)

In order to improve the rate at which a DRAM array is accessed so that it is not limited by the random cycle time of the array, DRAM arrays usually employ the concept of multibank interleaving. Multibank interleaving takes advantage of the fact that while random access to a particular bank is limited by the random cycle time, accesses to other banks need not be. With multibank interleaving, accesses to multiple DRAM banks that are on the same address/data bus are interleaved at a rate defined by the multibank interleave cycle time. In our terminology, each bank in an array has its own address and data bus and may be concurrently accessed. For our array organization, the concept of multibank interleaved mode is relevant to subbank access and not bank access, so in the rest of this discussion we use the terminology of multisubbank interleave mode and multisubbank interleave cycle. Thus, the multisubbank interleave cycle time is the
rate at which accesses may be interleaved between different subbanks of a bank. The multisubbank interleave cycle time depends on the degree of pipelining employed in the request and reply networks of a subbank, and is limited by the pipelining overhead. We assume minimum pipeline overhead and use the following simple equation to calculate multisubbank-interleave cycle time:

\[ T_{\text{multisubbank-interleave}} = \max(T_{\text{request-network}} + T_{\text{row-predec}}, T_{\text{reply-network}}) \] (7.8)

7.3.3 Retention Time and Refresh Period

An equation for the retention time of a DRAM array may be written as follows [43]:

\[ T_{\text{retention}} = \frac{C_{\text{dram-cell}} \Delta V_{\text{cell-worst}}}{I_{\text{worst-leak}}} \] (7.9)

where \( \Delta V_{\text{cell-worst}} \) is the worst-case change in the voltage stored in a DRAM cell which leads to a read failure, and \( I_{\text{cell-worst-leak}} \) is the worst-case leakage in a DRAM cell.

We assume that \( \Delta V_{\text{cell-worst}} \) is limited by \( V_{\text{min-sense}} \), the minimum input signal that may be detected by the bitline sense amplifier. Thus, for a given array organization, \( \Delta V_{\text{cell-worst}} \) may be calculated by solving the following equation for \( \Delta V_{\text{cell-worst}} \):

\[ V_{\text{min-sense}} = \frac{C_{\text{dram-cell}}}{C_{\text{dram-cell}} + C_{\text{bitline}}} \left( \frac{V_{\text{DD}}}{2} - \Delta V_{\text{cell-worst}} \right) \] (7.10)

If we assume that the differential voltage detected by the sense amplifier is independent of array organization, then this means that different array partitions would have different retention times depending on the charge transfer ratio between the DRAM cell and the bitlines. For each array organization, it’s thus possible to calculate the value for \( \Delta V_{\text{cell-worst}} \) using Equation 7.10, which may then be plugged into Equation 7.9 to find the retention time for that array organization.
The upper bound on the refresh period of a DRAM cell would be equal to its retention
time. We assume that a safety margin of 10% with respect to the retention time is built
into the refresh period and thus calculate the refresh period using the following equation:

\[ T_{\text{refresh}} = 0.9 T_{\text{retention}} \] (7.11)

### 7.4 Energy Model

During the read of a 0 from a DRAM cell, the true bitline is pulled down to GND
during the writeback. Energy is then consumed in restoring the bitline to \( V_{\text{DD}} \) during the
precharge operation. During the read of a 1 from a DRAM cell, because of our assumption
of \( V_{\text{DD}} \)-precharge, the voltage of the true bitline does not change but the voltage of the
complementary bitline gets pulled down to GND and needs to be restored to \( V_{\text{DD}} \). So
for DRAM, the power consumed in a bitline during a read may be approximated by the
following equation:

\[ E_{\text{dyn-read-bitline}} = C_{\text{bitline}} V_{\text{DD}}^2 \] (7.12)

#### 7.4.1 Refresh Power

Refreshing the data in each cell of the array consumes power. In order to carry out
refresh, every cell in the array needs to be accessed, its data read out, and then written
back.

\[ P_{\text{refresh}} = \frac{E_{\text{refresh}}}{T_{\text{refresh}}} \] (7.13)

\[ E_{\text{refresh}} = E_{\text{refresh-predec-blks}} + E_{\text{refresh-row-dec-drivers}} + E_{\text{refresh-bitlines}} \] (7.14)
\[ E_{\text{refresh-predec-blks}} = N_{\text{banks}}N_{\text{subbanks}}N_{\text{mats-in-subbank}}E_{\text{dyn-mat-predec-blks}} \]  (7.15)

\[ E_{\text{refresh-row-dec-drivers}} = 4N_{\text{banks}}N_{\text{subbanks}}N_{\text{mats-in-subbank}}E_{\text{dyn-mat-row-dec-drivers}} \]  (7.16)

\[ E_{\text{refresh-bitlines}} = (4N_{\text{banks}}N_{\text{subbanks}}N_{\text{mats-in-subbank}}N_{\text{subarr-cols}}) E_{\text{dyn-read-bitline}} \]  (7.17)

### 7.5 Area Model

#### 7.5.1 Area of Reference Cells

As mentioned earlier in Section 7.2.2, the use of \( V_{\text{DD}} \)-precharge leads to the use of reference cells in the array [15]. For our array organization, this means that there are two additional wordlines per subarray.

#### 7.5.2 Area of Refresh Circuitry

In order to enable continued scaling of a logic-based embedded DRAM cell in terms of performance and cell area, [92] describes a new scalable embedded DRAM cell that makes use of an access transistor with an intermediate gate-oxide of moderate thickness (2.2nm for 90/65nm). This transistor is a standard offering in the logic process which incorporates the embedded DRAM. Conventional cells [39] in earlier technologies made use of access transistors with much thicker gate-oxides. An effect of the scalable embedded DRAM cell described in [92] is that it results in the cell having a lower retention time and a lower refresh period (because of higher worst-case leakage - 10s of pAs compared to 1fA for commodity DRAM). The macro discussed in [29] that makes use of the cell described in [92] has a refresh period of 64\( \mu \)s compared to conventional macros which have refresh period of 64ms. This low refresh period required innovation at the circuit level through the development of a concurrent refresh scheme described in [29] in order to guarantee high availability of the DRAM macro. This concurrent refresh scheme adds an extra bank select port to each bank (subbank in our terminology) thereby allowing for
concurrent memory access and bank refresh operations in different banks. Each bank is equipped with a row address counter that contains the address of the row to be refreshed. A concurrent refresh scheduler composed of an up-shift-register and a down-shift-register is required in order to generate the bank select signals.

Because we loosely base the parameters of our logic-based embedded DRAM technology on information presented in [15, 29, 92], we model the overhead of the concurrent refresh scheme on area. For our organization in which each subbank has multiple mats, we assume that each mat incurs overhead of a row address counter placed at the center of the mat. Because of the requirements of the design of the concurrent refresh scheme, for our organization, we assume \( N_{\text{subbanks-in-mat}} \) number of concurrent refresh schedulers per bank.

7.6 Technology Modeling

7.6.1 Cell Characteristics

Similar to the SRAM technology assumptions, we assume two types of transistors in the DRAM array. One transistor type is used in the DRAM cell and wordline driver, while the other is used in the rest of the peripheral and global circuitry. Table 7.1 shows technology parameters of the DRAM cell that we consider in our modeling.

For COMM-DRAM technology, we obtain technology data by using projections from the ITRS and other sources [9, 32, 58, 59]; Table 7.2 shows this data. For LP-DRAM technology, we obtain technology data for the four technology nodes by using an approach that makes use of published data, scaling assumptions, and transistor characterization using MASTAR. For 90nm and 65nm, we use technology data from [29, 92]; Table 7.3 shows this data. In order to obtain technology data for the 45nm and 32nm technology nodes, we make the following scaling assumptions:

1. Storage capacitance of the LP-DRAM cell is assumed to remain fixed at the 90/65nm value of 20fF;
TABLE 7.1

TECHNOLOGY PARAMETERS OF THE DRAM CELL

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Meaning</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_{\text{dram}}$</td>
<td>Storage capacitance of a DRAM cell</td>
<td>F</td>
</tr>
<tr>
<td>$A_{\text{dram-cell}}$</td>
<td>Area occupied by the DRAM cell</td>
<td>$mm^2$</td>
</tr>
<tr>
<td>$AR_{\text{dram-cell}}$</td>
<td>Aspect ratio of the DRAM cell</td>
<td></td>
</tr>
<tr>
<td>$V_{\text{DD dram-cell}}$</td>
<td>Voltage representing a 1 in a DRAM cell</td>
<td>V</td>
</tr>
<tr>
<td>$V_{\text{th-dram-acc-transistor}}$</td>
<td>Threshold voltage of DRAM cell access transistor</td>
<td>mV</td>
</tr>
<tr>
<td>$L_{\text{dram-acc-transistor}}$</td>
<td>Length of DRAM cell access/wordline transistor</td>
<td>nm</td>
</tr>
<tr>
<td>$W_{\text{dram-acc-transistor}}$</td>
<td>Width of DRAM cell access transistor</td>
<td>nm</td>
</tr>
<tr>
<td>$EOT_{\text{dram-acc-transistor}}$</td>
<td>Equivalent oxide thickness of DRAM access transistors</td>
<td>nm</td>
</tr>
<tr>
<td>$I_{\text{on-dram-cell}}$</td>
<td>DRAM cell on-current under nominal conditions</td>
<td>$\mu A$</td>
</tr>
<tr>
<td>$I_{\text{off-dram-cell}}$</td>
<td>DRAM cell off-current under nominal conditions</td>
<td>pA</td>
</tr>
<tr>
<td>$I_{\text{worst-off-dram-cell}}$</td>
<td>DRAM cell off-current under worst-case conditions</td>
<td>A/µm</td>
</tr>
<tr>
<td>$V_{\text{PP}}$</td>
<td>Boosted wordline voltage applied to gate of access transistor</td>
<td>V</td>
</tr>
<tr>
<td>$I_{\text{on-dram-wordline-transistor}}$</td>
<td>On-current of wordline transistor</td>
<td>$\mu A/\mu$</td>
</tr>
</tbody>
</table>

2. The nominal off-current is assumed to remain fixed at 2pA for the cell;

3. Gate oxide thickness is scaled slowly in order to keep gate leakage low and sub-threshold current as the dominant leakage current. It has a value of 2.1nm for 45nm and 2nm for 32nm;

4. $V_{\text{DD dram-cell}}$ is scaled such that the electric field in the dielectric of the DRAM ($V_{\text{PP}}/EOT_{\text{dram-acc-transistor}}$) access transistor remains almost constant;

5. There is excellent correlation in the 180–130nm (for conventional thick-oxide device) and 90–65nm (for the intermediate-oxide device) scaling-factors for width and length of the DRAM cell access transistor. We assume that there would be good correlation in the 130–90nm and 65–45nm scaling-factors as well. For 32nm, we assume that the width and length are scaled in the same proportion as feature size;

6. We calculate area of the DRAM cell using the equation

   $$A_{\text{dram-cell}} = 10W_{\text{dram-acc-transistor}}L_{\text{dram-acc-transistor}}.$$  

   This equation has good correlation with the actual cell area of the 90 and 65 nm cells that made use of the intermediate-oxide based devices; and

7. We simply assume that nominal on-current of the cell can be maintained at the
65nm value. This would require aggressive scaling of the series parasitic resistance of the transistor.

With these scaling assumptions, we use MASTAR to model the transistors. It is assumed that the resulting channel doping concentrations calculated by MASTAR would be feasible. Table 7.4 summarizes the characteristics of the LP-DRAM cell for the four technology nodes.

### TABLE 7.2

**COMM-DRAM CELL TECHNOLOGY DATA FOR THE 90/65/45/32NM TECHNOLOGY NODES**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>90nm</th>
<th>65nm</th>
<th>45nm</th>
<th>32nm</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_{\text{dram}}$ (F)</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>$A_{\text{dram-cell}}$ (F - Feature size)</td>
<td>$6F^2$</td>
<td>$6F^2$</td>
<td>$6F^2$</td>
<td>$6F^2$</td>
</tr>
<tr>
<td>$V_{DD_{\text{dram-cell}}}$ (V)</td>
<td>1.6</td>
<td>1.3</td>
<td>1.1</td>
<td>1.0</td>
</tr>
<tr>
<td>$V_{\text{th-dram-acc-transistor}}$ (V)</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>$L_{\text{dram-acc-transistor}}$ (nm)</td>
<td>90</td>
<td>65</td>
<td>45</td>
<td>32</td>
</tr>
<tr>
<td>$W_{\text{dram-acc-transistor}}$ (nm)</td>
<td>90</td>
<td>65</td>
<td>45</td>
<td>32</td>
</tr>
<tr>
<td>$I_{\text{on-dram-cell}}$ ($\mu$A)</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>$I_{\text{off-dram-cell}}$ (fA)</td>
<td>0.01</td>
<td>0.01</td>
<td>0.01</td>
<td>0.01</td>
</tr>
<tr>
<td>$I_{\text{worst-off-dram-cell}}$ (fA)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$V_{PP}$ (V)</td>
<td>3.7</td>
<td>3.3</td>
<td>2.7</td>
<td>2.6</td>
</tr>
</tbody>
</table>
TABLE 7.3

LP-DRAM CELL TECHNOLOGY DATA FOR 90NM AND 65NM FROM [29, 92]

<table>
<thead>
<tr>
<th>Parameter</th>
<th>90nm</th>
<th>65nm</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_{\text{dram}}$ (F)</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>$A_{\text{dram-cell}}$ (F - feature size)</td>
<td>$20.7F^2$</td>
<td>$25.6F^2$</td>
</tr>
<tr>
<td>$V_{\text{DDdram-cell}}$</td>
<td>1.2</td>
<td>1.2</td>
</tr>
<tr>
<td>$V_{\text{th-dram-acc-transistor}}$</td>
<td>350</td>
<td>350</td>
</tr>
<tr>
<td>$L_{\text{dram-acc-transistor}}$ (nm)</td>
<td>120</td>
<td>120</td>
</tr>
<tr>
<td>$W_{\text{dram-acc-transistor}}$</td>
<td>140</td>
<td>90</td>
</tr>
<tr>
<td>$I_{\text{on-dram-cell}}$ ($\mu$A)</td>
<td>45</td>
<td>36</td>
</tr>
<tr>
<td>$I_{\text{off-dram-cell}}$ (pA)</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>$V_{\text{PP}}$ (V)</td>
<td>1.5</td>
<td>1.5</td>
</tr>
</tbody>
</table>

TABLE 7.4

LP-DRAM CELL TECHNOLOGY DATA FOR THE 90/65/45/32NM TECHNOLOGY NODES

<table>
<thead>
<tr>
<th>Parameter</th>
<th>90nm</th>
<th>65nm</th>
<th>45nm</th>
<th>32nm</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_{\text{dram}}$ (F)</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>$A_{\text{dram-cell}}$ (F - feature size)</td>
<td>$20.7F^2$</td>
<td>$25.6F^2$</td>
<td>$30.4F^2$</td>
<td>$30.6F^2$</td>
</tr>
<tr>
<td>$V_{\text{DDdram-cell}}$</td>
<td>1.2</td>
<td>1.2</td>
<td>1.1</td>
<td>1.1</td>
</tr>
<tr>
<td>$V_{\text{th-dram-acc-transistor}}$ (mV)</td>
<td>455</td>
<td>438</td>
<td>446</td>
<td>445</td>
</tr>
<tr>
<td>$L_{\text{dram-acc-transistor}}$ (nm)</td>
<td>120</td>
<td>120</td>
<td>78</td>
<td>56</td>
</tr>
<tr>
<td>$W_{\text{dram-acc-transistor}}$ (nm)</td>
<td>140</td>
<td>90</td>
<td>79</td>
<td>56</td>
</tr>
<tr>
<td>$I_{\text{on-dram-cell}}$ ($\mu$A)</td>
<td>45</td>
<td>36</td>
<td>36</td>
<td>36</td>
</tr>
<tr>
<td>$I_{\text{off-dram-cell}}$ (pA)</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>$I_{\text{worst-off-dram-cell}}$ (pA)</td>
<td>21.1</td>
<td>19.6</td>
<td>19.5</td>
<td>18.9</td>
</tr>
<tr>
<td>$V_{\text{PP}}$ (V)</td>
<td>1.6</td>
<td>1.6</td>
<td>1.5</td>
<td>1.5</td>
</tr>
</tbody>
</table>
CHAPTER 8

CACHE AND MAIN MEMORY CHIP MODELING

In this chapter we describe the modeling of caches and main memory chips in CACTI-D. We describe the organizational characteristics of caches and main memory chips and describe how the area, delay, and energy equations presented in Chapter 5 are modified.

8.1 Cache Modeling

8.1.1 Organization

As described in [95], a cache has a tag array in addition to a data array. In earlier versions of CACTI the data and tag arrays were modeled separately with separate code functions even though the data and tag arrays are structurally very similar. The essential difference between the tag array and the data array is that the tag array includes comparators that compare the input tag bits with the stored tags and produce the tag match output bits. Apart from the comparators, the rest of the peripheral/global circuitry and memory cells are identical for data and tag arrays. In CACTI-D, we leverage this similarity between the data and tag arrays and use the same set of functions for their modeling. For the tag array, we add the comparator area, delay and power models.

Figure 8.1 illustrates the organization of a set-associative tag array. Each mat includes comparators at the outputs of the sense amplifiers. These comparators compare the stored tag bits with the input tag bits and produce the tag match output bits. These tag match output signals are the way-select signals that serve as inputs to the data array. The way-
Figure 8.1. Organization of a set-associative tag array.

select signals traverse over the vertical and horizontal H-trees of the tag array to get to the edge of the tag array from where they are shipped to the data array. In a cache of normal access type, as shown in Figure 8.2, these way-select signals then enter the data array where, like the address and datain signals, they travel along the horizontal and vertical H-tree networks to get to mats in the accessed subbank. At the mats, these way-select signals are ‘anded’ with sense amplifier mux decode signals (if any) and the resultant signals serve as select signals for the sense amplifier mux which generates part of the output word from the mat. In a cache of fast access type [82], as shown in Figure 8.3, the output data corresponding to all ways are transmitted to the edge of the data array where the appropriate way is then selected based on the way-select signals. In a cache of sequential access type [82], the tag array is accessed first and only if there is a tag match is the data array accessed.
Figure 8.2. Organization of a data array in a set-associative cache of normal access type.

Figure 8.3. Organization of a data array in a set-associative cache of fast access type.
8.1.2 Delay Model

We present equations for access and cycle times of a cache. The access time of a cache depends on the type of cache access – whether it is normal, sequential or fast.

The equation for access time of a normal cache which is set-associative is as follows:

\[
T_{access-normal-set-associative} = \max(T_{tag-arr-access} + T_{data-arr-request-network} + T_{data-arr-senseamp-mux-decode}, T_{data-arr-mat}) + T_{data-arr-reply-network} \tag{8.1}
\]

\[
T_{tag-arr-access} = T_{tag-arr-request-network} + T_{tag-arr-mat} + T_{tag-arr-reply-network} \tag{8.2}
\]

In the above equation, \(T_{tag-arr-access}\), the access time of the tag array, is calculated using the following equation.

\[
T_{tag-arr-access} = T_{tag-arr-request-network} + T_{tag-arr-mat} + T_{tag-arr-reply-network} + T_{comparators} \tag{8.3}
\]

The equation for access time of a normal cache which is direct-mapped is as follows:

\[
T_{access-normal-direct-mapped} = \max(T_{tag-arr-access}, T_{data-arr-access}) \tag{8.4}
\]

The equation for access time of a sequentially accessed (tag array is accessed first before data array access begins) cache is as follows:

\[
T_{access-sequential} = T_{tag-arr-access} + T_{data-arr-access} \tag{8.5}
\]
The equation for access time of a fast cache is as follows:

$$T_{\text{access-fast}} = \max(T_{\text{tag-arr-access}}, T_{\text{data-arr-access}}) + T_{\text{way-select-mux}}$$  \quad (8.6)

8.1.3 Area Model

Total area of the cache is calculated by simply adding the areas occupied by the tag and data arrays.

$$A_{\text{cache}} = A_{\text{data-array}} + A_{\text{tag-array}}$$  \quad (8.7)

$A_{\text{tag-array}}$ is calculated using the equations presented in Section 5.1 of Chapter 5 with the area of the comparators also added.

8.1.4 Energy Model

The dynamic energy consumed in the cache and its standby leakage power are calculated by simply adding their values for the data and tag arrays. For the tag array, the leakage in the comparators is also considered.

$$E_{\text{dyn-energy-cache}} = E_{\text{dyn-energy-data-array}} + E_{\text{dyn-energy-tag-array}}$$  \quad (8.8)

$$P_{\text{leak-cache}} = P_{\text{leak-data-array}} + P_{\text{leak-tag-array}}$$  \quad (8.9)

8.2 Main Memory Chip Modeling

8.2.1 Organization

A main memory chip is different from an embedded memory in terms of its organization and operation. Main memory is usually organized as DIMMs (Dual In-line Memory Modules) which are typically composed of 8 or 16 main memory chips. Each main memory chip on a DIMM has 4 or 8 data output pins which are much lower than what is
typical of embedded memories. Because of their limited number of data pins, main memory chips operate in burst mode with a *burst length* of 4 or 8. Figure 8.4 illustrates a 64-bit main memory channel that is connected to two 2 DIMMs with each DIMM composed of 8 DRAM chips each producing 8 output bits. Modern main memory chips incorporate the concept of *internal prefetch width* which determines the number of bits that are prefetched internally inside the DRAM core. For DDR, the internal prefetch width is 2, for DDR2 it’s 4, for DDR3 it’s 8, and so on. Main memory chips are also characterized by *page size* which is equal to the number of sense amplifiers that are activated per access. Figure 8.5 shows the block diagram of a Micron 256Mb DDR2 main memory DRAM chip with 4 banks and 8 data output pins. It can be seen that because this is a DDR2 chip with 8 output pins, internally 4 times 8 = 32 bits are prefetched. It can also be seen that the page size of this chip is 8Kb. Because of their limited number of address pins, a main memory chip operates differently from an embedded memory. In an embedded memory all the address bits may be decoded in parallel. In a main memory chip, row and column address are multiplexed on the address bus. The row address is first applied and is used to latch data from a row into the sense amplifiers. After the data gets latched, the column address is then applied and the read or write access is carried out. We incorporate the impact of these organizational and operational features of a main memory chip into the CACTI-D models.

Main memory COMM-DRAM chips are operated using ACTIVATE, READ, WRITE and PRECHARGE commands.\footnote{It is important to note that in reality the operation of a main memory chip is more complicated than our simplified description here.} ACTIVATE latches data into a DRAM page. After read or write operations to this page are carried out, the page can be closed: i.e. data in the sense amplifiers are written back into the DRAM cells and the bitlines are restored to their precharged state. Activation and precharge of a page consumes time and energy, so reducing the number of ACTIVATE and PRECHARGE commands associated with a set...
of memory accesses can potentially reduce delay and energy. With regards to this, main memory chips can be operated using an open page policy [71]. When a main memory chip is operated under open page policy, the page is left open after an access. Thus subsequent accesses to the same page do not consume additional delay and energy for activation of the page. Delay and energy of activation and precharge may potentially be amortized over multiple accesses to the same page. Open page policy, however, does not produce much delay and energy benefits if the probability of hit in an open page is low. In such cases, having the page open can cause energy loss due to leakage in the sense amplifiers. Under such conditions, a closed page policy may be a better choice. When a main memory chip is operated with a closed page policy, the page is closed unless it is known that the subsequent access is to the same page.
Figure 8.5. Block diagram of a Micron 256Mb DDR2 main memory DRAM chip with 4 banks and 8 data output pins (Courtesy Micron Technology).

An important operational principle employed in main memory DRAM chips is the concept of multibank interleaving. The random cycle time \( t_{RC} \) of main memory DRAM chips tends to be quite large (typically around 50ns). But consecutive accesses to a different bank need not be limited by the random cycle time, so main memory DRAM chips also include a specification of multibank interleave cycle time, \( t_{RRD} \). A typical value for \( t_{RRD} \) would be 7.5ns, so with multibank interleaving the throughput from a main memory DRAM chip can be significantly increased.

8.2.2 Delay Model

In our main memory chip timing model we add equations for computation of activation delay \( T_{rcd} \), CAS latency, precharge delay, and random cycle time.
\[ T_{\text{rcd}} = T_{\text{request-network}} + T_{\text{row-predec}} + T_{\text{row-dec-driver}} + T_{\text{bitline}} + T_{\text{senseamp}} \] (8.10)

\[ T_{\text{cas-latency}} = T_{\text{request-network}} + T_{\text{senseamp-mux-predec}} + T_{\text{senseamp-mux-dec-driver}} + T_{\text{reply-network}} \] (8.11)

\[ T_{\text{precharge}} = T_{\text{request-network}} + T_{\text{writeback}} + T_{\text{wordline-reset}} + T_{\text{bitline-restore}} \] (8.12)

\[ T_{\text{random-cycle}} = T_{\text{rcd}} + T_{\text{cas-latency}} + T_{\text{precharge}} \] (8.13)

### 8.2.3 Energy Model

In our main memory chip energy model, we add equations for computation of energy consumed during ACTIVATE, READ, WRITE and PRECHARGE operations.

\[ E_{\text{activate}} = E_{\text{dyn-activate-request-network}} + (E_{\text{dyn-row-predec-bkls}} + E_{\text{dyn-decoder-drivers}} + E_{\text{senseamps}})N_{\text{mats-in-subbank}} \] (8.14)

\[ E_{\text{read}} = (E_{\text{dyn-read-request-network}} + E_{\text{senseamp-predec-bkls}} + E_{\text{senseamp-mux-decoder-drivers}} + E_{\text{dyn-read-reply-network}})L_{\text{burst}} \] (8.15)

\[ E_{\text{precharge}} = 4N_{\text{subbars-cols}}N_{\text{mats-in-subbank}}E_{\text{precharge-bitline}} \] (8.16)

\[ E_{\text{precharge-bitline}} = C_{\text{bitline}}V_{\text{DD}}\frac{V_{\text{DD}}}{2} \] (8.17)

\( E_{\text{dyn-activate-request-network}}, E_{\text{dyn-read-request-network}}, \text{ and } E_{\text{dyn-reply-request-network}} \) are calculated as described in Section 5.3.1. The number of signals that traverse in the various segments of the request and reply H-trees is calculated in a way similar to that described in Section 5.3.1. For an ACTIVATE command, the number of signals that enter the array is equal to the number of address bits required to decode a row. For a READ command, the number of signals that enter the array is equal to sum of the number of address bits
required to decode a column and the number of dataout bits that leave the array.

Background Power

A significant source of power consumption in main memory chips is the background power. Background power that is consumed when the DRAM chip is in active or precharged mode is dependent on the standby current consumed by pad circuitry, input registers, input and output buffers, clock drivers, DLL (Delay Locked Loop), and other circuitry. In CACTI-D, we do not analytically model such circuitry. Instead we study the background power consumption of a large number of SDR, DDR, DDR2, and DDR3 DRAM chips to find any discernible trend. Table 8.1 shows the maximum precharge standby current consumption of DRAM chips of various DDR generations with different capacities and system clock periods. It can be seen that maximum standby precharge currents remain the 50–80mA range. In CACTI-D, we assume a default maximum standby precharge current of 70mA for main memory chips.
<table>
<thead>
<tr>
<th>Capacity (Mb)</th>
<th>$V_{DD}$ (V)</th>
<th>$t_{CK}$ (ns)</th>
<th>Max precharge standby current (mA)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDRAM</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>128</td>
<td>3.3</td>
<td>7.5</td>
<td>50</td>
</tr>
<tr>
<td>DDR DRAM</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>256</td>
<td>2.5</td>
<td>5</td>
<td>60</td>
</tr>
<tr>
<td>DDR2 DRAM</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>256</td>
<td>1.8</td>
<td>2.5</td>
<td>45</td>
</tr>
<tr>
<td>512</td>
<td>1.8</td>
<td>2.5</td>
<td>70</td>
</tr>
<tr>
<td>1024</td>
<td>1.8</td>
<td>2.5</td>
<td>80</td>
</tr>
<tr>
<td>2048</td>
<td>1.8</td>
<td>3</td>
<td>70</td>
</tr>
<tr>
<td>DDR3 DRAM</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1024</td>
<td>1.5</td>
<td>1.5</td>
<td>75</td>
</tr>
<tr>
<td>2048</td>
<td>1.5</td>
<td>1.5</td>
<td>70</td>
</tr>
<tr>
<td>4096</td>
<td>1.5</td>
<td>1.5</td>
<td>75</td>
</tr>
</tbody>
</table>
In this chapter, we compare projections of area, delay and power produced by CACTI-D against published data of real cache designs. The area, delay and power of a real cache or RAM design are influenced by various factors. The design process inherently makes certain area, delay and power tradeoffs based on budgets and requirements. Area, delay and power are also influenced by design methodology, human bias and other practical considerations such as availability of IP from past designs etc. CACTI-D is based on generic assumptions of cache organization, circuits, design methodology, layout, design rules and technology, whereas a real cache design is based on specific choices of all these. With CACTI-D, however, we provide a number of knobs that can be turned in order to try to emulate a real cache design in a better way. We use information from real cache specifications to fix as many of the input parameters required by CACTI-D as possible, such as capacity, associativity, line size, technology-node, etc. In order to understand and explore the area, delay, and power tradeoffs that are possible, we vary parameters such as max_area_constraint and max_acc_time_constraint within reasonable bounds.

9.1 SRAM Validation

9.1.1 Sun SPARC 90nm L2 cache

[55] describes the implementation of a 90nm SPARC 4MB L2 cache. Table 9.1 shows the area, access time, random cycle time and power of the SPARC L2 cache. The clock
frequency of the CPU core itself is 1.6 GHz but the L2 cache has a throughput of two clock cycles, so we fix the random cycle time of the L2 cache as 800 MHz.

Table 9.2 presents the input parameters used with CACTI-D to model this cache. From the description of the cache, we could not be sure whether the cache access mode is ‘normal’ or ‘fast’, so we try out both scenarios. In order to explore a range of area, delay, and power tradeoffs, we vary max_area_constraint between 0 and 70% and max_delay_constraint between 0 and 30%. In order to meet the aggressive random cycle time of the cache, we optimize the solution for random cycle time only. For ‘wire type outside mat’, we try out both ‘semi-global’ and ‘global’ wire types.

Figure 9.1 shows bubble charts showing access time, area, and power of the SPARC L2 cache and the various solutions generated by CACTI-D. The area of the bubble represents the area of the cache. The bubble corresponding to the actual SPARC L2 properties is referred to as the “target” bubble. The charts shown in Figure 9.1 are for CACTI-D solutions with ‘fast’ access mode and ‘semi-global’ wire type outside mat. We believe that these values for the parameters are likely to be closest to the actual cache design.

As we do not know the operating conditions corresponding to the published value for power of the SPARC L2, we compute dynamic power for the CACTI-D solutions for three activity factors – 0.1, 0.5 and 1.0. Also, we assume that the ratio of read to write

---

**TABLE 9.1**

CHARACTERISTICS OF SUN’S SPARC 90NM L2 CACHE

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Area (mm(^2))</td>
<td>128</td>
</tr>
<tr>
<td>Access time (ns)</td>
<td>5</td>
</tr>
<tr>
<td>Clock frequency (MHz)</td>
<td>800</td>
</tr>
<tr>
<td>Total power (W)</td>
<td>8</td>
</tr>
</tbody>
</table>

...
TABLE 9.2

CACTI-D INPUT PARAMETERS USED FOR MODELING 90NM SPARC L2 CACHE

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capacity (MB)</td>
<td>4</td>
</tr>
<tr>
<td>Line size (bytes)</td>
<td>32</td>
</tr>
<tr>
<td>Associativity</td>
<td>4</td>
</tr>
<tr>
<td>Number of read/write ports</td>
<td>1</td>
</tr>
<tr>
<td>Number of exclusive ports</td>
<td>0</td>
</tr>
<tr>
<td>Number of banks</td>
<td>1</td>
</tr>
<tr>
<td>Technology-node (nm)</td>
<td>90</td>
</tr>
<tr>
<td>Output width (bits)</td>
<td>256</td>
</tr>
<tr>
<td>Specific tag</td>
<td>Yes</td>
</tr>
<tr>
<td>Tag width</td>
<td>34</td>
</tr>
<tr>
<td>Access mode</td>
<td>Normal/Fast</td>
</tr>
<tr>
<td>RAM cell technology</td>
<td>SRAM</td>
</tr>
<tr>
<td>Repeaters in bank H-trees</td>
<td>Yes</td>
</tr>
<tr>
<td>max_area_constraint (%)</td>
<td>0 - 70</td>
</tr>
<tr>
<td>max_acc_time_constraint (%)</td>
<td>0 - 30</td>
</tr>
<tr>
<td>max_repeater_delay_constraint (%)</td>
<td>10</td>
</tr>
<tr>
<td>Optimize for dynamic energy</td>
<td>No</td>
</tr>
<tr>
<td>Optimize for dynamic power</td>
<td>No</td>
</tr>
<tr>
<td>Optimize for leakage power</td>
<td>No</td>
</tr>
<tr>
<td>Optimize for cycle time</td>
<td>Yes</td>
</tr>
<tr>
<td>Temperature (K)</td>
<td>360</td>
</tr>
<tr>
<td>mat_leak_reduction_due_to_sleep_transistors_factor</td>
<td>1</td>
</tr>
<tr>
<td>SRAM cell/wordline technology flavor</td>
<td>ITRS HP</td>
</tr>
<tr>
<td>Peripheral/Global circuitry technology flavor</td>
<td>ITRS HP</td>
</tr>
<tr>
<td>Interconnect projection type</td>
<td>Conservative</td>
</tr>
<tr>
<td>Wire type inside mat</td>
<td>Semi-global</td>
</tr>
<tr>
<td>Wire type outside mat</td>
<td>Semi-global/Global</td>
</tr>
</tbody>
</table>

accesses is 3. Note that only CACTI-D solutions that can meet the random cycle time of the SPARC L2 have been considered. It can be seen that many solutions have access time, area and power that are quite similar to that of the SPARC L2 cache. Table 9.3 shows error percentages of prominent CACTI-D solutions with respect to the SPARC L2.
An activity factor of 1 has been assumed for computation of dynamic power. It can be seen that the CACTI-D solution with best access time produces errors of -1%, -23%, and -39% in access time, area, and power with respect to the SPARC L2 cache. Considering the generic nature of CACTI-D modeling these errors are reasonable.

For the 90nm SPARC L2, Figure 9.2 shows bubble charts and Table 9.4 shows the error percentages for CACTI-D solutions with ‘fast’ access mode and ‘global’ wire type outside mat; Figure 9.3 and Table 9.5 show the same for CACTI-D solutions with ‘normal’ access mode and ‘semi-global’ wire type outside mat; Figure 9.4 and Table 9.6 show the same for CACTI-D solutions with ‘normal’ access mode and ‘global’ wire type outside mat. It can be seen that for all assumptions, the error percentages of the CACTI-D solutions are reasonable.
Figure 9.1. Access time, area and power of the 90nm SPARC L2 cache and of solutions generated by CACTI-D. The CACTI-D solutions assume ‘fast’ access mode and ‘semi-global’ wire type outside mat. The 3 plots correspond to 3 activity factors assumed while computing dynamic power for the CACTI-D-generated solutions.
TABLE 9.3

ERROR PERCENTAGES OF PROMINENT SOLUTIONS GENERATED BY CACTI-D WITH RESPECT TO A 90NM SPARC L2 CACHE. THE CACTI-D SOLUTIONS ASSUME ‘FAST’ ACCESS MODE AND ‘SEMI-GLOBAL’ WIRE TYPE OUTSIDE MAT. AN ACTIVITY FACTOR OF 1 HAS BEEN ASSUMED FOR CALCULATION OF DYNAMIC POWER

<table>
<thead>
<tr>
<th>Solution</th>
<th>% error in acc time</th>
<th>% error in area</th>
<th>% error in power</th>
<th>Avg of acc time, and power % errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best % error in acc time</td>
<td>1</td>
<td>-31</td>
<td>-35</td>
<td>22</td>
</tr>
<tr>
<td>Best % error in area</td>
<td>3</td>
<td>4</td>
<td>-26</td>
<td>11</td>
</tr>
<tr>
<td>Best % error in power</td>
<td>26</td>
<td>-11</td>
<td>-25</td>
<td>21</td>
</tr>
<tr>
<td>Best average of area, acc time, and power % errors</td>
<td>3</td>
<td>4</td>
<td>-26</td>
<td>11</td>
</tr>
<tr>
<td>Best average of area and acc time % errors</td>
<td>3</td>
<td>4</td>
<td>-26</td>
<td>11</td>
</tr>
<tr>
<td>Best average of acc time and power % errors</td>
<td>3</td>
<td>4</td>
<td>-26</td>
<td>11</td>
</tr>
<tr>
<td>Best acc time</td>
<td>-1</td>
<td>-23</td>
<td>-39</td>
<td>21</td>
</tr>
<tr>
<td>Best area</td>
<td>7</td>
<td>-33</td>
<td>-34</td>
<td>25</td>
</tr>
<tr>
<td>Best power</td>
<td>-1</td>
<td>-23</td>
<td>-39</td>
<td>21</td>
</tr>
</tbody>
</table>
Figure 9.2. Access time, area and power of the 90nm SPARC L2 cache and of solutions generated by CACTI-D. The CACTI-D solutions are for assumptions of ‘fast’ access mode and ‘global’ wire type outside mat. The 3 plots correspond to 3 activity factors assumed while computing dynamic power for the CACTI-D solutions.
### TABLE 9.4

ERROR PERCENTAGES OF PROMINENT SOLUTIONS GENERATED BY CACTI-D WITH RESPECT TO A 90NM SPARC L2 CACHE UNDER ASSUMPTIONS OF ‘FAST’ ACCESS MODE AND ‘GLOBAL’ WIRE TYPE OUTSIDE MAT. AN ACTIVITY FACTOR OF 1 HAS BEEN ASSUMED FOR CALCULATION OF DYNAMIC POWER

<table>
<thead>
<tr>
<th>Solution</th>
<th>% error in acc time</th>
<th>% error in area</th>
<th>% error in power</th>
<th>Avg of acc time, area and power % errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best % error in acc time</td>
<td>-2</td>
<td>26</td>
<td>-29</td>
<td>19</td>
</tr>
<tr>
<td>Best % error in area</td>
<td>-22</td>
<td>0</td>
<td>-42</td>
<td>21</td>
</tr>
<tr>
<td>Best % error in power</td>
<td>-2</td>
<td>26</td>
<td>-29</td>
<td>19</td>
</tr>
<tr>
<td>Best average of area, acc time, and power % errors</td>
<td>-2</td>
<td>26</td>
<td>-29</td>
<td>19</td>
</tr>
<tr>
<td>Best average of area and acc time % errors</td>
<td>-22</td>
<td>0</td>
<td>-42</td>
<td>21</td>
</tr>
<tr>
<td>Best average of acc time and power % errors</td>
<td>-2</td>
<td>26</td>
<td>-29</td>
<td>19</td>
</tr>
<tr>
<td>Best acc time</td>
<td>-2</td>
<td>26</td>
<td>-29</td>
<td>19</td>
</tr>
<tr>
<td>Best area</td>
<td>-16</td>
<td>-17</td>
<td>-38</td>
<td>24</td>
</tr>
<tr>
<td>Best power</td>
<td>-22</td>
<td>0</td>
<td>-42</td>
<td>21</td>
</tr>
</tbody>
</table>
Figure 9.3. Access time, area and power of the 90nm SPARC L2 cache and of solutions generated by CACTI-D. The CACTI-D solutions are for assumptions of ‘normal’ access mode and ‘semi-global’ wire type outside mat. The 3 plots correspond to 3 activity factors assumed while computing dynamic power for the CACTI-D solutions.
**TABLE 9.5**

ERROR PERCENTAGES OF PROMINENT SOLUTIONS GENERATED BY CACTI-D WITH RESPECT TO A 90NM SPARC L2 CACHE FOR A NORMAL CACHE UNDER ASSUMPTIONS OF ‘NORMAL’ ACCESS MODE AND ‘SEMI-GLOBAL’ WIRE TYPE OUTSIDE MAT. AN ACTIVITY FACTOR OF 1 HAS BEEN ASSUMED FOR CALCULATION OF DYNAMIC POWER

<table>
<thead>
<tr>
<th>Solution</th>
<th>% error in acc time</th>
<th>% error in area</th>
<th>% error in power</th>
<th>Avg of acc time, area and power % errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best % error in acc time</td>
<td>32</td>
<td>-40</td>
<td>-56</td>
<td>42</td>
</tr>
<tr>
<td>Best % error in area</td>
<td>37</td>
<td>-8</td>
<td>-51</td>
<td>32</td>
</tr>
<tr>
<td>Best % error in power</td>
<td>37</td>
<td>-8</td>
<td>-51</td>
<td>32</td>
</tr>
<tr>
<td>Best average of area, acc time and power % errors</td>
<td>37</td>
<td>-8</td>
<td>-51</td>
<td>32</td>
</tr>
<tr>
<td>Best average of area and acc time % errors</td>
<td>37</td>
<td>-8</td>
<td>-51</td>
<td>32</td>
</tr>
<tr>
<td>Best average of acc time and power % errors</td>
<td>37</td>
<td>-8</td>
<td>-51</td>
<td>32</td>
</tr>
<tr>
<td>Best acc time</td>
<td>32</td>
<td>-40</td>
<td>-56</td>
<td>42</td>
</tr>
<tr>
<td>Best area</td>
<td>32</td>
<td>-40</td>
<td>-56</td>
<td>42</td>
</tr>
<tr>
<td>Best power</td>
<td>33</td>
<td>-38</td>
<td>-61</td>
<td>44</td>
</tr>
</tbody>
</table>
Figure 9.4. Access time, area and power of the 90nm SPARC L2 cache and of solutions generated by CACTI-D. The CACTI-D solutions are for assumptions of ‘normal’ access mode and ‘global’ wire type outside mat. The 3 plots correspond to 3 activity factors assumed while computing dynamic power for the CACTI-D solutions.
TABLE 9.6

ERROR PERCENTAGES OF PROMINENT SOLUTIONS GENERATED BY CACTI-D WITH RESPECT TO A 90NM SPARC L2 CACHE FOR A NORMAL CACHE UNDER ASSUMPTIONS OF ‘NORMAL’ ACCESS MODE AND ‘GLOBAL’ WIRE TYPE OUTSIDE MAT. AN ACTIVITY FACTOR OF 1 HAS BEEN ASSUMED FOR CALCULATION OF DYNAMIC POWER

<table>
<thead>
<tr>
<th>Solution</th>
<th>% error in acc time</th>
<th>% error in area</th>
<th>% error in power</th>
<th>Avg of acc time, area and power % errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best % error in acc time</td>
<td>9</td>
<td>-28</td>
<td>-61</td>
<td>33</td>
</tr>
<tr>
<td>Best % error in area</td>
<td>10</td>
<td>-9</td>
<td>-59</td>
<td>26</td>
</tr>
<tr>
<td>Best % error in power</td>
<td>11</td>
<td>30</td>
<td>-53</td>
<td>31</td>
</tr>
<tr>
<td>Best average of area, acc time and power % errors</td>
<td>10</td>
<td>-9</td>
<td>-59</td>
<td>26</td>
</tr>
<tr>
<td>Best average of area and acc time % errors</td>
<td>10</td>
<td>-9</td>
<td>-59</td>
<td>26</td>
</tr>
<tr>
<td>Best average of acc time and power % errors</td>
<td>11</td>
<td>30</td>
<td>-53</td>
<td>31</td>
</tr>
<tr>
<td>Best acc time</td>
<td>9</td>
<td>-28</td>
<td>-61</td>
<td>33</td>
</tr>
<tr>
<td>Best area</td>
<td>18</td>
<td>-35</td>
<td>-58</td>
<td>37</td>
</tr>
<tr>
<td>Best power</td>
<td>9</td>
<td>-28</td>
<td>-61</td>
<td>33</td>
</tr>
</tbody>
</table>
9.1.2 Intel Xeon 65nm L3 cache

Table 9.7 shows the area, access time, dynamic power and leakage power of an Intel Xeon L3 cache in a 65nm process. [41] mentions that the access time of the cache is less than 9ns; we assume the access time to be 9 ns. The clock frequency of the core itself is given to be 3.4GHz and the clock frequency of the L3 is given to be half that of the core [27]. Also, because the output bus width of the L3 is 256 bits, it would require two cycles to transmit a 64-byte line, so we fix the random cycle frequency of the L3 to be one-fourth that of the CPU, i.e. 850MHz. The dynamic power of the cache comes out to be 5.4W based on information from [27, 89], however [21] mentions that the cache consumes about 1.7W for “average applications”. We speculate that these differences in dynamic power numbers that have been quoted are because of different activity factors in the cache caused due to measurements or simulations of applications with different characteristics. While carrying out comparisons of the area, delay and power of the Intel cache with those of the solutions generated by CACTI-D, we consider both values of power.

| TABLE 9.7 |

CHARACTERISTICS OF INTEL XEON’S 65NM L3 CACHE

<table>
<thead>
<tr>
<th>Area (mm$^2$)</th>
<th>200</th>
<th>Measured from die photo [74]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access time (ns)</td>
<td>9</td>
<td>[41]</td>
</tr>
<tr>
<td>Clock frequency (MHz)</td>
<td>850</td>
<td>[27]</td>
</tr>
<tr>
<td>Dynamic power (W)</td>
<td>1.7</td>
<td>5.4</td>
</tr>
<tr>
<td>Leakage power (W)</td>
<td>6.6</td>
<td>[27,89]</td>
</tr>
</tbody>
</table>

The 65nm process offers transistors with 35nm gate-lengths. The cache itself, however, makes use of longer-channel devices with lengths that are about 10% longer than the nominal. The longer-channel devices have on-currents that are about 10% less than
the nominal devices but have leakage that is lower by a factor of 3. The cache operates in a voltage domain different from that of the cores. The cores can operate at 1.25V while the cache operates at 1.1V.

In order to control leakage in the cache, the Intel cache implements n and p sleep transistors at the level of ‘blocks’ within subarrays (Each subarray within the Intel cache is composed of multiple ‘blocks’ with one block within a subarray activated per access. The ‘subarray’ of the Intel cache is not the same as that of CACTI-D). The impact of these sleep transistors is that leakage power in all blocks that are not activated during an access is cut down by half.

Table 9.8 shows the input parameters used with CACTI-D to model the Intel L3 cache. In order to compare the power numbers produced by CACTI-D with those of the Intel cache in a fair manner, we assume the use of leakage control mechanisms within CACTI-D similar to that used in the Intel cache. To model the longer-channel devices that have been used in the Intel cache which reduce leakage by a factor of 3, we also reduce the leakage of the CACTI-D 65 nm high-performance transistors by a factor of 3. Also, we set mat_leak_reduction_due_to_sleep_transistors_factor to 2 so that leakage of all mats that are not activated during an access is cut down by half. As in the SPARC L2, for ‘wire type outside mat’, we try out both ‘semi-global’ and ‘global’ wire types.

Figure 9.5 shows bubble charts of access time, area, and power of the Intel cache and the various solutions generated by CACTI-D. The area of the bubble again represents area of the cache. The charts shown in Figure 9.5 are for CACTI-D solutions with ‘conservative’ interconnect projections and ‘semi-global’ wire type outside mat as we believe that these values for the parameters are likely to be closest to the actual cache design. Again, as we do not know the operating conditions corresponding to the published value for power, we compute dynamic power for the CACTI-D solutions for three activity factors – 0.1, 0.5 and 1.0. Again, we assume that the ratio of read to write accesses is 3. There
TABLE 9.8

CACTI-D INPUT PARAMETERS USED FOR MODELING 65NM INTEL XEON L3 CACHE

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capacity (MB)</td>
<td>16</td>
</tr>
<tr>
<td>Line size (bytes)</td>
<td>64</td>
</tr>
<tr>
<td>Associativity</td>
<td>16</td>
</tr>
<tr>
<td>Number of read/write ports</td>
<td>1</td>
</tr>
<tr>
<td>Number of exclusive ports</td>
<td>0</td>
</tr>
<tr>
<td>Number of banks</td>
<td>2</td>
</tr>
<tr>
<td>Technology-node (nm)</td>
<td>65</td>
</tr>
<tr>
<td>Output width (bits)</td>
<td>512</td>
</tr>
<tr>
<td>Specific tag</td>
<td>No</td>
</tr>
<tr>
<td>Access mode</td>
<td>Serial</td>
</tr>
<tr>
<td>RAM cell technology</td>
<td>SRAM</td>
</tr>
<tr>
<td>max_area_constraint (%)</td>
<td>0 - 50</td>
</tr>
<tr>
<td>max_delay_constraint (%)</td>
<td>0 - 30</td>
</tr>
<tr>
<td>max_repeater_delay_constraint (%)</td>
<td>10</td>
</tr>
<tr>
<td>Optimize for dynamic energy</td>
<td>No</td>
</tr>
<tr>
<td>Optimize for dynamic power</td>
<td>No</td>
</tr>
<tr>
<td>Optimize for leakage power</td>
<td>No</td>
</tr>
<tr>
<td>Optimize for cycle time</td>
<td>Yes</td>
</tr>
<tr>
<td>Temperature (K)</td>
<td>360</td>
</tr>
<tr>
<td>mat_leak_reduction_due_to_sleep_transistors_factor</td>
<td>2</td>
</tr>
<tr>
<td>SRAM cell/wordline technology flavor</td>
<td>ITRS HP</td>
</tr>
<tr>
<td>Peripheral/Global circuitry technology flavor</td>
<td>ITRS HP</td>
</tr>
<tr>
<td>Interconnect projection type</td>
<td>Conservative</td>
</tr>
<tr>
<td>Wire type inside mat</td>
<td>Semi-global</td>
</tr>
<tr>
<td>Wire type outside mat</td>
<td>Semi-global/Global</td>
</tr>
</tbody>
</table>

are two targets for the Intel cache corresponding to the two values of dynamic power shown in Table 9.7. It can be in Figure 9.5 that when an activity factor of 0.1 is assumed the bubbles corresponding to CACTI-D solutions are close to the target Xeon L3 bubble with lower dynamic power, and when the activity factor is assumed to be 1, the CACTI-D bubbles move closer to the target Xeon L3 bubble with higher dynamic power. It can be
Figure 9.5. Access time, area and power of the 65nm Xeon L3 cache and of solutions generated by CACTI-D. The CACTI-D solutions assume ‘semi-global’ wire type outside mat. The 3 plots correspond to 3 activity factors assumed while computing dynamic power for the CACTI-D-generated solutions.

seen from Figure 9.5 that many CACTI-D solutions have area, access time and power that are quite similar to that of the Xeon L3. Table 9.9 shows error percentages of prominent CACTI-D solutions with respect to the Xeon L3. Again it can be seen from these tables that the errors produced by CACTI-D modeling are reasonable.

For the 65nm Xeon L3, Figure 9.6 shows bubble charts and Table 9.10 shows the error percentages for CACTI-D solutions with ‘global’ wire type outside mats. It can be seen that even with global wire type assumed outside mats, the error percentages of the CACTI-D solutions are reasonable.
TABLE 9.9

ERROR PERCENTAGES OF PROMINENT SOLUTIONS GENERATED BY CACTI-D WITH RESPECT TO A 65NM INTEL XEON L3 CACHE WHEN WE ASSUME THAT THE DYNAMIC POWER CONSUMED BY THE CACHE IS 5.4W. THE CACTI-D SOLUTIONS ASSUME ‘SEMI-GLOBAL’ WIRE TYPE OUTSIDE MAT. AN ACTIVITY FACTOR OF 1 HAS BEEN ASSUMED FOR CALCULATION OF DYNAMIC POWER

<table>
<thead>
<tr>
<th>Solution</th>
<th>% error in acc time</th>
<th>% error in area</th>
<th>% error in power</th>
<th>Avg of acc time, area and power % errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best % error in acc time</td>
<td>0</td>
<td>-25</td>
<td>-8</td>
<td>11</td>
</tr>
<tr>
<td>Best % error in area</td>
<td>4</td>
<td>-19</td>
<td>4</td>
<td>9</td>
</tr>
<tr>
<td>Best % error in power</td>
<td>4</td>
<td>-19</td>
<td>4</td>
<td>9</td>
</tr>
<tr>
<td>Best average of area, acc time</td>
<td>4</td>
<td>-19</td>
<td>4</td>
<td>9</td>
</tr>
<tr>
<td>and power % errors</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Best average of area and acc</td>
<td>4</td>
<td>-19</td>
<td>4</td>
<td>9</td>
</tr>
<tr>
<td>time % errors</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Best average of acc time and</td>
<td>4</td>
<td>-19</td>
<td>4</td>
<td>9</td>
</tr>
<tr>
<td>power % errors</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Best acc time</td>
<td>-12</td>
<td>-23</td>
<td>-24</td>
<td>20</td>
</tr>
<tr>
<td>Best area</td>
<td>-12</td>
<td>-23</td>
<td>-24</td>
<td>20</td>
</tr>
<tr>
<td>Best power</td>
<td>-8</td>
<td>-7</td>
<td>-25</td>
<td>14</td>
</tr>
</tbody>
</table>
Figure 9.6. Access time, area and power of the 65nm Xeon L3 cache and of solutions generated by CACTI-D. The CACTI-D solutions assume ‘global’ wire type outside mat. The 3 plots correspond to 3 activity factors assumed while computing dynamic power for the CACTI-D solutions.
TABLE 9.10

ERROR PERCENTAGES OF PROMINENT SOLUTIONS GENERATED BY
CACTI-D WITH RESPECT TO A 65NM INTEL XEON L3 CACHE WHEN WE
ASSUME THAT THE DYNAMIC POWER CONSUMED BY THE CACHE IS 5.4W.
THE CACTI-D SOLUTIONS ASSUME ‘GLOBAL’ WIRE TYPE OUTSIDE MAT. AN
ACTIVITY FACTOR OF 1 HAS BEEN ASSUMED FOR CALCULATION OF
DYNAMIC POWER

<table>
<thead>
<tr>
<th>Solution</th>
<th>% error in acc time</th>
<th>% error in area</th>
<th>% error in power</th>
<th>Avg of acc time, area and power % errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best % error in acc time</td>
<td>-19</td>
<td>-1</td>
<td>-12</td>
<td>11</td>
</tr>
<tr>
<td>Best % error in area</td>
<td>-20</td>
<td>-8</td>
<td>-5</td>
<td>11</td>
</tr>
<tr>
<td>Best % error in power</td>
<td>-20</td>
<td>-8</td>
<td>-5</td>
<td>11</td>
</tr>
<tr>
<td>Best average of area, acc time and power % errors</td>
<td>-20</td>
<td>-8</td>
<td>-5</td>
<td>11</td>
</tr>
<tr>
<td>Best average of area and acc time % errors</td>
<td>-19</td>
<td>-1</td>
<td>-12</td>
<td>11</td>
</tr>
<tr>
<td>Best average of acc time and power % errors</td>
<td>-20</td>
<td>-8</td>
<td>-5</td>
<td>11</td>
</tr>
<tr>
<td>Best acc time</td>
<td>-29</td>
<td>-8</td>
<td>-27</td>
<td>21</td>
</tr>
<tr>
<td>Best area</td>
<td>-29</td>
<td>-8</td>
<td>-27</td>
<td>21</td>
</tr>
<tr>
<td>Best power</td>
<td>-27</td>
<td>16</td>
<td>-29</td>
<td>24</td>
</tr>
</tbody>
</table>
9.2 DRAM Validation

In order to validate the DRAM models, we compare projections produced by CACTI-D against timing and power data of a 78nm Micron 1Gb DDR3-1066 x8 DRAM device [4]. While the area efficiency of the actual device is not known, we assume its area efficiency to be 56% based on the area efficiency value specified in the ITRS [77] for a 6$F^2$ cell based DRAM. For timing, we refer to the datasheet of the DRAM device and for power we use the DDR3 Micron power calculator [5]. Table 9.11 shows the timing and power data obtained for the 78nm Micron device and the error values with respect to the chosen CACTI-D solution. Multiple CACTI-D solutions are feasible with varied area/delay/power tradeoffs; because of the premium on price per bit of commodity DRAM we select one with high area efficiency. The ACTIVATE, READ, and WRITE energies for the Micron device have been computed by appropriate specification of system usage conditions in the Micron power calculator, and then using the values of power components and average delay between commands that it generates. Note that ACTIVATE energy includes energy due to both activation and precharging. It can be seen that there is good agreement in all metrics with an average error of 16%.
RESULTS OF CACTI-D DRAM MODEL VALIDATION WITH RESPECT TO A 78NM MICRON 1GB DDR3-1066 X8 DRAM

<table>
<thead>
<tr>
<th>Metric</th>
<th>Actual value</th>
<th>CACTI-D error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Area efficiency</td>
<td>56%</td>
<td>-6.2%</td>
</tr>
<tr>
<td>Activation delay ($t_{RCD}$) (ns)</td>
<td>13.1</td>
<td>4.5%</td>
</tr>
<tr>
<td>CAS latency (ns)</td>
<td>13.1</td>
<td>-5.8%</td>
</tr>
<tr>
<td>Row cycle time ($t_{RC}$) (ns)</td>
<td>52.5</td>
<td>-8.2%</td>
</tr>
<tr>
<td>ACTIVATE energy (nJ)</td>
<td>3.1</td>
<td>-25.2%</td>
</tr>
<tr>
<td>READ energy (nJ)</td>
<td>1.6</td>
<td>-32.2%</td>
</tr>
<tr>
<td>WRITE energy (nJ)</td>
<td>1.8</td>
<td>-33%</td>
</tr>
<tr>
<td>Refresh power (mW)</td>
<td>3.5</td>
<td>29%</td>
</tr>
</tbody>
</table>

116
In this chapter we demonstrate the utility of CACTI-D by first carrying out studies that demonstrate its parametrization features related to solution optimization and technology. We then use CACTI-D to carry out an architectural study. First in Section 10.1, we look at the impact of the solution optimization features of CACTI-D. Next in Section 10.2 we look at the impact of considering different device types in the memory array. In Section 10.3 we look at the impact of different assumptions in interconnect technology and in Section 10.4, we look at the impact of RAM technology. Finally in Section 10.5, we illustrate the potential applicability of CACTI-D to the design and analysis of future memory hierarchies by carrying out an architectural study of stacked last level SRAM, LP-DRAM, and COMM-DRAM caches for a multicore multithreaded architecture at the 32nm technology-node.

Table 10.1 shows the default parameter values that we have used while carrying out the studies in Sections 10.1 through 10.4. For these studies we use plain RAMs. For each study, we present charts that show the following metrics: access time, random cycle time, area, dynamic energy per read access, and standby leakage power.

10.1 Impact of CACTI-D Solution Optimization

As was mentioned earlier in Section 2.2, in CACTI-D, we adopt a new approach to find the optimal solution. Our new approach allows users to exercise more control on
TABLE 10.1

DEFAULT CACTI-D INPUT PARAMETERS USED FOR STUDIES IN THIS CHAPTER

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capacity (MB)</td>
<td>16</td>
</tr>
<tr>
<td>Output width (bits)</td>
<td>512</td>
</tr>
<tr>
<td>Number of banks</td>
<td>1</td>
</tr>
<tr>
<td>Number of read/write ports</td>
<td>1</td>
</tr>
<tr>
<td>Number of exclusive read ports</td>
<td>0</td>
</tr>
<tr>
<td>Number of exclusive write ports</td>
<td>0</td>
</tr>
<tr>
<td>Technology-node (nm)</td>
<td>65</td>
</tr>
<tr>
<td>RAM cell technology</td>
<td>SRAM</td>
</tr>
<tr>
<td>max_area_constraint (%)</td>
<td>40</td>
</tr>
<tr>
<td>max_acc_time_constraint (%)</td>
<td>10</td>
</tr>
<tr>
<td>max_repeater_delay_constraint (%)</td>
<td>10</td>
</tr>
<tr>
<td>Optimize for dynamic energy</td>
<td>No</td>
</tr>
<tr>
<td>Optimize for dynamic power</td>
<td>No</td>
</tr>
<tr>
<td>Optimize for leakage power</td>
<td>No</td>
</tr>
<tr>
<td>Optimize for cycle time</td>
<td>Yes</td>
</tr>
<tr>
<td>Temperature (K)</td>
<td>360</td>
</tr>
<tr>
<td>mat_leak_reduction_due_to_sleep_transistors_factor</td>
<td>1</td>
</tr>
<tr>
<td>SRAM cell/wordline technology flavor</td>
<td>ITRS HP</td>
</tr>
<tr>
<td>Peripheral/Global circuitry technology flavor</td>
<td>ITRS HP</td>
</tr>
<tr>
<td>Interconnect projection type</td>
<td>Conservative</td>
</tr>
<tr>
<td>Wire type inside mat</td>
<td>Semi-global</td>
</tr>
<tr>
<td>Wire type outside mat</td>
<td>Semi-global</td>
</tr>
</tbody>
</table>

area, delay and power of the final solution. The optimization is carried out in the following steps: first, we find all solutions with area efficiency that is within a certain percentage (user-supplied value) of the area efficiency of the solution with best area efficiency. We refer to this area constraint as max_area_constraint. Next, from this reduced set of solutions that satisfy the max_area_constraint, we find all solutions with access time that is within a certain percentage of the best access time solution. We refer to this access time constraint as max_acc_time_constraint. To the subset of solutions that results after the application
of max\_acc\_time\_constraint, we apply the optimization function shown as Equation 2.1 in Section 2.2. Figure 10.1 shows the impact of varying max\_area\_constraint for a 16MB SRAM. As max\_area\_constraint is increased, the number of subarrays in the SRAM is allowed to grow, and so the area grows steadily. As the number of subarrays increases, the components of delay within a mat decreases and the access time falls up to a point after which it starts to increase again. The random cycle time keeps decreasing as the number of subarrays increases because the wordline and bitline delays keep getting smaller. The trend for dynamic read energy per access shows some up-and-down variation. The energy consumption of the SRAM is mainly a function of the energy consumption of the bitlines, and request and reply networks. In Figure 10.1, we have also shown the number of “cell heights” that are accessed for each max\_area\_constraint. It can be seen that the energy consumption in the bitlines is correlated to the number of “cell heights” that are accessed. In general, energy consumption of the bitlines decreases with max\_area\_constraint whereas the energy consumption of the request and reply networks increases. The minimum dynamic read energy per access occurs for a max\_area\_constraint of 30%. The standby leakage power keeps growing as max\_area\_constraint is increased and the area of the RAM keeps increasing.

Figure 10.2 shows the impact of varying max\_acc\_time\_constraint. For the default set of SRAM parameters shown in Table 10.1, in which the SRAM is optimized for dynamic random cycle time, it can be seen that the solution with best access time (corresponding to max\_acc\_time\_constraint of 0) also has the best area, dynamic read energy per access, and standby leakage power. Relaxation of the max\_acc\_time\_constraint only improves random cycle time and does not provide any other benefits for this case. In Figure 10.2, we also show the trends when the SRAM is optimized for dynamic read energy per access. In this case, for a max\_acc\_time\_constraint of 10% and a degradation of 18% in access time, there are improvements of 17% in dynamic read energy per access, and 5% in
Figure 10.1. Access time, random cycle time, area, dynamic read energy per access and leakage power of a 16MB SRAM as max_area_constraint is varied.

standby leakage power. Random cycle time, however, degrades by 219% and area by 14%. For a max_acc_time_constraint of 10% and a degradation of 4% in access time, however, the degradation in random cycle and area are better at 34% and 9% respectively. Corresponding improvements in dynamic read energy per access and standby leakage power are at 12% and 3% respectively.
Figure 10.2. Access time, random cycle time, area, dynamic read energy per access and leakage power of a 16MB SRAM as max_acc_time_constraint is varied.

Figure 10.3 shows the impact of varying max_repeater_delay_constraint. The max_repeater_delay_constraint changes the separation distance and sizing of repeaters/buffers in the H-tree networks and is useful for trading off delay for energy benefits. It can be seen here that varying max_repeater_delay_constraint does not lead to energy savings much unless the access time is allowed to degrade heavily. Initially, as
max_repeater_delay_constraint is increased from 0 to 20%, it can be seen that the access time does not change and there are no energy savings. This is because of the maximum limit that we have imposed on transistor size in CACTI-D. For values of max_repeater_delay_constraint between 0 and 20%, the sizing of the repeater comes out to be larger than the maximum allowed transistor size and is therefore being fixed at the maximum allowed transistor size (the maximum allowed transistor size was fixed at 100*feature size for NMOS transistors). For a max_repeater_delay_constraint of 50%, the access time degrades by about 4% for about a 14% improvement in dynamic read energy per access. The best improvement in dynamic read energy per access of about 20% is for a max_repeater_delay_constraint of 100%. This improvement comes without changing the area of the solution but with a corresponding degradation of almost 27% in the access time.

Figure 10.4 shows the impact of optimizing the solution generated by CACTI-D for a 16MB SRAM in different ways. Table 10.2 shows the different optimization scenarios targeting metrics of random cycle time, dynamic read energy per access, dynamic power and standby leakage power. The percentage variation between the worst and best values for each metric shown in Figure 10.4 is as follows: access time (5%), random cycle time (297%), area (27%), dynamic read energy per access (39%), and standby leakage power (24%). These variations illustrate the dependence of memory performance estimation on the kind of optimization that is applied.

10.2 Impact of Device Technology

Figure 10.5 illustrates the tradeoffs associated with assuming different types of devices in the memory cells/wordline drivers and the rest of the peripheral/global circuitry. Three scenarios are considered:

1. ITRS HP only;
Figure 10.3. Access time, random cycle time, area, dynamic read energy per access and leakage power of a 16MB SRAM as max_repeater_delay_constraint is varied.

2. ITRS LSTP (memory cells/wordline drivers) + ITRS HP (peripheral/global circuitry);

3. ITRS LSTP only;

In order to capture the impact of device technology only, we assume the same organization for all three scenarios, and choose the organization corresponding to presence
Figure 10.4. Access time, random cycle time, area, dynamic read energy per access and leakage power of a 16MB SRAM under different optimization function scenarios.

of ITRS HP transistors in both memory cells/wordline drivers and peripheral/global circuitry. With respect to “ITRS HP only”, on average over the considered capacities, “ITRS LSTP + ITRS HP” exhibits an improvement of 74% in the standby leakage power. Areas and dynamic read energies per access remain almost identical, and the improvement comes at the cost of 8% worse access time and 62% worse random cycle time. “ITRS
TABLE 10.2

DIFFERENT SOLUTION OPTIMIZATION SCENARIOS TARGETING METRICS OF RANDOM CYCLE TIME, DYNAMIC READ ENERGY PER ACCESS, DYNAMIC POWER AND STANDBY LEAKAGE POWER

<table>
<thead>
<tr>
<th>Optimization Scenario</th>
<th>Optimize for random cycle time</th>
<th>Optimize for dynamic energy</th>
<th>Optimize for dynamic power</th>
<th>Optimize for leakage power</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>B</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>C</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>D</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>E</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>F</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

LSTP only” shows an improvement of almost 100% in standby leakage power with respect to “ITRS HP only”. This improvement comes at a cost of 170% worse access time, 229% worse random cycle time, and 10% worse dynamic read energy per access.

10.3 Impact of Interconnect Technology

Figure 10.6 illustrates the dependence of memory performance on interconnect technology assumptions. As described in Chapter 6 on “Technology Modeling”, instead of assuming a single set of scaling assumptions for interconnect technology, we consider aggressive and conservative scaling projections as in [36, 38]. For each SRAM capacity for the aggressive interconnect technology assumptions we use the same organization that is generated for the conservative interconnect technology assumptions. For each RAM capacity the area, random cycle time, dynamic read energy per access, and standby leakage remain more or less the same with either aggressive or conservative technology assumptions. The lower resistance per unit length of the aggressive projections leads to lowering of the access time by about 10% on an average. For smaller technologies, the impact of
Figure 10.5. Access time, random cycle time, area, dynamic read energy per access and leakage power of SRAMs for different 65nm device technology assumptions.

Interconnect technology assumptions would be more. Figure 10.7 shows the impact of wire type on memory performance. As described in Chapter 6 on “Technology Modeling”, wires outside a mat can be of either ‘semi-global’ or ‘global’ type. For either wire type, we use the same organization, the one that is generated for semi-global type. Global type wire outside mats leads to an improvement of access time by 22% and dynamic read energy per access by 8% with the random cycle time and standby leakage power remain-
ing more or less the same. The area, however, gets worse by 28% because of the greater pitch of the global wire type.

Figure 10.6. Access time, random cycle time, area, dynamic read energy per access and leakage power of SRAMs under aggressive and conservative interconnect technology assumptions.
Figure 10.7. Access time, random cycle time, area, dynamic read energy per access and leakage power of SRAMs when “semi-global” or “global” wire type is used for wires outside a mat.
10.4 Impact of RAM Technology

Figure 10.8 illustrates the dependence of memory performance on the type of RAM cell technology – SRAM, LP-DRAM, and COMM-DRAM. For each capacity and RAM technology, we consider CACTI-D solutions with the best access time amongst all solutions. It can be seen that up to a capacity close to 4MB, the access time of SRAM is lower than that of LP-DRAM after which access time of LP-DRAM is better. For the larger capacities, the decreased interconnect delay of the LP-DRAMs because of their smaller area reduces their access time. On average, the area of the LP-DRAMs is smaller than that of the SRAMs by a factor of 2.6. Because of the destructive readout and subsequent writeback, it can be seen that random cycle time of the LP-DRAMs is higher than that of the SRAMs; on average it’s about 2.9 times worse. Multisubbank interleave cycle time of the LP-DRAMs is lower than the random cycle time of the SRAMs for capacities less than around 2MB and then it gets worse. With further pipelining in the request and reply networks, multibank interleave cycle time can be made smaller. Dynamic read energy per access of the LP-DRAMs is worse than that of SRAMs by a factor of 2 on average for capacities less than 1MB, but the values are very similar for capacities equal to or greater than 1MB. On average the standby leakage of the LP-DRAMs is lower than that of the SRAMs by a factor of about 5.

Access time of the COMM-DRAMs is consistently worse than that of the LP-DRAMs; on average it’s worse by a factor of 2. Random cycle time of the COMM-DRAMs is also worse than that of the LP-DRAMs and it gets more worse as capacity increases; on average it’s worse by a factor of about 2. On average, multisubbank interleave cycle time of the COMM-DRAMs is worse than that of the LP-DRAMs by a factor of 2.3. Area of the COMM-DRAMs is smaller than that of the LP-DRAMs by a factor of about 3. Dynamic read energy per access of the COMM-DRAMs is lower than that of the LP-DRAMs for some capacities and higher for others. For the largest two RAMs that we have considered
(16MB and 32MB), the dynamic energy of COMM-DRAM technology is lower. The standby leakage power of COMM-DRAMs is three orders of magnitude smaller than that of the LP-DRAMs.
Figure 10.8. Access time, cycle time, area, dynamic read energy per access and standby leakage power of 65nm SRAM, LP-DRAM and COMM-DRAM technologies. Area, dynamic read energy per access and standby leakage power are split up into two charts based on capacity.
10.5 Architectural Study

We illustrate the utility of CACTI-D by applying it to the design and architectural evaluation of stacked last level SRAM, LP-DRAM, or COMM-DRAM based caches at the 32nm node for a multicore multithreaded processor architecture [84].

10.5.1 System Architecture

Figure 10.9 shows a block diagram of our assumed system architecture. The processor chip is assumed to be composed of a 2-layer die stack with the top die implementing the shared last level (L3) cache. The bottom core die is composed of 8 multithreaded 32nm Niagara-like cores with one 4-way SIMD FPU per core. Each core is assumed to have 32KB 8-way set-associative private SRAM based L1 instruction and data caches and a 1MB 8-way set-associative private SRAM based unified L2 cache. The main memory subsystem is assumed to be composed of two channels with each channel connected to a single-ranked 8GB DIMM that is made up of 8Gb DDR4-3200 devices which are also assumed to be fabricated on 32nm technology.

We assume that the LLC is composed of 8 banks connected to the 8 L2 banks of the core die through a crossbar implemented in the core die. For the 3D connections, we assume face-to-face through-silicon via technology similar to that described in [68] which has sub-FO4 communications delays. We compute the area occupied by the bottom die by scaling the area of Niagara [49] core components to 32nm and using CACTI-D to compute the area of the L1 and L2 caches. The area available per LLC bank is fixed to be 1/8th of the bottom core area and this comes out to be $6.2\text{mm}^2$.

10.5.2 Application Benchmarks

SPEC benchmarks fit in caches much smaller than the 192MB caches we study, and many other benchmarks have had their data sets limited to reduce simulation running
time. In order to exercise our very large caches, we use a subset of NAS Parallel Benchmark (NPB) [42] applications in our chip multiprocessor system LLC study. We scaled the version 3.2.1 NPB applications to match our simulated system. We skipped the initialization phase of each benchmark, and then started timing simulation and data collection. We executed 10 billion instructions after initialization. OpenMP versions of NPB applications were used to distribute the workload across 32 hardware threads. The benchmarks were compiled with Intel C/Fortran compilers (ver. 10.1).

10.5.3 Simulation Methodology

For architectural simulation, we use COTSon [31] developed at HP Labs. A full-system emulator executes each multithreaded NPB application on top of the guest operating system (Linux) and generates instruction sequences with supplementary information
### TABLE 10.3

**PROJECTIONS OF KEY PROPERTIES OF THE VARIOUS CACHES AND MAIN MEMORY CHIP AT THE 32NM NODE**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>L1</th>
<th>L2</th>
<th>L3</th>
<th>Main Mem</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SRAM</td>
<td>LP DRAM</td>
<td>COMM DRAM</td>
<td>DRAM chip</td>
</tr>
<tr>
<td>Capacity</td>
<td>32KB</td>
<td>1MB</td>
<td>24MB</td>
<td>48MB</td>
</tr>
<tr>
<td>Number of banks</td>
<td>1</td>
<td>1</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>Number of subbanks</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>32</td>
</tr>
<tr>
<td>Associativity</td>
<td>8</td>
<td>8</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>Clock frequency, w.r.t. CPU</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Access time (CPU cycles)</td>
<td>2</td>
<td>3</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>Random cycle time (CPU cycles)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Area ((mm^2))</td>
<td>0.17</td>
<td>2.0</td>
<td>49.6</td>
<td>45.6</td>
</tr>
<tr>
<td>Area efficiency (%)</td>
<td>25</td>
<td>67</td>
<td>64</td>
<td>36</td>
</tr>
<tr>
<td>Standby/Leakage power (W)</td>
<td>0.009</td>
<td>0.157</td>
<td>3.6</td>
<td>2.0</td>
</tr>
<tr>
<td>Refresh power (W)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.3</td>
</tr>
<tr>
<td>Dynamic read energy per cache line (nJ)</td>
<td>0.07</td>
<td>0.27</td>
<td>0.54</td>
<td>0.54</td>
</tr>
</tbody>
</table>

like thread ID and synchronization primitives annotated. The timing simulator takes these instruction sequences, distributes them into the simulated cores by thread mapping, and executes them while following constraints imposed by synchronization primitives such as locks and barriers. Four hardware threads are executed concurrently in a simulated core. On average each thread is modeled to execute one instruction every 4 cycles with up to one memory request generated to the L1 cache per cycle. In order to model SIMD execution, a floating point arithmetic instruction is assumed to be executed every cycle. For cache coherence, a MESI protocol is modeled.
10.5.4 Results

CACTI-D Cache Projections

We use CACTI-D to obtain projections for all levels of the memory hierarchy including L1, L2, and L3 caches, and main memory DRAM chips. With the optimization methodology of CACTI-D, for a given memory/cache input specification it is possible to obtain various feasible solutions that exhibit tradeoffs in area, access time, cycle times, dynamic energy, and leakage power. We apply optimizations and selected appropriate configurations for each level. Table 10.3 shows the values of key metrics of the caches and main memory used in our architectural study. For this study, we fix the clock frequency of the processor cores to 2GHz based on the access time of the 32KB L1 instruction/data cache.

For the 8-banked L3 cache, we consider 5 design options: 24MB (SRAM), 48MB and 72MB (LP-DRAM), and 96MB and 192MB (COMM-DRAM). For each DRAM technology, we choose one solution that is optimized for capacity (config C) and another one that uses smaller mats with better energy and delay properties (config ED) and the same associativity as the SRAM L3. We choose the clock domain for each cache configuration by limiting the maximum number of pipeline stages inside any of the caches to 6. Note that the access times shown in Table 10.3 are just for cache access and do not include communication/control delays. It can be seen that the access time and random cycle time of the 6MB LP-DRAM L3 bank is the same as that of the 3MB SRAM L3 bank. The access times of the COMM-DRAM configurations are about 3 times higher than that of comparable LP-DRAM configurations but still much smaller than that of the main memory DRAM chip because of their smaller capacity\(^1\). By sacrificing capacity, the 48MB LP-DRAM and the 96MB COMM-DRAM are able to have lower dynamic energy per read access than the 72MB LP-DRAM and the 192MB COMM-DRAM respectively. Be-

\(^1\)The access time shown for the main memory DRAM chip is the sum of \(t_{RCD}\) and CAS latency.
cause of the large access granularity of a main memory DRAM access and because 8 chips have to be accessed in parallel, the dynamic energy of a read access is very high compared to L3 access. Leakage of both LP-DRAM L3s (2.0W and 2.1W) is less than that of the SRAM L3 (3.6W) even though we have modeled an aggressive leakage control mechanism for the SRAM L3 similar to that of the 65nm Xeon [21].

We compute miss penalties for use inside the architectural simulator by considering cache hit times, tag array access times, and communication and control delays involved in transfer of information between cache levels. While computing miss penalty on an L2 miss we consider the communication delay through the crossbar between L2 and L3. Inside CACTI-D we incorporate a model [93] for the delay and energy consumed in a crossbar. We compute the length of the interconnect route between L2 and L3 by measuring the dimensions of the 8x8 crossbar from the Niagara2 die photo and then scaling to 32nm.

IPC

For all applications and cache configurations, Figure 10.10(a) shows the IPC and average read latency, and Figure 10.10(b) shows the components of execution time. In Figure 10.10(b) the execution time is broken down into six categories: threads are processing instructions that are not waiting for memory requests; threads are stalled and L2 caches are serving requests; threads are stalled and L3 caches are serving requests; threads are stalled and main memories are serving requests; threads are idle waiting for all the threads to reach a barrier; threads are waiting for other threads to release locks. From Figure 10.10(b), it can be seen that memory access time takes up a majority of the execution cycles. For most of the applications, introduction of a shared L3 cache leads to IPC improvement. On average across all applications, the SRAM, LP-DRAM ED, LP-DRAM C, COMM-DRAM ED, and COMM-DRAM C L3s show IPC improvements of
Figure 10.10. (a) IPC and average read latency, and (b) Components of execution time of NPB applications on different cache configurations.

27%, 35%, 38%, 39%, and 42% respectively. The IPC improvement has good correlation with the average read latency improvement because the threads are executed in order. Amongst all applications, ft.B and lu.C show the highest IPC improvement with the incorporation of a shared L3 cache. For both these applications, the LP-DRAM L3s have better IPCs compared to the SRAM and COMM-DRAM L3s. For both these applications, the working set does not fit in the L2 cache and presence of the shared L3 cache benefits these applications greatly. The LP-DRAM L3s perform better than the COMM-DRAM L3s for these applications because they achieve sufficiently low L3 miss rates for these
applications and have better latency and bandwidth properties. For applications ua.C and cg.C, the IPC does not improve much with the introduction of the L3 cache. For ua.C, the number accesses that go to L3 are few, and cg.C exhibits poor locality. For the remaining applications, bt.C, is.C, mg.B, and sp.C, the COMM-DRAM L3s show the best IPC improvement. For these applications, the working set size is bigger than the capacity of the L3 caches. The larger capacities reduce the number of main memory accesses, reduce the average main memory access time and improve IPC.

Memory Hierarchy Power and System Energy-Delay

For each application Figure 10.11(a) shows the breakdown of power consumed in the memory hierarchy of the different system configurations. We assume a memory bus power of 2mW/Gb/s suitable for the 2013 time-frame. On average, over all applications, the memory hierarchy with no L3 consumes 6.6W, which accounts for 23% of system power. The addition of an LLC has the potential to reduce power by reducing the number of accesses to main memory, thereby saving on bus power and dynamic power consumed in the memory chips. However, addition of an LLC can increase memory hierarchy power by consuming standby/leakage and refresh power. For each application, addition of the logic process based SRAM or LP-DRAM L3s results in an increase in the memory hierarchy power because reduction in main memory and bus power is not enough to overcome the additional leakage of these logic process based caches. On average, the SRAM, LP-DRAM config ED, and LP-DRAM config C increase memory hierarchy power by 58%, 37%, and 35% respectively. The COMM-DRAM L3s perform best by increasing the memory hierarchy power by the least amount. For bt.C, ft.B, lu.C, and ua.C, the COMM-DRAM L3s are successful in reducing the memory hierarchy power slightly. Overall, on average, the 96MB config ED and the 192MB config C COMM-DRAM L3s increase memory hierarchy power by 1.2% and 2.3% respectively. Thus the memory hierarchy
power of ED and C configurations are not that different. For either DRAM technology, the lower dynamic energy of config ED does not help much because L3 dynamic power is a relatively small percentage of total memory hierarchy power. Similarly the greater capacity and worse dynamic energy of config C does not hurt much in terms of power because both config C and config ED have comparable leakage power. These trends when combined with the fact that performance of the benchmarks is not too sensitive to L3 access time and random cycle time suggest that leakage power should be a primary consideration in design of L3.
In all system configurations including those with COMM-DRAM L3s, the main power drain in the memory hierarchy is the main memory chips. In the system with the 192MB COMM-DRAM L3, on average, dynamic power in main memory DRAMs accounts for 26% of the memory hierarchy power while standby accounts for 22%. Our system architecture included only two DIMMs distributed over two channels. In systems with more memory capacity per channel, standby power in the memory chips can account for even greater percentage and become a dominant component of system power.

For each application and for each system configuration, Figure 10.11(b) shows breakdown of system power into core/uncore power and memory hierarchy power, and system energy-delay product that is normalized with respect to the system with no L3. We compute core/uncore power by scaling the power of the 90nm Niagara (63W) to 32nm. We scale the power by assuming linear scaling for capacitance, increase in CPU clock from 1.2GHz to 2GHz, reduction in VDD from 1.2V to 0.9V, and 40% of power to be leakage power. We also adjust core/uncore power suitably to account for the fact that the processor die that we have considered has 8 4-way SIMD FPUs per core whereas 90nm Niagara had just 1 FPU per chip. With these assumptions, we compute total core/uncore power of the bottom die to be 22.3W.

Because of the high contribution of standby/leakage power in both memory hierarchy power and core power, when the addition of an L3 results in reduction of execution time, it is also beneficial for system energy-delay. Because of its high leakage power, the SRAM L3 is able to improve system energy-delay for only 4 applications even though it improves execution time of 7. On average the SRAM L3 improves system energy-delay by 12%. The LP-DRAM L3s do better than the SRAM L3 with the 48MB and 72MB LP-DRAM L3s improving system-energy delay by 18% and 24% respectively. The COMM-DRAM L3s perform best with the 96MB and 192MB COMM-DRAM L3s improving system energy-delay by 33% and 40% respectively.
We also studied the temperature increase due to stacking of L3 die of different technologies by using HotSpot [80]. The maximum power density is with the stacked SRAM L3, but even for this scenario because we have modeled long-channel HP transistors and an aggressive leakage control mechanism, the maximum power consumed per bank is only about 450mW. The maximum observed temperature difference between the different technologies was less than 1.5K.
CHAPTER 11

CONCLUSIONS

In this thesis we have described the modeling of a comprehensive memory modeling tool we call CACTI-D that supports modeling of logic process based DRAM (LP-DRAM) and commodity DRAM (COMM-DRAM) technologies in addition to SRAM. CACTI-D includes a number of major improvements over version 4 of the well-known memory modeling CACTI. The base technology modeling has been changed from simple linear scaling of the original 0.8 micron technology to models based on the ITRS roadmap. Data for different ITRS device types has been incorporated in CACTI-D. Interconnect technology data has also been updated so that it is now based off well-documented models and data. This has been achieved by an extensive revamp and rewrite of the CACTI code base. Various organizational and circuit assumptions have been clarified and updated. The modeling has also been restructured in such a way that it is now more modular and easier to extend and evolve. With CACTI-D, modeling of the complete memory hierarchy with consistent models all the way from SRAM based L1 caches through main memory DRAMs on DIMMs now becomes possible.

Area, delay and power projections from CACTI-D have been compared against published data available for two prominent 90nm and 65nm SRAM caches and a 78nm DDR3 DRAM chip. Taking into account the extremely generic nature of CACTI-D, there is good agreement between the results produced by CACTI-D and the published data.

We illustrated the utility of CACTI-D by carrying out a study of last level cache (LLC)
tradeoffs involving SRAM, LP-DRAM, and COMM-DRAM L3 caches for a multicore multithreaded architecture. We used CACTI-D to model the area, access time, dynamic read/write energy per access, and standby/leakage power of all memory components of our study including L1, L2, and last level L3 caches, and main memory DRAMs. Then we carried out an architectural study to evaluate the different system configurations through simulation of benchmarks with large data set sizes. On average, execution time of applications was reduced by the addition of any of the considered L3s, but the 96MB and 192MB COMM-DRAM L3s achieved the best reduction of 39% and 43% respectively. The COMM-DRAM LLCs also performed best in terms of system power and system energy-delay product.

The work carried out in this thesis can be extended in several ways. Currently CACTI-D supports SRAM, LP-DRAM, and COMM-DRAM technologies. However, there are other interesting memory technologies such as Innovative Silicon’s Z-RAM [57], MoSys’ 1T-SRAM [34], and T-RAM Semiconductor’s T-RAM [63]. It would be good if CACTI-D could support these memory technologies as well. Also, non-volatile memories (NVMs) [2, 3, 7] are gaining in prominence and their importance is expected to grow in future memory systems. It would be interesting to investigate modeling of NVMs also within the CACTI-D framework. With 3D stacking technology, it’s likely that future chips will be composed of heterogeneous memory elements, so it would be good to extend CACTI-D to support all these diverse technologies.

CACTI-D currently supports modeling of only a few basic main memory specifications. In reality, main memory chips have much more sophisticated specifications related to timing and power. It would be interesting to extend CACTI-D to capture more of these specifications. DRAMsim [91] is a simulation framework that can simulate the operation of main memory systems. If CACTI-D is combined with DRAMsim the resulting tool can be a powerful tool for design and analysis of main memory DRAM chips.
Wattch [20] is a framework for power analysis of processor microarchitectures and it made use of an earlier version of CACTI as the foundation for technology modeling, and for modeling array-based structures in the microarchitecture. Wattch modeled a conventional superscalar processor. It would be interesting to develop a framework similar to Wattch that makes use of CACTI-D as the foundation for technology modeling and can model contemporary multicore microarchitectures. In [37], Horowitz has suggested that in order to combat the high cost of chip design, “chip generator” tools should be developed. At one level CACTI-D may be thought of as a simplified, stripped-down version of a memory compiler. CACTI-D supports different kinds of memory technologies by capturing their essential differences in a common framework based off consistent assumptions. As shown in this thesis, CACTI-D can be useful for carrying out preliminary architectural studies that compare different memory design options. Once a preliminary architectural study is completed, ultimately memory design would typically be accomplished using a memory compiler. Similarly, we believe a tool that can model different kinds of chips or microarchitectures can be useful for conducting preliminary studies before an actual chip generator is invoked. A framework that could capture the essential features and differences of contemporary processors such as Intel Core [22] and Atom [33], Sun Niagara [49] and Rock [87], IBM Cell [67], MIPS [6] and ARM [1] would be very useful for preliminary architectural studies.

Processing-In-Memory (PIM) architectures [48,66,86] are revolutionary architectures which are based on the idea that processing logic and main memory DRAM are combined together in the same chip, and have been of research interest for a number of years. PIM chips have typically assumed a merged logic and DRAM fabrication process, that is a single homogeneous process that allows processing of both the CPU logic and DRAM. With stacked die technology, however, it becomes possible to envision PIM chips that are assembled out of dies that are manufactured in heterogeneous fabrication processes. A
tool like CACTI-D would be very valuable for carrying out technology based explorations of PIM architectures.


151


