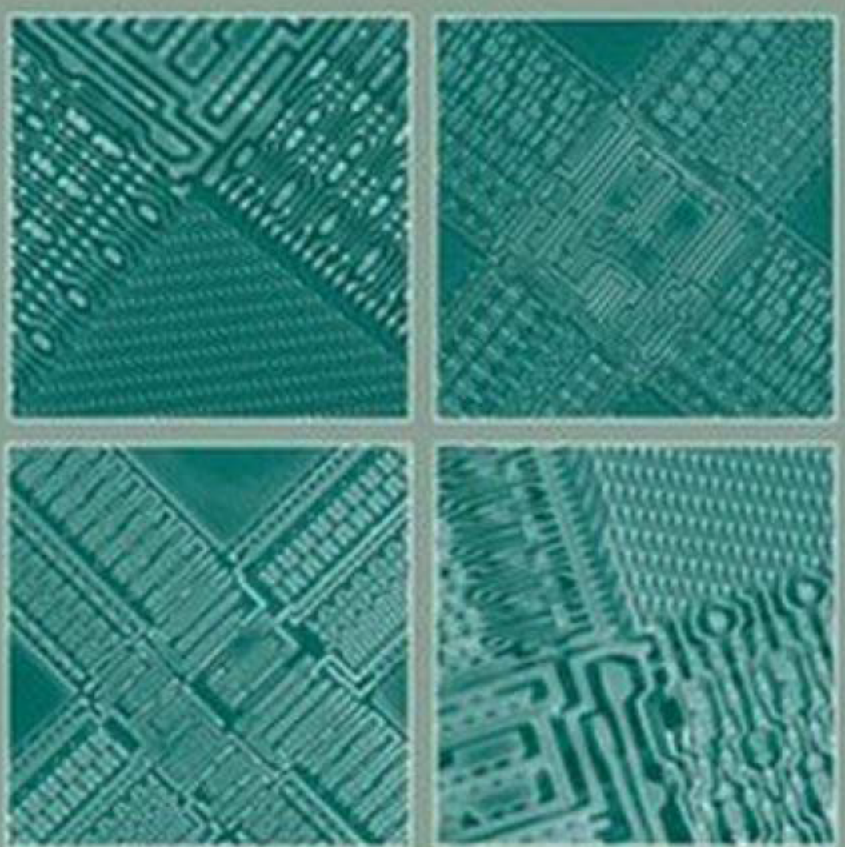
DRAM CIRCUIT DESIGN

Fundamental and High-Speed Topics



Brent Keeth • R. Jacob Baker
Brian Johnson • Feng Lin

DRAM Circuit Design

IEEE Press
445 Hoes Lane
Piscataway, NJ 08854

IEEE Solid-State Circuits Society, Sponsor
IEEE SSCS Liaison to the IEEE Press, Stuart K. Tewksbury

Technical Editing and Illustrations
Mary J. Miller, Micron Technology, Inc.

Technical Reviewers
Jim Frenzel, University of Idaho
Zhao Zhang, Iowa State University

DRAM Circuit Design

Fundamental and High-Speed Topics

Brent Keeth
R. Jacob Baker
Brian Johnson
Feng Lin

IEEE Press Series on Microelectronic Systems
Stuart K. Tewksbury and Joe E. Brewer, *Series Editors*

IEEE Solid-State Circuits Society, *Sponsor*

IEEE PRESS

WILEY-INTERSCIENCE
A John Wiley & Sons, Inc., Publication

Published by John Wiley & Sons, Inc., Hoboken, New Jersey.
Published simultaneously in Canada.

Library of Congress Cataloging-in-Publication Data is available.

ISBN 978-0-470-18475-2

Printed in the United States of America.

10 9 8 7 6 5 4 3 2 1

To Susi, John, Katie, Gracie, Dory, and Faith (B.K.)
To Julie, Kyri, and Josh (J.B.)
To Cassandra, Parker, Spencer, and Dolpha (B.J.)
To Hui, Luan, and Conrad (F.L.)
To Leah and Nicholas (M.M.)

Contents

Chapter 10 Output Circuit Paths

Chapter 11 Timing Circuits

Chapter 12 Control Logic Design

Preface

From the core memory that rocketed into space during the Apollo moon missions to the solid-state memories used in today's commonplace computer, memory technology has played an important, albeit quiet, role during the prior and current centuries. It has been quiet in the sense that memory, although necessary, is not glamorous and sexy, and is instead being relegated to the role of a commodity. Yet, it is important because memory technology, specifically, CMOS DRAM technology, has been one of the greatest driving forces in the advancement of solid-state technology. It remains a driving force today, despite the segmentation in its market space.

The very nature of the commodity memory market, with high product volumes and low pricing, is what ultimately drives the technology. To survive, let alone remain viable over the long term, memory manufacturers work aggressively to drive down their manufacturing costs while maintaining, if not increasing, their share of the market. One of the best tools to achieve this goal remains the ability of manufacturers to shrink their technology, essentially in getting more memory chips per wafer through process scaling. Unfortunately, with all memory manufacturers pursuing the same goals, it is literally a race to see who can get there first. As a result, there is tremendous pressure to advance the state of the art—more so than in other related technologies due to the commodity status of memory.

While the memory industry continues to drive forward, most people can relax and enjoy the benefits—except for those of you who need to join in the fray. For you, the only way out is straight ahead, and it is for you that we have written the second edition of this book.

The goal of *DRAM Circuit Design: Fundamental and High-Speed Topics* is to bridge the gap between the introduction to memory design available in most CMOS circuit texts and the advanced articles on DRAM design that

are available in technical journals and symposium digests. The book introduces the reader to DRAM theory, history, and circuits in a systematic, tutorial fashion. The level of detail varies, depending on the topic. In most cases, however, our aim is merely to introduce the reader to a functional element and illustrate it with one or more circuits. After gaining familiarity with the purpose and basic operation of a given circuit, the reader should be able to tackle more detailed papers on the subject.

The second half of the book is completely devoted to advanced concepts pertaining to state-of-the-art high-speed and high-performance DRAM memory. The two halves of the book are worlds apart in content. This is intentional and serves to rapidly advance the reader from novice to expert through the turning of a page.

The book begins in Chapter 1 with a brief history of DRAM device evolution from the first 1Kbit device to the 64Mbit synchronous devices. This chapter introduces the reader to basic DRAM operation in order to lay a foundation for more detailed discussion later. Chapter 2 investigates the DRAM memory array in detail, including fundamental array circuits needed to access the array. The discussion moves into array architecture issues in Chapter 3, including a design example comparing known architecture types to a novel, stacked digitline architecture. This design example should prove useful, for it delves into important architectural trade-offs and exposes underlying issues in memory design. Chapter 4 then explores peripheral circuits that support the memory array, including column decoders and redundancy. The reader should find Chapter 5 very interesting due to the breadth of circuit types discussed. This includes data path elements, address path elements, and synchronization circuits. Chapter 6 follows with a discussion of voltage converters commonly found on DRAM designs. The list of converters includes voltage regulators, voltage references, $V_{DD}/2$ generators, and voltage pumps.

Chapter 7 introduces the concept of high-performance memory and underlying market forces. Chapter 8 examines high-speed DRAM memory architectures including a discussion of performance-cost trade-offs. Chapter 9 takes a look at the input circuit path of high-performance DRAM while Chapter 10 takes a complementary look at the output circuit path. Chapter 11 dives into the complicated world of high-performance timing circuits including delay-lock-loops and phase-lock-loops. Chapter 12 ties it all together by tackling the difficult subject of control logic design. This is an especially important topic in high-performance memory chips. Chapter 13 looks at power delivery and examines methods to improve performance through careful analysis and design of the power delivery network. Finally, Chapter 14 discusses future work in high-performance memory. We wrap

up the book with the Appendix, which directs the reader to a detailed list of papers from major conferences and journals.

Brent Keeth
R. Jacob Baker
Brian Johnson
Feng Lin

Chapter
1

An Introduction to DRAM

Dynamic random access memory (DRAM) integrated circuits (ICs) have existed for more than thirty years. DRAMs evolved from the earliest kilobit (Kb) generation to the gigabit (Gb) generation through advances in both semiconductor process and circuit design technology. Tremendous advances in process technology have dramatically reduced feature size, permitting ever higher levels of integration. These increases in integration have been accompanied by major improvements in component yield to ensure that overall process solutions remain cost-effective and competitive. Technology improvements, however, are not limited to semiconductor processing. Many of the advances in process technology have been accompanied or enabled by advances in circuit design technology. In most cases, advances in one have enabled advances in the other. In this chapter, we introduce some fundamentals of the DRAM IC, assuming that the reader has a basic background in complementary metal-oxide semiconductor (CMOS) circuit design, layout, and simulation [1].

1.1 DRAM TYPES AND OPERATION

To gain insight into how modern DRAM chips are designed, it is useful to look into the evolution of DRAM. In this section, we offer an overview of DRAM types and modes of operation.

1.1.1 The 1k DRAM (First Generation)

We begin our discussion by looking at the 1,024-bit DRAM (1,024 x 1 bit). Functional diagrams and pin connections appear in Figure 1.1 and Figure 1.2, respectively. Note that there are 10 address inputs with pin labels R_1–R_5 and C_1–C_5. Each address input is connected to an on-chip

address input buffer. The input buffers that drive the *row (R)* and *column (C)* decoders in the block diagram have two purposes: to provide a known *input capacitance* (C_{IN}) on the address input pins and to detect the input address signal at a known level so as to reduce timing errors. The level V_{TRIP}, an idealized trip point around which the input buffers slice the input signals, is important due to the finite transition times on the chip inputs (Figure 1.3). Ideally, to avoid distorting the duration of the logic zeros and ones, V_{TRIP} should be positioned at a known level relative to the maximum and minimum input signal amplitudes. In other words, the reference level should change with changes in temperature, process conditions, *input maximum amplitude* (V_{IH}), and *input minimum amplitude* (V_{IL}). Having said this, we note that the input buffers used in first-generation DRAMs were simply inverters.

Continuing our discussion of the block diagram shown in Figure 1.1, we see that five address inputs are connected through a decoder to the 1,024-bit memory array in both the row and column directions. The total number of addresses in each direction, resulting from decoding the 5-bit word, is 32. The single memory array is made up of 1,024 memory elements laid out in a square of 32 rows and 32 columns. Figure 1.4 illustrates the conceptual layout of this memory array. A memory element is located at the intersection of a row and a column.

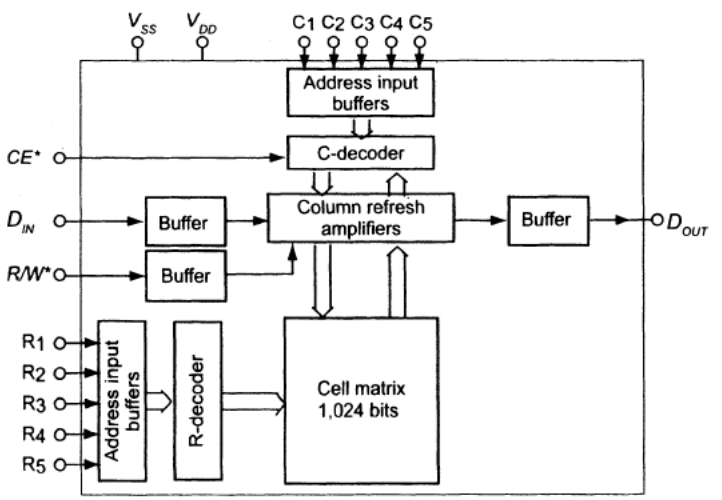


Figure 1.1 1,024-bit DRAM functional diagram.

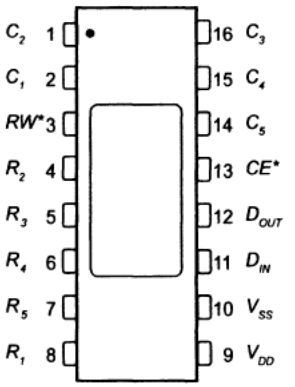


Figure 1.2 1,024-bit DRAM pin connections.

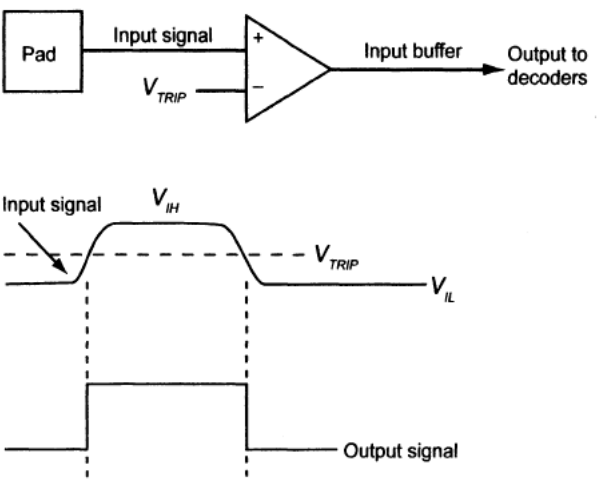


Figure 1.3 Ideal address input buffer.

By applying an address of all zeros to the 10 address input pins, the memory data located at the intersection of row0, *RA*0, and column 0, *CA*0, is accessed. (It is either written to or read out, depending on the state of the *R/W** input and assuming that the *CE** pin is LOW so that the chip is enabled.)

It is important to realize that a single bit of memory is accessed by using both a row and a column address. Modern DRAM chips reduce the number of external pins required for the memory address by using the same pins for both the row and column address inputs *(address multiplexing)*. A clock signal *row address strobe (RAS*)* strobes in a row address and then, on the same set of address pins, a clock signal *column address strobe (CAS*)* strobes in a column address at a different time.

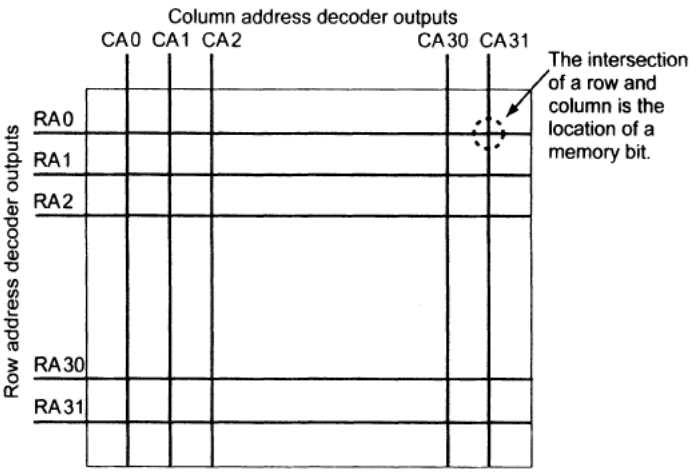


Figure 1.4 Layout of a 1,024-bit memory array.

Also note how a first-generation memory array is organized as a logical square of memory elements. (At this point, we don't know what or how the memory elements are made. We just know that there is a circuit at the intersection of a row and column that stores a single bit of data.) In a modern DRAM chip, many smaller memory arrays are organized to achieve a larger memory size. For example, 1,024 smaller memory arrays, each composed of 256 kbits, may constitute a 256-Meg (256 million bits) DRAM.

1.1.1.1 Reading Data Out of the 1k DRAM. Data can be read out of the DRAM by first putting the chip in the Read mode by pulling the *R/W** pin HIGH and then placing the chip enable pin *CE** in the LOW state. Figure 1.5 illustrates the timing relationships between changes in the address inputs and data appearing on the D_{OUT} pin. Important timing specifications present in this figure are *Read cycle time* (t_{RC}) and *Access time* (t_{AC}). The term t_{RC} specifies how fast the memory can be read. If t_{RC} is 500 ns, then the DRAM can supply 1-bit words at a rate of 2 MHz. The term t_{AC} specifies the maximum length of time after the input address is changed before the output data (D_{OUT}) is valid.

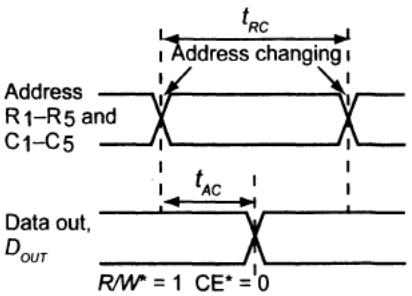


Figure 1.5 **1k** DRAM Read cycle.

1.1.1.2 Writing to the 1k DRAM. Writing data to the DRAM is accomplished by bringing the *R/W** input LOW with valid data present on the D_{IN} pin. Figure 1.6 shows the timing diagram for a Write cycle. The term *Write cycle time (t_{WC})* is related to the maximum frequency at which we can write data into the DRAM. The term *Address to Write delay time (t_{AW})* specifies the time between the address changing and the *R/W** input going LOW. Finally, *Write pulse width (t_{WP})* specifies how long the input data must be present before the *R/W** input can go back HIGH in preparation for another Read or Write to the DRAM. When writing to the DRAM, we can think of the *R/W** input as a clock signal.

1.1.1.3 Refreshing the 1k DRAM. The dynamic nature of DRAM requires that the memory be refreshed periodically so as not to lose the contents of the memory cells. Later we will discuss the mechanisms that lead to the dynamic operation of the memory cell. At this point, we discuss how memory Refresh is accomplished for the 1k DRAM.

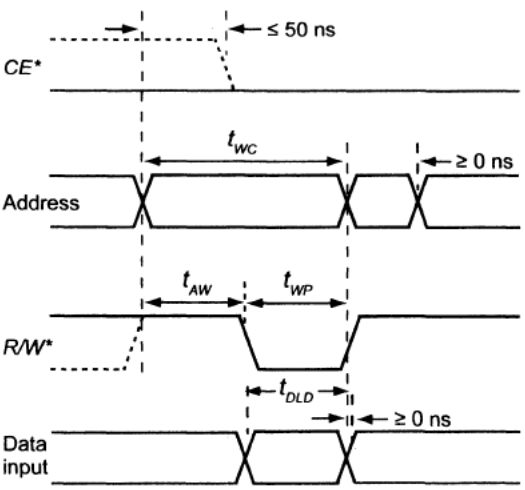


Figure 1.6 1k DRAM Write cycle.

Refreshing a DRAM is accomplished internally: external data to the DRAM need not be applied. To refresh the DRAM, we periodically access the memory with every possible row address combination. A timing diagram for a Refresh cycle is shown in Figure 1.7. With the *CE** input pulled HIGH, the address is changed, while the *R/W** input is used as a strobe or clock signal. Internally, the data is read out and then written back into the same location at full voltage; thus, logic levels are restored (or refreshed).

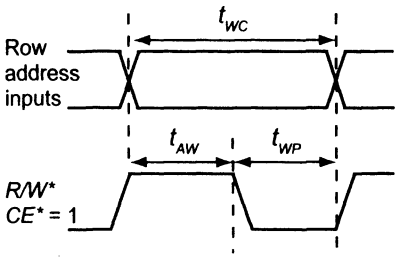


Figure 1.7 1k DRAM Refresh cycle.

1.1.1.4 A Note on the Power Supplies. The voltage levels used in the 1k DRAM are unusual by modern-day standards. In reviewing Figure 1.2, we see that the 1k DRAM chip uses two power supplies: V_{DD} and V_{SS}. To begin, V_{SS} is a greater voltage than V_{DD}: V_{SS} is nominally 5 V, while V_{DD} is −12 V. The value of V_{SS} was set by the need to interface to logic circuits that were implemented using transistor-transistor logic (TTL) logic. The 17 V difference between V_{DD} and V_{SS} was necessary to maintain a large signal-to-noise ratio in the DRAM array. We discuss these topics in greater detail later in the book. The V_{SS} power supply used in modern DRAM designs, at the time of this writing, is generally zero; the V_{DD} is in the neighborhood of 1.5 V.

1.1.1.5 The 3-Transistor DRAM Cell. One of the interesting circuits used in the 1k DRAM (and a few of the 4k and 16k DRAMs) is the 3-transistor DRAM memory cell shown in Figure 1.8. The column- and rowlines shown in the block diagram of Figure 1.1 are split into Write and Read line pairs. When the Write rowline is HIGH, M1 turns ON. At this point, the data present on the Write columnline is passed to the gate of M2, and the information voltage charges or discharges the input capacitance of M2. The next, and final, step in writing to the mbit cell is to turn OFF the Write rowline by driving it LOW. At this point, we should be able to see why the memory is called dynamic. The charge stored on the input capacitance of M2 will leak off over time.

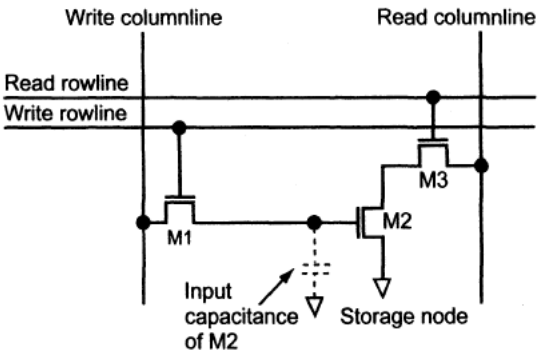


Figure 1.8 3-transistor DRAM cell.

If we want to read out the contents of the cell, we begin by first precharging the Read columnline to a known voltage and then driving the Read rowline HIGH. Driving the Read rowline HIGH turns M3 ON and allows M2 either to pull the Read columnline LOW or to not change the precharged voltage of the Read columnline. (If M2's gate is a logic LOW, then M2 will be OFF, having no effect on the state of the Read columnline.) The main drawback of using the 3-transistor DRAM cell, and the reason it is no longer used, is that it requires two pairs of column and rowlines and a large layout area. Modern 1-transistor, 1-capacitor DRAM cells use a single rowline, a single columnline, and considerably less area.

1.1.2 The 4k–64 Meg DRAM (Second Generation)

We distinguish second-generation DRAMs from first-generation DRAMs by the introduction of multiplexed address inputs, multiple memory arrays, and the 1-transistor/1-capacitor memory cell. Furthermore, second-generation DRAMs offer more modes of operation for greater flexibility or higher speed operation. Examples are *page mode, nibble mode, static column mode, fast page mode* (FPM), and *extended data out* (EDO). Second-generation DRAMs range in size from 4k (4,096 x 1 bit, i.e., 4,096 address locations with 1-bit input/output word size) up to 64 Meg (67,108,864 bits) in memory sizes of 16 Meg x 4 organized as 16,777,216 address locations with 4-bit input/output word size, 8 Meg x 8, or 4 Meg x 16.

Two other major changes occurred in second-generation DRAMs: (1) the power supply transitioned to a single 5 V and (2) the technology advanced from NMOS to CMOS. The change to a single 5 V supply occurred at the 64kbit density. It simplified system design to a single power supply for the memory, processor, and any TTL (transistor-transistor logic) used in the system. As a result, rowlines had to be driven to a voltage

greater than 5 V to turn the NMOS access devices fully ON (more on this later), and the substrate held at a potential less than zero. For voltages outside the supply range, charge pumps are used (see Chapter 6). The move from NMOS to CMOS, at the 1Mb density level, occurred because of concerns over speed, power, and layout size. At the cost of process complexity, complementary devices improved the design.

1.1.2.1 Multiplexed Addressing. Figure 1.9 shows a 4k DRAM block diagram, while Figure 1.10 shows the pin connections for a 4k chip. Note that compared to the block diagram of the 1k DRAM shown in Figure 1.1, the number of address input pins has decreased from 10 to 6, even though the memory size has quadrupled. This is the result of using multiplexed addressing in which the same address input pins are used for both the row and column addresses. The row address strobe *(RAS*)* input clocks the address present on the DRAM address pins A_0 to A_5 into the row address latches on the falling edge. The column address strobe *(CAS*)* input clocks the input address into the column address latches on its falling edge.

Figure 1.11 shows the timing relationships between *RAS**, *CAS**, and the address inputs. Note that t_{RC} is still (as indicated in the last section) the random cycle time for the DRAM, indicating the maximum rate we can write to or read from a DRAM. Note too how the row (or column) address must be present on the address inputs when *RAS** (or *CAS*)* goes LOW. The parameters t_{RAS} and t_{CAS} indicate how long *RAS** or *CAS** must remain LOW after clocking in a column or row address. The parameters t_{ASR}, t_{RAH}, t_{ASC}, and t_{CAH} indicate the setup and hold times for the row and column addresses, respectively.

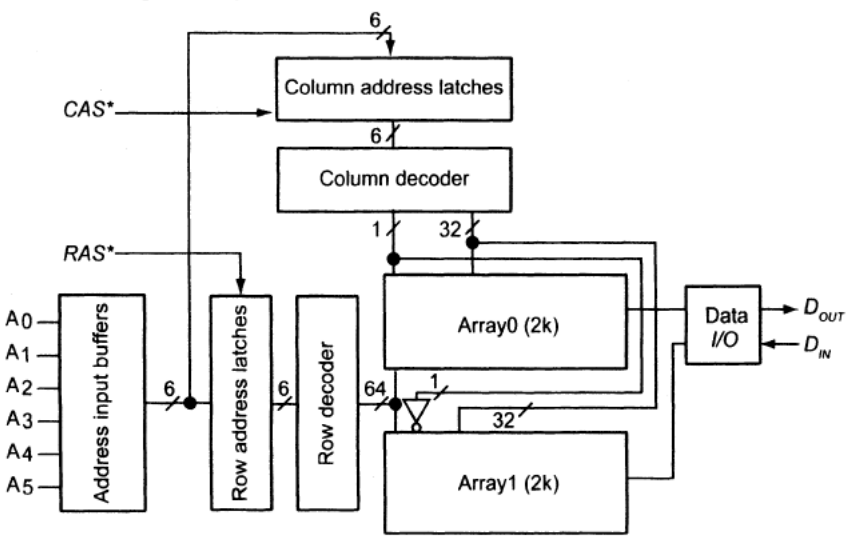


Figure 1.9 Block diagram of a 4k DRAM.

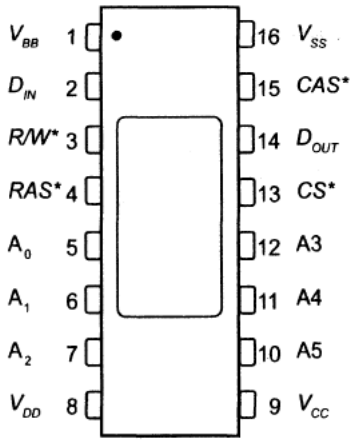


Figure 1.10 4,096-bit DRAM pin connections.

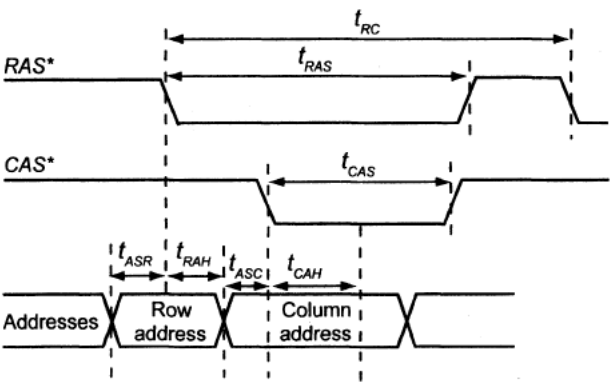


Figure 1.11 Address timing.

1.1.2.2 Multiple Memory Arrays. As mentioned earlier, second-generation DRAMs began to use multiple or segmented memory arrays. The main reason for splitting up the memory into more than one array at the cost of a larger layout area can be understood by considering the parasitics present in the dynamic memory circuit element. To understand the origins of these parasitics, consider the modern DRAM memory cell comprising one MOSFET and one capacitor, as shown in Figure 1.12.

In the next section, we cover the operation of this cell in detail. Here we introduce the operation of the cell. Data is written to the cell by driving the rowline (a.k.a., wordline) HIGH, turning ON the MOSFET, and allowing the columnline (a.k.a., digitline or bitline) to charge or discharge the storage capacitor. After looking at this circuit for a moment, we can make the following observation.

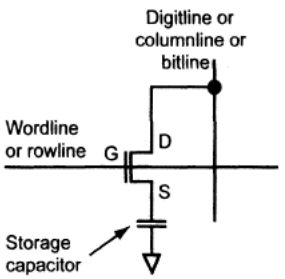


Figure 1.12 1-transistor, 1-capacitor (1T1C) memory cell.

1. The wordline (rowline) may be fabricated using polysilicon (poly). This allows the MOSFET to be formed by crossing the poly wordline over an n+ active area.
2. To write a full V_{CC} logic voltage (where V_{CC} is the maximum positive power supply voltage) to the storage capacitor, the rowline must be driven to a voltage greater than V_{CC} + the n-channel MOSFET threshold voltage (with body effect). This voltage, $> V_{CC} + V_{TH}$, is often labeled V_{CC} *pumped* (V_{CCP}).
3. The bitline (columnline) may be made using metal or polysilicon. The main concern, as we'll show in a moment, is to reduce the parasitic capacitance associated with the bitline.

Consider the row of N dynamic memory elements shown in Figure 1.13. Typically, in a modern DRAM, N is 512, which is also the number of bitlines. When a row address is strobed into the DRAM, via the address input pins using the falling edge of *RAS**, the address is decoded to drive a wordline (rowline) to V_{CCP}. *This turns ON an entire row in a DRAM memory array.* Turning ON an entire row in a DRAM memory array allows the information stored on the capacitors to be sensed (for a Read) via the bitlines or allows the charging or discharging, via the bitlines, of the storage capacitors (for a Write). Opening a row of data by driving a wordline HIGH is a ***very important*** concept for understanding the modes of DRAM operation. For Refresh, we only need to supply row addresses during a Refresh operation. For page Reads—when a row is open—a large amount of data, which is set by the number of columns in the DRAM array, can be accessed by simply changing the column address.

We're now in a position to answer the question: "Why are we limited to increasing the number of columnlines (or bitlines) used in a memory array?" or "Why do we need to break up the memory into smaller memory arrays?" The answer to these questions comes from the realization that the

more bitlines we use in an array, the longer the delay through the wordline (Figure 1.13).

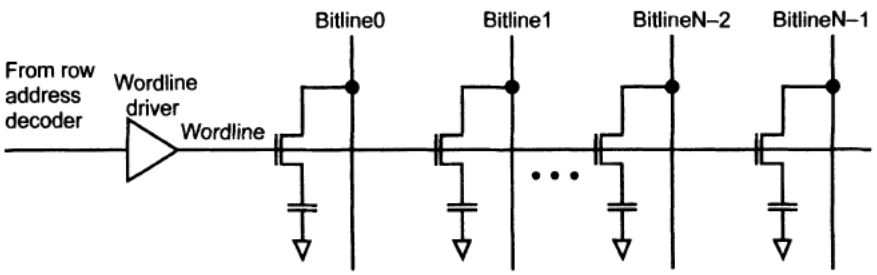


Figure 1.13 Row of N dynamic memory elements.

If we drive the wordline on the left side of Figure 1.13 HIGH, the signal will take a finite time to reach the end of the wordline (the wordline on the right side of Figure 1.13). This is due to the distributed resistance/capacitance structure formed by the resistance of the polysilicon wordline and the capacitance of the MOSFET gates. The delay limits the speed of DRAM operation. To be precise, it limits how quickly a row can be opened and closed. To reduce this *RC* time, a polycide wordline is formed by adding a silicide, for example, a mixture of a refractory metal such as tungsten with polysilicon, on top of polysilicon. Using a polycide wordline will have the effect of reducing the wordline resistance. Also, additional drivers can be placed at different locations along the wordline, or the wordline can be *stitched* at various locations with metal.

The limitations on the additional number of wordlines can be understood by realizing that by adding more wordlines to the array, more parasitic capacitance is added to the bitlines. This parasitic capacitance becomes important when sensing the value of data charge stored in the memory element. We'll discuss this in more detail in the next section.

1.1.2.3 Memory Array Size. A comment is in order about memory array size and how addressing can be used for setting word and page size. (We'll explain what this means in a moment.) If we review the block diagram of the 4k DRAM shown in Figure 1.9, we see that two 2k-DRAM memory arrays are used. Each 2k memory is composed of 64 wordlines and 32 bitlines for 2,048 memory elements/address locations per array. In the block diagram, notice that a single bit, coming from the column decoder, can be used to select data, via the bitlines, from Array0 or Array1.

From our discussion earlier, we can open a row in Array0 while at the same time opening a row in Array1 by simply applying a row address to the input address pins and driving *RAS** LOW. Once the rows are open, it is a simple matter of changing the column address to select different data asso-

ciated with the same open row from either array. If our word size is 1 bit, we could define a page as being 64 bits in length (32 bits from each array). We could also define our page size as 32 bits with a 2-bit word for input/output. We would then say that the DRAM is a 4k DRAM organized as 2k x 2. Of course, in the 4k DRAM, in which the number of bits is small, the concepts of page reads or size aren't too useful. We present them here simply to illustrate the concepts. Let's consider a more practical and modern configuration.

Suppose we have a 64-Meg DRAM organized as 16 Meg x 4 (4 bits input/output) using 4k row address locations and 4k column address locations (12 bits or pins are needed for each 4k of addressing). If our (sub) memory array size is 256kbits, then we have a total of 256 memory arrays on our DRAM chip. We'll assume that there are 512 wordlines and 512 bitlines (digitlines), so that the memory array is logically square. (However, physically, as we shall see, the array is not square.) Internal to the chip, in the address decoders, we can divide the row and column addresses into two parts: the lower 9 bits for addressing the wordlines/bitlines in a 256k memory array and the upper 3 bits for addressing one of the 64 "group-of-four" memory arrays (6 bits total coming from the upper 3 bits of the row and column addresses).

Our 4-bit word comes from the group-of-four memory arrays (one bit from each memory array). We can define a page of data in the DRAM by realizing that when we open a row in each of the four memory arrays, we are accessing 2k of data (512 bits/array x 4 arrays). By simply changing the column address without changing the row address and thus opening another group-of-four wordlines, we can access the 2k "page" of data. With a little imagination, we can see different possibilities for the addressing. For example, we could open 8 group-of-four memory arrays with a row address and thus increase the page size to 16k, or we could use more than one bit at a time from an array to increase word size.

1.1.2.4 Refreshing the DRAM. Refreshing the DRAM is accomplished by sequentially opening each row in the DRAM. (We'll discuss how the DRAM cell is refreshed in greater detail later in the book.) If we use the 64-Meg example in the last section, we need to supply 4k row addresses to the DRAM by changing the external address inputs from 000000000000 to 111111111111 while clocking the addresses into the DRAM using the falling edge of *RAS**. In some DRAMs, an internal row address counter is present to make the DRAM easier to refresh. The general specification for 64-Meg DRAM Refresh is that all rows must be refreshed at least every 64 ms, which is an average of 15.7 μs per row. This means, that if the Read cycle time t_{RC} is 100 ns (see Figure 1.11), it will take 4,096 • 100 ns or

410 μs to refresh a DRAM with 4k of row addresses. The percentage of time the DRAM is unavailable due to Refresh can be calculated as 410 μs/64 ms or 0.64% of the time. Note that the Refresh can be a burst, taking 410 μs as just described, or distributed, where a row is refreshed every 15.7 μs.

1.1.2.5 Modes of Operation. From the last section, we know that we can open a row in one or more DRAM arrays concurrently, allowing a page of data to be written to or read from the DRAM. In this section, we look at the different modes of operation possible for accessing this data via the column address decoder. Our goal in this section is not to present all possible modes of DRAM operation but rather to discuss the modes that have been used in second-generation DRAMs. These modes are page mode, nibble mode, static column mode, fast page mode, and extended data out.

Figure 1.14 shows the timing diagram for a age mode Read, Write, and Read-Modify-Write. We can understand this timing diagram by first noticing that when *RAS** goes LOW, we clock in a row address, decode the row address, and then drive a wordline in one or more memory arrays to V_{CCP}. The result is an open row(s) of data sitting on the digitlines columnlines). *Only one row can be opened in any single array at a time.* Prior to opening a row, the bitlines are precharged to a known voltage. (Precharging to $V_{CC}/2$ is typically performed using internal circuitry.) Also notice at this time that data out, D_{OUT}, is in a Hi-Z state; that is, the DRAM is not driving the bus line connected to the D_{OUT} pin.

The next significant timing event occurs when *CAS** goes LOW and the column address is clocked into the DRAM (Figure 1.14). At this time, the column address is decoded, and, assuming that the data from the open row is sitting on the digitlines, it is steered using the column address decoder to D_{OUT}. We may have an open row of 512 bits, but we are steering only one bit to D_{OUT}. Notice that when *CAS** goes HIGH, D_{OUT} goes back to the Hi-Z state.

By strobing in another column address with the same open row, we can select another bit of data (again via the column address decoder) to steer to the D_{OUT} pin. In this case, however, we have changed the DRAM to the Write mode (Figure 1.14). This allows us to write, with the same row open via the D_{IN} pin in Figure 1.10, to any column address on the open row. *Later, second-generation DRAMs used the same pins for both data input and output to reduce pin count. These bidirectional pins are labeled DQ.*

The final set of timing signals in Figure 1.14 (the right side) read data out of the DRAM with *R/W** HIGH, change *R/W** to a LOW, and then write to the same location. Again, when *CAS** goes HIGH, D_{OUT} goes back to the Hi-Z state.

The remaining modes of operation are simple modifications of page mode. As seen in Figure 1.15, FPM allows the column address to propagate into the column circuits while *CAS** is HIGH. The speed of the DRAM thus improves by reducing the delay between *CAS** going LOW and valid data present, or accessed, on D_{OUT} (t_{CAC}). EDO is simply an FPM DRAM that doesn't force D_{OUT} to a Hi-Z state immediately when *CAS** goes HIGH. The data out of the DRAM is thus available for a longer period of time, allowing for faster system operation. In general, opening the row is the operation that takes the longest amount of time. Once a row is open, the data sitting on the columnlines can be steered to D_{OUT} at a fast rate. Interestingly, using column access modes has been the primary method of boosting DRAM performance over the years, especially in double-data rate (DDR) DRAMs (see Chapter 8).

The other popular modes of operation in second-generation DRAMs were the static column and nibble modes. Static column mode DRAMs used flow-through latches in the column address path. When a column address was changed externally, with *CAS** LOW, the column address fed directly to the column address decoder. (The address wasn't clocked on the falling edge of *CAS**.) This increased the speed of the DRAM by preventing the outputs from going into the Hi-Z state with changes in the column address.

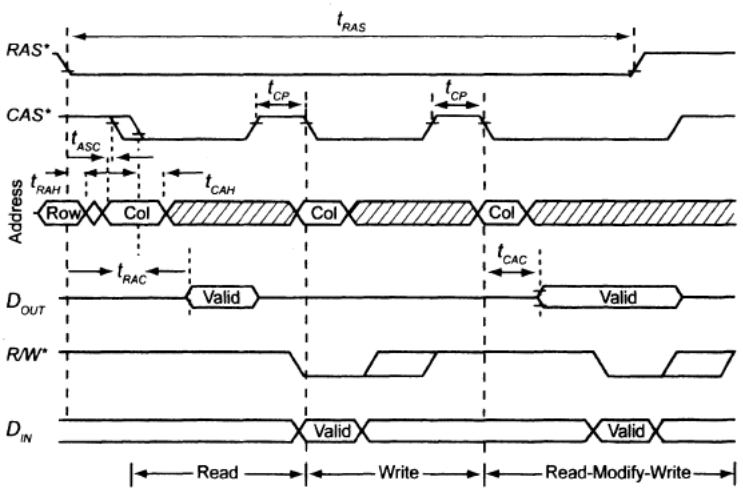


Figure 1.14 Page mode.

Nibble mode DRAMs used an internal presettable address counter so that by strobing *CAS**, the column address would change internally. Figure 1.16 illustrates the timing operation for a nibble mode DRAM. The first time *CAS** transitions LOW (first being defined as the first transition after *RAS** goes LOW), the column address is loaded into the counter. If *RAS** is

held LOW and *CAS** is toggled, the internal address counter is incremented, and the sequential data appears on the output of the DRAM. The term *nibble mode* comes from limiting the number of *CAS** cycles to four (a nibble).

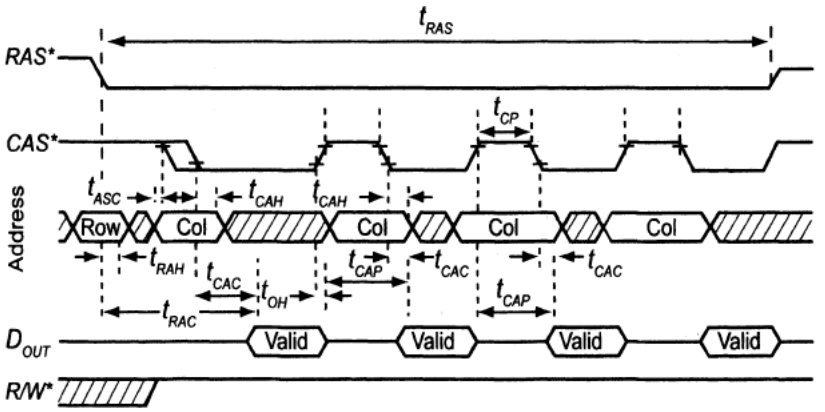


Figure 1.15 Fast page mode.

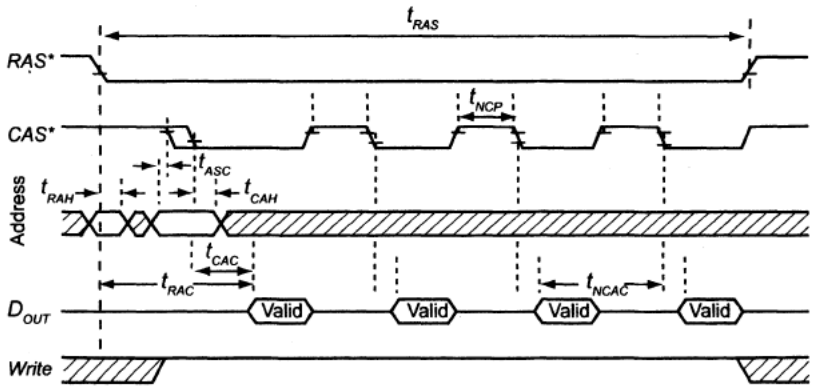


Figure 1.16 Nibble mode.

1.1.3 Synchronous DRAM (Third Generation)

Synchronous DRAMs (SDRAMs) are made by adding a synchronous interface between the basic core DRAM operation/circuitry of second-generation DRAMs and the control coming from off-chip to make the DRAM operation faster. All commands and operations to and from the DRAM are executed on the rising edge of a master or command clock signal that is common to all SDRAMs and labeled *CLK*. See Figure 1.17 for the pin connections of a 64Mb SDRAM with 16-bit input/output (*I/O*).

SDRAMs operate with a maximum *CLK* frequency in the range of 100–167 MHz, clocking data on one edge of the clock. This means that if a 64Mb SDRAM is organized as a x16 part (that is, the input/output word size is 16 bits), the maximum rate at which the words can be written to the part is 200–334 MB/s.

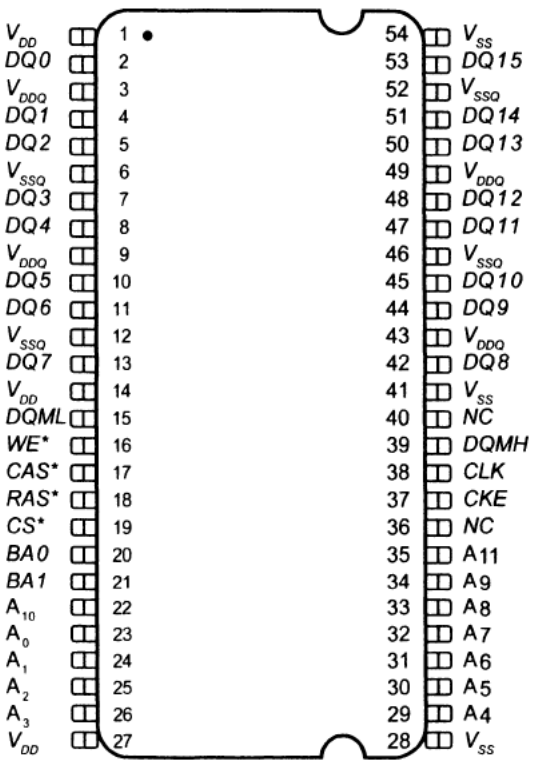


Figure 1.17 Pin connections of a 64Mb SDRAM with 16-bit *I/O*.

A variation of the SDRAM is the *double-data-rate SDRAM* (DDR SDRAM, or simply DDR DRAM). The DDR parts register commands and operations on the rising edge of the clock signal while allowing data to be transferred on both the rising and falling edges. A differential input clock signal is used in the DDR DRAM with the labeling of, not surprisingly, *CLK* and *CLK**. In addition, the DDR DRAM provides an output data strobe, labeled DQS, synchronized with the output data and the input *CLK*. DQS is used at the controller to strobe in data from a DRAM. The big benefit of using a DDR part is that the data transfer rate can be twice the clock frequency because data can be transferred on both the rising and falling edges of *CLK*. This means that when using a 133 MHz clock, the data written to and read from the DRAM can be transferred at 266M words/s. Using

the numbers from the previous paragraph, this means that a 64Mb DDR SDRAM with an input/output word size of 16 bits will transfer data to and from the memory controller at 400–572 MB/s.

Figure 1.18 shows the block diagram of a 64Mb SDRAM with 16-bit *I/O*. Note that although *CLK* is now used for transferring data, we still have the second-generation control signals *CS**, *WE**, *CAS**, and, *RAS** present on the part. (*CKE* is a clock enable signal which, unless otherwise indicated, is assumed HIGH.) Let's discuss how these control signals are used in an SDRAM by recalling that in a second-generation DRAM, a Write was executed by first driving *WE** and *CS** LOW. Next a row was opened by applying a row address to the part and then driving *RAS** LOW. (The row address is latched on the falling edge of *RAS**.) Finally, a column address was applied and latched on the falling edge of *CAS**. A short time later, the data applied to the part would be written to the accessed memory location.

For the SDRAM Write, we change the syntax of the descriptions of what's happening in the part. However, the fundamental operation of the DRAM circuitry is the same as that of the second-generation DRAMs. We can list these syntax changes as follows:

1. The memory is segmented into banks. For the 64Mb memory of Figure 1.17 and Figure 1.18, each bank has a size of 16Mbs (organized as 4,096 row addresses [12 bits] x 256 column addresses [8 bits] x 16 bits [16 *DQ I/O* pins]). As discussed earlier, this is nothing more than a simple logic design of the address decoder (the banks can be laid out so that they are physically in the same area). The bank selected is determined by the addresses *BA*0 and *BA*1.
2. In second-generation DRAMs, we said, "We open a row," as discussed earlier. In SDRAM, we now say, "We activate a row in a bank." We do this by issuing an active command to the part. Issuing an active command is accomplished on the rising edge of *CLK* with a row/bank address applied to the part with *CS** and *RAS** LOW, while *CAS** and *WE** are held HIGH.
3. In second-generation DRAMs, we said, "We write to a location given by a column address," by driving *CAS** LOW with the column address applied to the part and then applying data to the part. In an SDRAM, we write to the part by issuing the Write command to the part. Issuing a Write command is accomplished on the rising edge of *CLK* with a column/bank address applied to the part: *CS**, *CAS**, and *WE** are held LOW, and *RAS** is held HIGH.

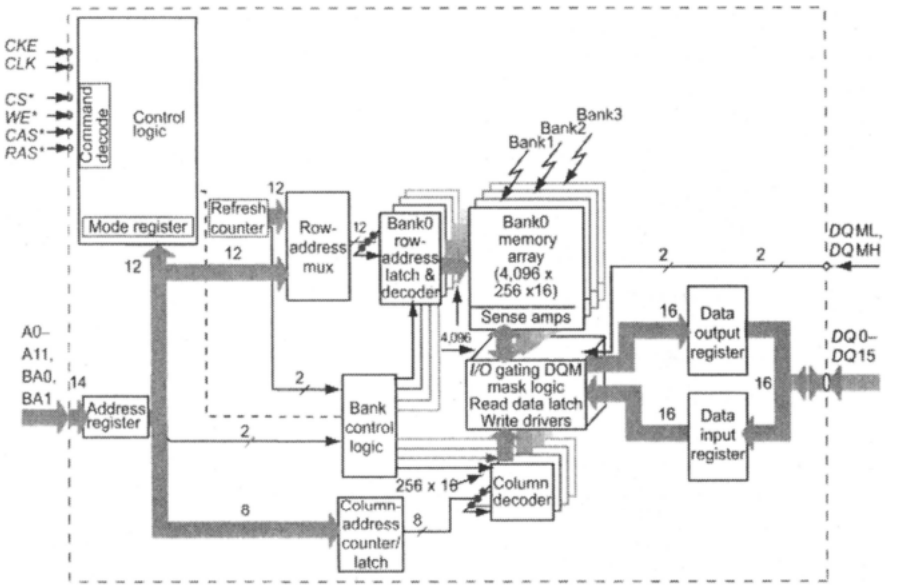


Figure 1.18 Block diagram of a 64Mb SDRAM with 16-bit *I/O*.

Table 1.1 shows the commands used in an SDRAM. In addition, this table shows how inputs/outputs *(DQs)* can be masked using the *DQ* mask (DQM) inputs. This feature is useful when the DRAM is used in graphics applications.

SDRAMs often employ pipelining in the address and data paths to increase operating speed. Pipelining is an effective tool in SDRAM design because it helps disconnect operating frequency and access latency. Without pipelining, a DRAM can only process one access instruction at a time. Essentially, the address is held valid internally until data is fetched from the array and presented to the output buffers. This single instruction mode of operation ties operating frequency and access time (or latency) together. However, with pipelining, additional access instructions can be fed into the SDRAM before prior access instructions have completed, which permits access instructions to be entered at a higher rate than would otherwise be allowed. Hence, pipelining increases operating speed.

Table 1.1 SDRAM commands. (Notes: 1)

Name	*CS**	*RAS**	*CAS**	*WE**	DQM	ADDR	*DQ*s	Notes
Command inhibit (NOP)	H	X	X	X	X	X	X	—
No operation (NOP)	L	H	H	H	X	X	X	—
Active (select bank and activate row)	L	L	H	H	X	Bank/ row	X	3
Read (select bank and column, and start Read burst)	L	H	L	H	L/H[8]	Bank/ col	X	4
Write (select bank and column, and start Write burst)	L	H	L	L	L/H[8]	Bank/ col	Valid	4
Burst terminate	L	H	H	L	X	X	Active	—
PRECHARGE (deactive row in bank or banks)	L	L	H	L	X	Code	X	5
Auto-Refresh ors self-refresh (enter self-refresh mode)	L	L	L	H	X	X	X	6, 7
Load mode register	L	L	L	L	X	Op-code	X	2
Write ENABLE/ output ENABLE	—	—	—	—	L	—	Active	8
Write inhibit/ output Hi-Z	—	—	—	—	H	—	Hi-Z	8

Notes

1. *CKE* is HIGH for all commands shown except for self-refresh.
2. *A*0–*A*11 define the op-code written to the mode register.
3. *A*0–*A*11 provide row address, and *BA*0, *BA*1 determine which bank is made active.
4. *A*0–*A*9 (x4), *A*0–*A*8 (x8), or *A*0–*A*7 (x16) provide column address; *A*10 HIGH enables the auto *PRECHARGE* feature (nonpersistent), while *A*10 LOW disables the auto *PRECHARGE* feature; *BA*0, *BA*1 determine which bank is being read from or written to.
5. *A*10 LOW: *BA*0, *BA*1 determine the bank being precharged. *A*10 HIGH: all banks precharged and *BA*0, *BA*1 are "don't care."
6. This command is Auto-Refresh if *CKE* is HIGH and Self-Refresh if *CKE* is LOW.

7. Internal Refresh counter controls row addressing; all inputs and *I/O*s are "don't care" except for *CKE*.
8. Activates or deactivates the *DQ*s during Writes (zero-clock delay) and Reads (two-clock delay).

Pipeline stages in the data path can also be helpful when synchronizing output data to the system clock. *CAS* latency refers to a parameter used by the SDRAM to synchronize the output data from a Read request with a particular edge of the system clock. A typical Read for an SDRAM with *CAS* latency set to three is shown in Figure 1.19. SDRAMs must be capable of reliably functioning over a range of operating frequencies while maintaining a specified *CAS* latency. This is often accomplished by configuring the pipeline stage to register the output data to a specific clock edge, as determined by the *CAS* latency parameter.

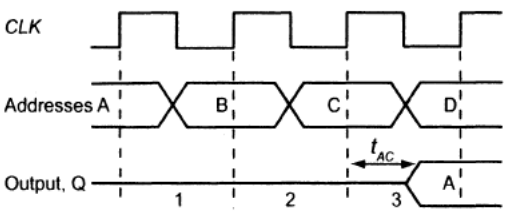


Figure 1.19 SDRAM with a latency of three.

At this point, we should understand the basics of SDRAM operation, but we may be asking, "Why are SDRAMs potentially faster than second-generation DRAMs such as EDO or FPM?" The answer to this question comes from the realization that it's possible to activate a row in one bank and then, while the row is opening, perform an operation in some other bank (such as reading or writing). In addition, one of the banks can be in a *PRECHARGE* mode (the bitlines are driven to $V_{CC}/2$) while accessing one of the other banks and, thus, in effect hiding *PRECHARGE* and allowing data to be continuously written to or read from the SDRAM. (Of course, this depends on which application and memory address locations are used.) We use a mode register, as shown in Figure 1.20, to put the SDRAM into specific modes of operation for programmable operation, including pipelining and burst Reads/Writes of data [2].

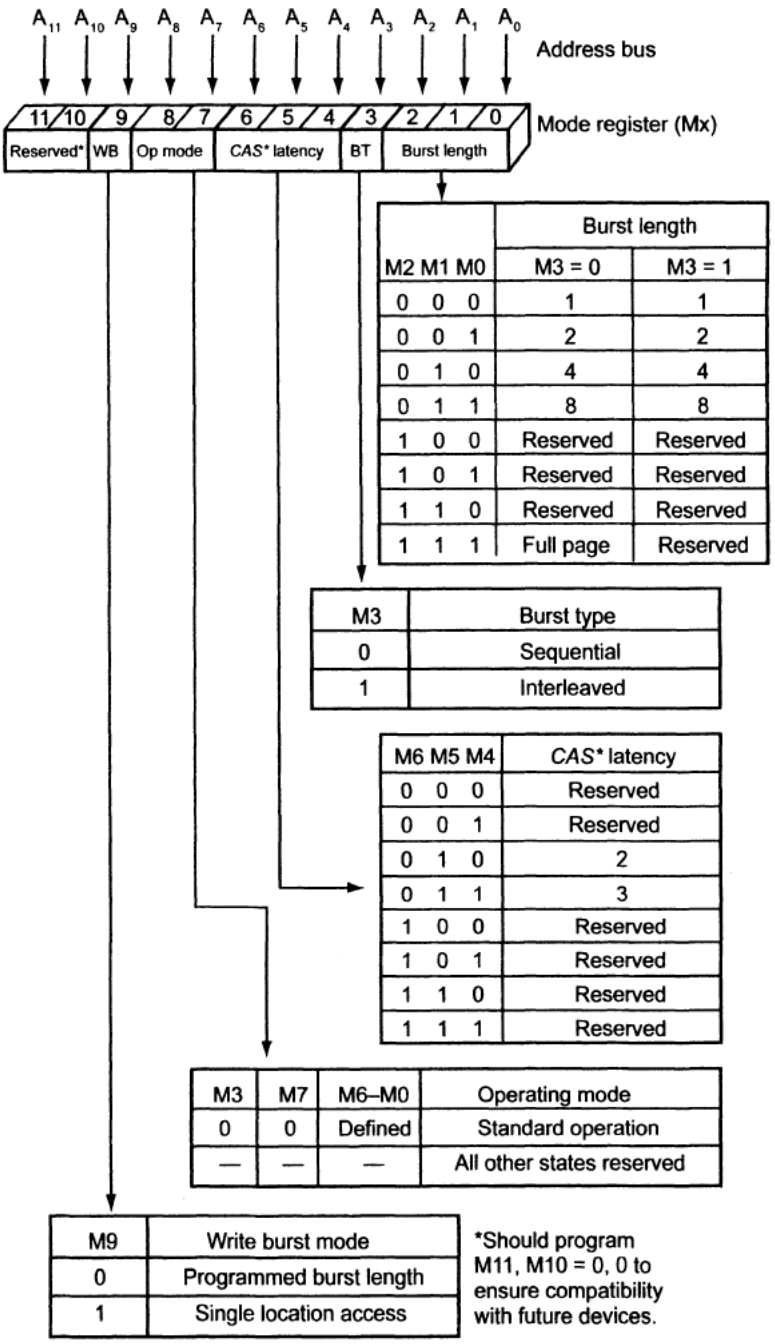


Figure 1.20 Mode register.

1.2 DRAM BASICS

A modern DRAM memory cell or memory bit (mbit), as shown in Figure 1.21, is formed with one transistor and one capacitor, accordingly referred to as a 1T1C cell. The mbit is capable of holding binary information in the form of stored charge on the capacitor. The mbit transistor operates as a switch interposed between the mbit capacitor and the digitline. Assume that the capacitor's common node is biased at $V_{CC}/2$, which we will later show as a reasonable assumption. Storing a logic one in the cell requires a capacitor with a voltage of $+V_{CC}/2$ across it. Therefore, the charge stored in the mbit capacitor is

$$Q = \frac{V_{CC}}{2} \cdot C \tag{1.1}$$

where C is the capacitance value in farads. Conversely, storing a logic zero in the cell requires a capacitor with a voltage of $-V_{CC}/2$ across it. Note that the stored charge on the mbit capacitor for a logic zero is

$$Q = \frac{-V_{CC}}{2} \cdot C \tag{1.2}$$

The charge is negative with respect to the $V_{CC}/2$ common node voltage in this state. Various leakage paths cause the stored capacitor charge to slowly deplete. To return the stored charge and thereby maintain the stored data state, the cell must be refreshed. The required refreshing operation is what makes DRAM memory dynamic rather than static.

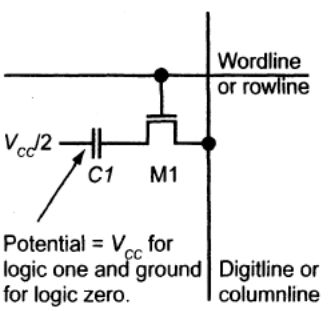


Figure 1.21 1T1C DRAM memory cell.
(Note the rotation of the rowline and columnline.)

The digitline referred to earlier consists of a conductive line connected to a multitude of mbit transistors. The conductive line is generally con-

structed from either metal or silicide/polycide polysilicon. Because of the quantity of mbits connected to the digitline and its physical length and proximity to other features, the digitline is highly capacitive. For instance, a typical value for digitline capacitance on a 50 nm process might be 120 fF. Digitline capacitance is an important parameter because it dictates many other aspects of the design. We discuss this further in Section 2.1. For now, we continue describing basic DRAM operation.

The mbit transistor gate terminal is connected to a wordline rowline). The wordline, which is connected to a multitude of mbits, is actually formed of the same polysilicon as that of the transistor gate. The wordline is physically orthogonal to the digitline. A memory array is formed by tiling a selected quantity of mbits together such that mbits along a given digitline do not share a common wordline and mbits along a common wordline do not share a common digitline. Examples of this are shown in Figures 1.22 and 1.23. In these layouts, mbits are paired to share a common contact to the digitline, which reduces the array size by eliminating duplication.

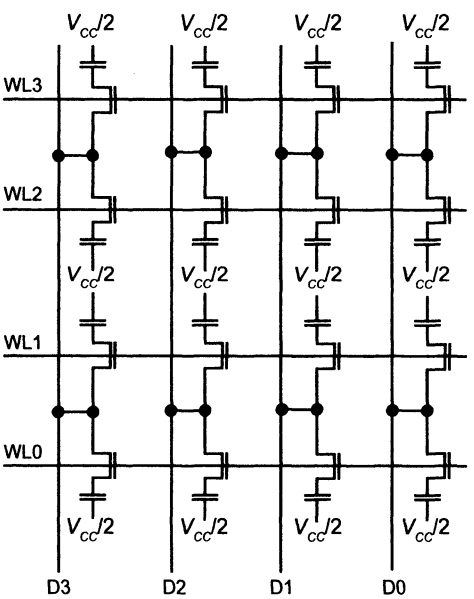


Figure 1.22 Open digitline memory array schematic.

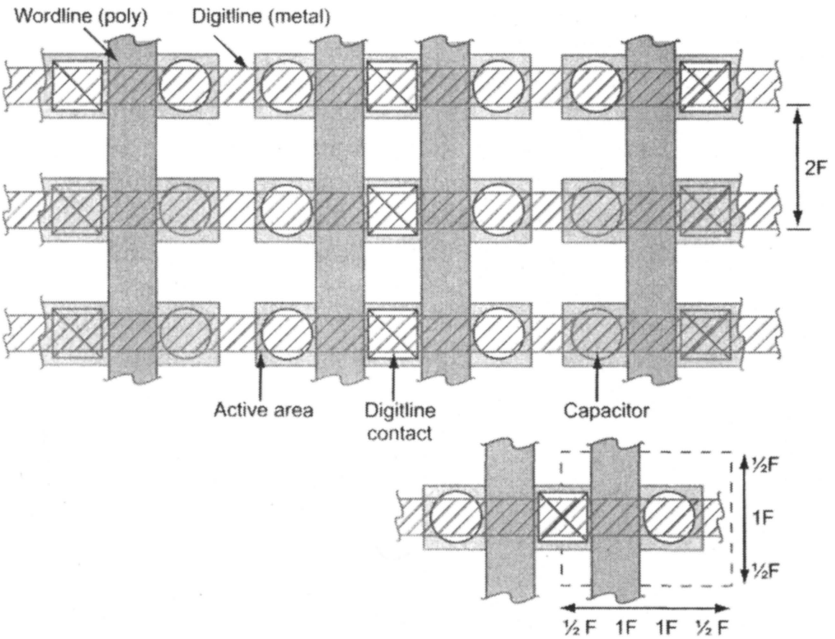


Figure 1.23 Open digitline memory array layout.

1.2.1 Access and Sense Operations

Next, we examine the access and sense operations. We begin by assuming that the cells connected to D1, in Figure 1.24, have logic one levels ($+V_{CC}/2$) stored on them and that the cells connected to D0 have logic zero levels ($-V_{CC}/2$) stored on them. Next, we form a digitline pair by considering two digitlines from adjacent arrays. The digitline pairs, labeled D0/D0* and D1/D1*, are initially equilibrated to $V_{CC}/2$ V. All wordlines are initially at 0 V, ensuring that the mbit transistors are OFF. Prior to a wordline firing, the digitlines are electrically disconnected from the $V_{CC}/2$ bias voltage and allowed to float. They remain at the $V_{CC}/2$ *PRECHARGE* voltage due to their capacitance.

To read mbit1, wordline WL0 changes to a voltage that is at least one transistor V_{TH} above V_{CC}. This voltage level is referred to as V_{CCP} or V_{PP}. To ensure that a full logic one value can be written back into the mbit capacitor, V_{CCP} must remain greater than one V_{TH} above V_{CC}. The mbit capacitor begins to discharge onto the digitline at two different voltage levels depending on the logic level stored in the cell. For a logic one, the capacitor begins to discharge when the wordline voltage exceeds the digitline *PRECHARGE* voltage by V_{TH}. For a logic zero, the capacitor begins to discharge when the wordline voltage exceeds V_{TH}. Because of the finite rise time of the word-

line voltage, this difference in turn-on voltage translates into a significant delay when reading ones, as seen in Figure 1.25.

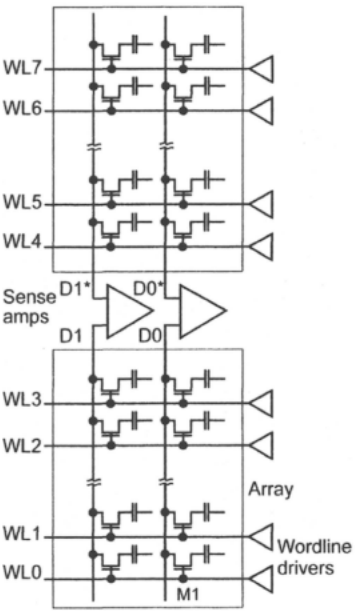


Figure 1.24 Simple array schematic (an open DRAM array).

Accessing a DRAM cell results in charge sharing between the mbit capacitor and the digitline capacitance. This charge sharing causes the digitline voltage either to increase for a stored logic one or to decrease for a stored logic zero. Ideally, only the digitline connected to the accessed mbit will change. In reality, the other digitline voltage also changes slightly, due to parasitic coupling between digitlines and between the firing wordline and the other digitline. (This is especially true for the folded bitline architecture discussed later.) Nevertheless, a differential voltage develops between the two digitlines. The magnitude of this voltage difference, or signal, is a function of the *mbit capacitance* (C_{mbit}), *digitline capacitance* (C_{digit}), and voltage stored on the cell prior to access (V_{cell}). See Figure 1.26. Accordingly,

$$V_{signal} = V_{cell} \cdot \frac{C_{mbit}}{C_{digit} + C_{mbit}} \tag{1.3}$$

A V_{signal} of 200 mV is yielded from a design in which $V_{cell} = 1.00$, $C_{mbit} = 20$ fF, and $C_{digit} = 200$ fF.

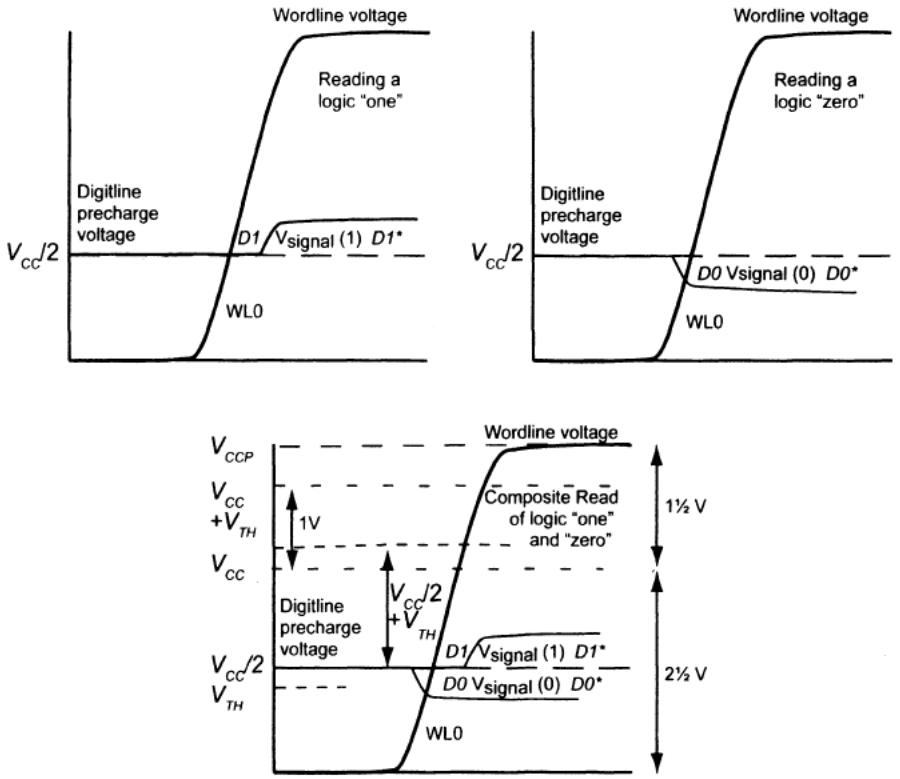


Figure 1.25 Cell access waveforms.

After the cell has been accessed, sensing occurs. Sensing is essentially the amplification of the digitline signal or the differential voltage between the digitlines. Sensing is necessary to properly read the cell data and refresh the mbit cells. (The reason for forming a digitline pair now becomes apparent.) presents a schematic diagram for a simplified sense amplifier circuit: a cross-coupled NMOS pair and a cross-coupled PMOS pair. The sense amplifiers also appear like a pair of cross-coupled inverters in which *ACT* and *NLAT** provide power and ground. The NMOS pair or Nsense-amp has a common node labeled *NLAT** (for *Nsense-amp latch*).

Similarly, the Psense-amp has a common node labeled *ACT* (for *Active pull-up*). Initially, *NLAT** is biased to $V_{CC}/2$, and *ACT* is biased to V_{SS} or signal ground. Because the digitline pair D1 and D1* are both initially at $V_{CC}/2$, the Nsense-amp transistors are both OFF. Similarly, both Psense-amp transistors are OFF. Again, when the mbit is accessed, a signal develops across the digitline pair. While one digitline contains charge from the cell access, the other digitline does not but serves as a reference for the Sensing operation. The sense amplifiers are generally fired sequentially: the Nsense-amp first, then the Psense-amp. Although designs vary at this point,

the higher drive of NMOS transistors and better V_{TH} matching offer better sensing characteristics by Nsense-amps and thus lower error probability compared to Psense-amps.

Waveforms for the Sensing operation are shown in Figure 1.28. The Nsense-amp is fired by bringing *NLAT** (Nsense-amp latch) toward ground. As the voltage difference between *NLAT** and the digitlines (D1 and D1* in) approaches V_{TH}, the NMOS transistor, whose gate is connected to the higher voltage digitline, begins to conduct. This conduction occurs first in the subthreshold and then in the saturation region as the gate-to-source voltage exceeds V_{TH} and causes the low-voltage digitline to discharge toward the *NLAT** voltage. Ultimately, *NLAT** will reach ground and the digitline will be brought to ground potential. Note that the other NMOS transistor will not conduct: its gate voltage is derived from the low-voltage digitline, which is being discharged toward ground. In reality, parasitic coupling between digitlines and limited subthreshold conduction by the second transistor result in a temporary voltage drop on the high digitline, as seen in Figure 1.28.

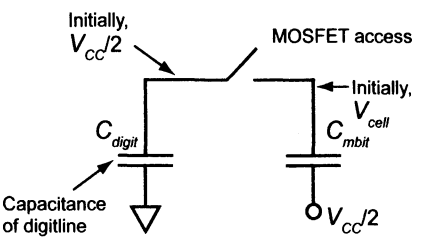


Figure 1.26 DRAM charge sharing..

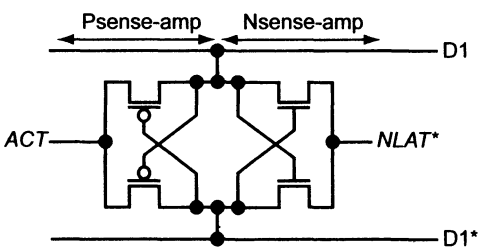


Figure 1.27 Sense amplifier schematic.

Sometime after the Nsense-amp fires, *ACT* will be brought toward V_{CC} to activate the Psense-amp, which operates in a complementary fashion to the Nsense-amp. With the low-voltage digitline approaching ground, there is a strong signal to drive the appropriate PMOS transistor into conduction. This conduction, again moving from subthreshold to saturation, charges the

high-voltage digitline toward *ACT*, ultimately reaching V_{CC}. Because the mbit transistor remains ON, the mbit capacitor is refreshed during the Sensing operation. The voltage, and hence charge, which the mbit capacitor held prior to accessing, is restored to a full level: V_{CC} for a logic one and ground for a logic zero. It should be apparent now why the minimum wordline voltage is a V_{TH} above V_{CC}. If V_{CCP} were anything less, a full V_{CC} level could not be written back into the mbit capacitor. The mbit transistor source voltage V_{source} cannot be greater than $V_{gate} - V_{TH}$ because this would turn OFF the transistor.

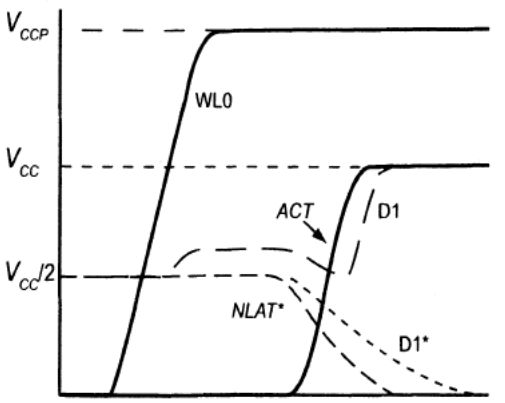


Figure 1.28 Sensing operation waveforms.

1.2.2 Write Operation

A Write operation is similar to a Sensing and Restore operation except that a separate Write driver circuit determines the data that is placed into the cell. The Write driver circuit is generally a tristate inverter connected to the digitlines through a second pair of pass transistors, as shown in Figure 1.29. These pass transistors are referred to as *I/O* transistors. The gate terminals of the *I/O* transistors are connected to a common *column select* (*CSEL*) signal. The *CSEL* signal is decoded from the column address to select which pair (or multiple pairs) of digitlines is routed to the output pad or, in this case, the Write driver.

In most current DRAM designs, the Write driver simply overdrives the sense amplifiers, which remain ON during the Write operation. After the new data is written into the sense amplifiers, the amplifiers finish the Write cycle by restoring the digitlines to full rail-to-rail voltages. An example is shown in Figure 1.30 in which D1 is initially HIGH after the Sensing operation and LOW after the writing operation. A Write operation usually involves only 2–4 mbits within an array of mbits because a single *CSEL* line is generally connected to only four pairs of *I/O* transistors. The remain-

ing digitlines are accessed through additional *CSEL* lines that correspond to different column address locations.

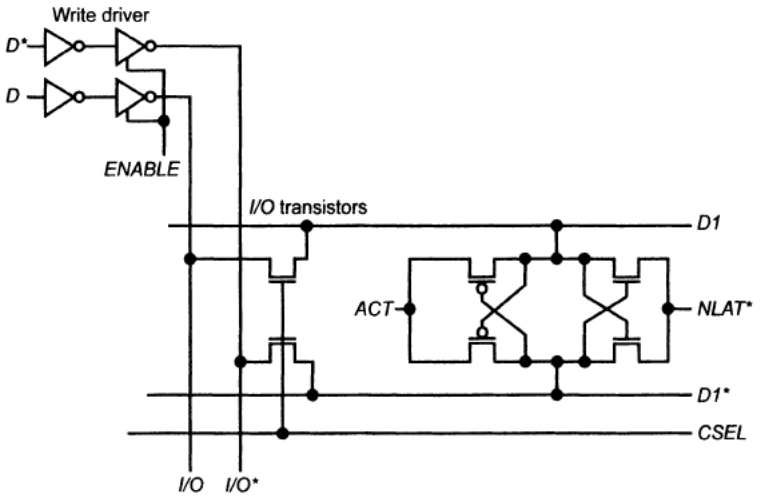


Figure 1.29 Sense amplifier schematic with *I/O* devices.

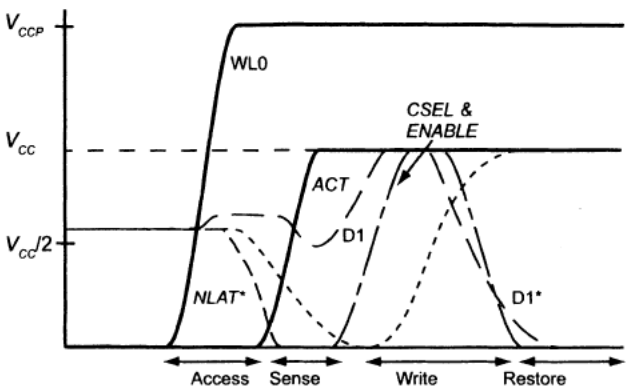


Figure 1.30 Write operation waveforms.

1.2.3 Opening a Row (Summary)

Opening a row of mbits in a DRAM array is a fundamental operation for both reading and writing to the DRAM array. Sometimes the chain of events from a circuit designer's point of view, which lead to an open row, is called the *RAS** timing chain. We summarize the *RAS** timing chain of events below, assuming that for a second-generation DRAM both *RAS** and *CAS** are HIGH. (It's easy to extend our discussion to third-generation DRAMs where *RAS** and *CAS** are effectively generated from the control logic.)

1. Initially, both *RAS** and *CAS** are HIGH. All bitlines in the DRAM are driven to $V_{CC}/2$, while all wordlines are at 0 V. This ensures that all of the mbit's access transistors in the DRAM are OFF.
2. A valid row address is applied to the DRAM and *RAS** goes LOW. While the row address is being latched, on the falling edge of *RAS**, and decoded, the bitlines are disconnected from the $V_{CC}/2$ bias and allowed to float. The bitlines at this point are charged to $V_{CC}/2$, and they can be thought of as capacitors.
3. The row address is decoded and applied to the wordline drivers. This forces only one rowline in at least one memory array to V_{CCP}. Driving the wordline to V_{CCP} turns ON the mbits attached to this rowline and causes charge sharing between the mbit capacitance and the capacitance of the corresponding bitline. The result is a small perturbation (upwards for a logic one and downwards for a logic zero) in the bitline voltages.
4. The next operation is Sensing, which has two purposes: a) to determine if a logic one or zero was written to the cell and b) to refresh the contents of the cell by restoring a full logic zero (0 V) or one (V_{CC}) to the capacitor. Following the wordlines going HIGH, the Nsense-amp is fired by driving, via an n-channel MOSFET, *NLAT** to ground. The inputs to the sense amplifier are two bitlines: the bitline we are sensing and the bitline that is not active (a bitline that is still charged to $V_{CC}/2$—an inactive bitline). Pulling *NLAT** to ground results in one of the bitlines going to ground. Next, the *ACT* signal is pulled up to V_{CC}, driving the other bitline to V_{CC}. Some important notes:

   (a) It doesn't matter if a logic one or logic zero was sensed because the inactive and active bitlines are pulled in opposite directions.

   (b) The contents of the active cell, after opening a row, are restored to full voltage levels (either 0 V or V_{CC}). The entire DRAM can be refreshed by opening each row.

Now that the row is open, we can write to or read from the DRAM. In either case, it is a simple matter of steering data to or from the active array(s) using the column decoder. When writing to the array, buffers set the new logic voltage levels on the bitlines. The row is still open because the wordline remains HIGH. (The row stays open as long as *RAS** is LOW.)

When reading data out of the DRAM, the values sitting on the bitlines are transmitted to the output buffers via the *I/O* MOSFETs. To increase the speed of the reading operation, this data, in most situations, is transmitted to

the output buffer (sometimes called a *DQ* buffer) either through a helper flip-flop or another sense amplifier.

A note is in order here regarding the word size stored in or read out of the memory array. We may have 512 active bitlines when a single rowline in an array goes HIGH (keeping in mind once again that only one wordline in an array can go HIGH at any given time). This literally means that we could have a word size of 512 bits from the active array. By adjusting the word size, or the number of these 512 bits we send to the chip's output, we can change the speed on power performance of the chips. For example, if our chip's *I/O* is x4 (by 4, that is, 4-bit input/output), then sending eight bits at a time (out of these 512) provides data for two clock edges. Sending 32 bits provides data for eight clock edges. The cost of transmitting larger word sizes is larger bus widths (more layout area). However, the benefit is that more time is available to send the data once it is in the pipeline. The output word size is the basic difference between the DDR, DDR2, and DDR3 DRAM topologies.

1.2.4 Open/Folded DRAM Array Architectures

Throughout the book, we make a distinction between the open array architecture as shown in Figures 1.22 and 1.24 and the folded DRAM array used in many modern DRAMs and seen in Figure 1.31. At the cost of increased layout area, folded arrays increase noise immunity by moving sense amp inputs next to each other. These sense amp inputs come directly from the DRAM array. The term *folded* comes from taking the DRAM arrays seen in Figure 1.24 and folding them together to form the topology seen in Figure 1.31.

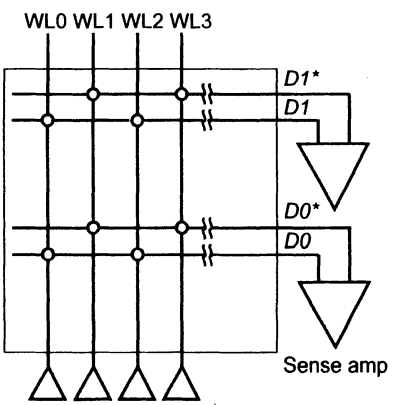


Figure 1.31 A folded DRAM array.

REFERENCES

[1] R. J. Baker, *CMOS: Circuit Design, Layout, and Simulation*, 2nd ed. Hoboken, NJ: John Wiley & Sons, Inc., and IEEE Press, 2005.

[2] Micron Technology, Inc., Synchronous DRAM datasheet, 1999.

Chapter
7

An Introduction to High-Speed DRAM

Historically, the semiconductor industry has been the slave to two masters: cost and performance. Cost, or more specifically cost reduction, is a matter of pure economic survival for most companies in this industry. It is cost after all that drives the industry to aggressively shrink process technology year after year. Memory chips, due to their status as a commodity product, are highly sensitive to cost. This fact fosters additional incentives for memory companies to push their process technology development faster than most other segments of the semiconductor industry. The historical record is littered with the names of companies that produced memory products at one time or another, but subsequently dropped out of the marketplace. A key contributor to this attrition is the ability or inability of any given company to remain cost competitive. Market forces, not supplier controls, determine the basic selling price for commodity memory—most notably, the balance or imbalance in supply and demand. This leaves cost as the only component of a balance sheet that can be controlled. Control costs and survive. Lose control and fade away.

7.1 THE PERFORMANCE PARADIGM

While a complete analysis of the cost issues facing memory companies would be interesting, the remainder of this book deals with a topic more fascinating and dear to engineering professionals: performance. The word *performance*, much like the words *quality* or *value*, is rather ambiguous. Yet, when we talk about performance, most people assume we are referring to *high* performance. This is a natural and appropriate assumption, as we rarely strive for low performance, let alone write books about it. Just to set

the record straight, the remaining chapters deal entirely with high-performance DRAM.

Now, DRAM may not be what first comes to mind when you hear the phrase *high performance*. We agree with that sentiment as images of race cars and fighter aircraft flash through our minds. But, when we come back to reality and refocus our attention on the semiconductor industry, DRAM memory devices still don't come to mind. Rather, we are more likely to consider microprocessors. Clearly, these devices are on the leading edge of high performance, and they benefit from the use of the latest transistor, metal, and packaging technology in the quest for speed and processing muscle.

Memory, on the other hand, tends to limp along with slower transistors, a minimum of metal layers, and inexpensive packaging. The goal at DRAM companies has always been to meet the necessary speed targets while keeping manufacturing costs to an absolute minimum. This doesn't mean that the DRAM processes are not highly advanced. They clearly *are* advanced. The real difference between microprocessor and DRAM process technology has been the focus of process development. While the focus for microprocessors has been performance, the focus in the memory sector has always been cost. For DRAM, this focus has become a myopic quest for smaller and smaller process geometries, with the ultimate goal of yielding more die per wafer and, hence, at a lower cost than the competition.

Only recently has the need for higher performance DRAM become a pressing concern. There are two sectors of the memory industry that are driving this issue. The first is high-performance desktop systems; the second is high-end graphics cards. Both of these sectors are extremely competitive, and they are highly prized for the pricing margins and marketing mileage available to the best in class. With processor speeds racing into the multi-gigahertz range there are greater incentives for the memory subsystem to try and keep pace. Pricing pressures are lower in this sector, which gives DRAM manufacturers more latitude with their process technology, design architectures, and packaging in order to realize desired performance gains. These performance gains and the underlying technology eventually trickle down, over time, to the higher volume desktop market—increasing performance levels for the mainstream market, but ultimately raising the bar again in the high-performance desktop market.

DRAM devices developed for the graphics card markets follow a similar path, except at a more accelerated pace. High-end graphics cards create a need for very high-performance DRAM devices. Given lower pricing pressure, DRAM manufacturers can improve their process technology, designs, and packaging to significantly increase performance. Ultimately, these improvements trickle down to other portions of the graphics market and

even into those main memory devices being developed for the high-performance desktop market.

7.2 PERFORMANCE FOR DRAM MEMORY DEVICES

So, how do we define high performance for DRAM memory devices? Historically, asynchronous, fast page mode-style DRAM device performance was dictated by timing specifications, such as t_{AA} (access time from column access), t_{CAC} (access time from *CAS**), t_{RAC} (access time from *RAS**), and t_{PC} (Read or Write cycle time). Note that these specs, save the final one, are all measures of access time, which is the amount of time it takes for the device to drive out data following the transition of the column address, the column address strobe (*CAS**), or the row address strobe (*RAS**). Only the final spec, t_{PC}, relates to the cycle time, that is, the ability of the device to deliver data at a sustained rate from one column address to another. There were many other specifications on an asynchronous DRAM data sheet, but these four, far and away, defined performance.

What was so important about these particular specifications? With regard to t_{AA}, t_{CAC}, and t_{RAC}, these specs all defined some sort of delay: delay that the memory controller must endure every time that new data is needed from the DRAM. The shorter the delay, the sooner the memory controller could ultimately deliver data to the processor. Whenever the processor waits for data, its performance, and that of the system, suffers. Any improvement in these critical DRAM timing specifications ultimately leads to improvements in system performance. As a result, the memory industry naturally migrates towards the development of devices with faster access and shorter cycle times.

Following the advent of synchronous operation, DRAM device specifications began a slow migration towards performance metrics related to clock frequency. The most important timing parameters became t_{AC} (access time from clock), t_{CK} (clock cycle time), *CL* (*CAS* latency), and t_{RCD} (active to read or write delay). Most of these new parameters are specified in terms of nanoseconds, except for *CAS* latency, which is specified in number of clock cycles. Again, all, save t_{CK}, are a measure of delay. While the t_{AC} parameter is somewhat important, because it defines data availability relative to clock transitions, real performance is measured by *CAS* latency and t_{CK}. These two parameters define how fast the DRAM device can cycle (t_{CK}) and how many clock cycles the memory controller, and ultimately the system, must wait for data following a request for data (a Read operation).

Of course, system designers want their DRAM devices to have the lowest *CAS* latency possible at any given operating frequency. This combina-

tion yields the best system performance, but reality being what it is, won't result in acceptable device yields for the memory manufacturer. Generally, both t_{CK} and *CAS* latency incrementally improve with successive generations. Both memory manufacturers and system builders manage device performance, and the inevitable improvements, through a system known as *speed grading*. Basically, a speed grade is a set of timing parameters associated with device operation at a specific data bus frequency. At higher bus frequencies, these timing parameters become more demanding and harder to meet. Fortunately for the memory industry, devices that are unable to meet all of the timing parameters for a specific speed grade can be down-binned to a slower speed grade and sold into other market segments. Speed grades are not only important to ensure that components from different suppliers are compatible, but also to support down-binning strategies that improve overall production yields.

Synchronous DRAM (SDRAM), despite notable improvements in performance, represents only an evolution in technology. When SDRAM first appeared, manufacturers essentially converted existing asynchronous designs by merely changing *I/O* circuits to accommodate the synchronous interface. Even the device specifications remained rooted in asynchronous behavior. The real advantage gained by synchronous operation occurred in the output timing specs. Asynchronous designs had very limited data rates due to the fact that the memory controller could not issue a new data request (Read operation) until it successfully captured data from the previous Read. The controller drove *CAS** LOW to initiate the Read operation and held *CAS** LOW until it was able to capture the Read data. This meant that column cycle times included not just the DRAM access delay, but also the *I/O* transfer delays, including flight time. Accordingly, column cycle time was not optimized, because the controller was precluded from issuing a subsequent read command until it had captured data from the current read command.

Extended Data Out (EDO) asynchronous DRAMs reduced column cycle time by hiding some of the *I/O* transfer delay. While traditional asynchronous devices, such as Page Mode (PM) or Fast Page Mode (FPM), only drive Read data while *CAS** is LOW, EDO devices continue to fire Read data after *CAS** transitions HIGH. This allows the memory controller to begin to cycle the column for a subsequent Read while concurrently capturing data from the current Read. This seemingly small change allowed a reasonable gain in system performance.

SDRAM represents another step in the steady evolution of DRAM technology towards higher performance. The synchronous interface specification is important, as it essentially divides the Read operation into two separate components: the column access phase and the data output phase.

Essentially, the data output operation was pushed into a separate clock cycle from that of the array access. This meant that device speed and the associated clock frequency were now only limited by how fast the memory array could perform column accesses. Synchronization ultimately increased performance because the data output operation became a scheduled event, isolated from the array access.

Double data rate (DDR)-style DRAM devices and their faster cousins, graphics double data rate (GDDR) devices, represent an evolution in technology compared to synchronous DRAM (SDRAM). For each Read command, they drive two bits of data for every clock cycle: one bit on the rising edge and a second bit on the falling edge. This simple change, dating from technology developed in the early 1980s, effectively doubles data transfer rates to and from the DRAM [1]. The various permutations of DDR and GDDR, such as DDR2, DDR3, GDDR2, GDDR3, and GDDR4, encompass evolutionary advances in technology to achieve higher overall data transfer rates. These advances are enabled by changes in burst length, improvements in signaling technology and packaging, and significant advances in circuit design.

7.3 UNDERLYING TECHNOLOGY IMPROVEMENTS

In the balance of the book we discuss the underlying technology improvements that enable high-speed DRAM. Many of these improvements don't show up all at once, but begin to appear, in various forms, at each step of the high-speed staircase. Some of the advances are somewhat delayed, if not outright restrained, by legacy baggage that gets carried along from one generation to the next. We discuss the importance of this issue as well in upcoming chapters.

REFERENCE

[1] M. F. Novak, K. M. Guttag, and D. J. Redwine. 1987. Control of data access to memory for improved video system. US Patent 4,688,197, issued Aug. 18, 1987.