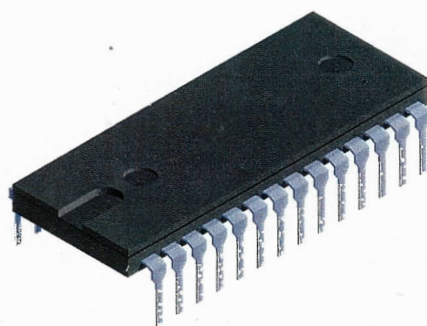


NEW DRAM TECHNOLOGIES

A Comprehensive Analysis of the New Architectures

SECOND EDITION



Steven A. Przybylski

MICRODESIGN
RESOURCES

From the publisher of MICROPROCESSOR REPORT

Netlist EX2029
Samsung v Netlist
IPR2023-00454

New DRAM Technologies

**A
Comprehensive
Analysis
of the
New Architectures**

Second Edition

Steven A. Przybylski

with

Michelle M. Deatrick

**MICRODESIGN
RESOURCES**

Author:

Steven A. Przybylski
Consultant
3281 Lynn Oaks Dr.
San Jose, CA 95117
408.984.2719, FAX: 408.246.2412
sp@acm.org

With:

Michelle M. Deatruck

Publisher:

MicroDesign Resources
874 Gravenstein Hwy. So.
Suite 14
Sebastopol, CA 95472
707.824.4004, FAX: 707.823.0504

Copyright © 1994,1996 Steven A. Przybylski

ISBN 1-885330-06-5

All Rights Reserved. No part of this report may be reproduced, stored in a retrieval system, or transmitted in any form or by any means without prior written permission.

MicroDesign Resources is a trademark of Ziff Communications Co.

This report contains and analyzes information from publicly available sources and from industry contacts. While we consider these sources reliable, we cannot guarantee the accuracy, whether past, present or future, of the information or analyses contained herein. Readers assume full responsibility for any use made of this report or its contents, in whole or in part.

Trademark names are used throughout this report. These trademarks are the property of their respective owners.

Please inform the author of any errors of fact or omission so that they may be corrected in future editions of the report.

Printed on recycled paper.

Executive Summary

The Dynamic RAM (DRAM) interface has changed very little in two decades. The growing inadequacy of this interface is driven by two long-term trends:

- **DRAM capacities have consistently quadrupled every three years.** The next-generation 64-Mbit devices, expected to be price per bit leaders by 1997 or 1998, will contain 8 Mbytes each. Consequently, by the end of the decade, most computer systems will be shipped with four or fewer DRAMs.
- **DRAM performance has not kept pace with processor performance.** DRAM access times have improved by only 7% per year while processor performance has grown at about 60% per year. While the available bandwidth per bit stored has declined continuously and dramatically, the demands placed on memory have increased.

These two trends result in a dilemma. In an era with strong motivations for both greater performance and fewer DRAMs per system, the existing DRAM interface is capable of improving either bandwidth or memory granularity (increment size), but not both simultaneously:

- **High-bandwidth memory systems cannot be made with only a few DRAMs.**
- **Increasingly, adequate performance only comes with excessive granularity.**

The need for change in the DRAM interface is pressing. The semiconductor industry is responding with a variety of new DRAM architectures that offer, in varying degrees, shorter access times, more bandwidth, and finer memory system granularity—all at a price. This report explores the need for change, contrasts the alternatives offered, and discusses the markets in which they are likely to prosper. Depending on the degree of change to the interface, alternative DRAMs are classified as conventional, evolutionary, or revolutionary:

- **Conventional choices** involve relatively minor changes to the generic interface: increased widths and/or extended data-out times.
 - **Extended Data-Out (EDO) DRAMs**, also called **Hyper-Page-Mode DRAMs**, incorporate a small change to the semantics of one control signal. This change allows the operating frequencies of memory systems to increase from approximately 25 MHz to 33–40 MHz. Even in conjunction with an increase in device width—from $\times 4$ up to $\times 8$ and $\times 16$ —this performance improvement is too small to solve the bandwidth problem in the long term. However, the EDO DRAMs' large number of suppliers and very small price premium will make wide EDO DRAMs a good fit for many applications during the near term.
- **Evolutionary architectures** provide substantially more bandwidth than do the generic DRAMs. Evolutionary DRAMs continue to rely on the same separate data bus and multiplexed address bus as the old generic DRAMs.
 - **Burst EDO (BEDO) DRAMs** allow pseudosynchronous bursts of four at frequencies up to 66 MHz. Burst EDO devices can be built on the same die as generic and EDO DRAMs. Consequently, Burst EDO DRAMs should command only a negligible price premium. However, since faster-than-average DRAMs are needed to run at 66 MHz, only a few DRAM manufacturers are committed to producing Burst EDO devices.
 - Enhanced Memory Systems' **Enhanced DRAMs (EDRAMs)** offer dramatically reduced access times and an improved data-write interface. The 25% cost premium is substantial. However, in small, tightly coupled systems, the cost premium can be offset by eliminating a secondary cache. Initial 4-Mbit offerings are in production though the supply will remain severely constrained until at least 1Q1996.

Continues next page

- **Synchronous DRAMs (SDRAMs)**, developed and standardized by JEDEC, are well placed to become the next commodity part. Their high-speed synchronous interface and optional dual-bank architecture offer substantially higher bandwidths than are possible with generic DRAMs. SDRAMs are in production at the 4- and 16-Mbit levels. The initial adoption of SDRAMs was inhibited by a host of factors, including high price premiums, lack of standardized SIMMs, incompatibilities among the vendors' offerings, and confusion over the choice of a high-speed alternative to TTL levels. Though some of the incompatibilities persist, many of the initial barriers, especially high premiums, are disappearing.
- Mitsubishi's **Cache DRAMs (CDRAMs)** add a small block of SRAM and a synchronous interface to a generic DRAM. The result is a high-performance device that offers low latency and high sustained bandwidth. CDRAMs suffer from a high pin-count and a 15% cost premium. They are well suited to graphics applications and to smaller computer systems in which main memory and graphics memory are combined into a unified memory structure. As with EDRAMs, CDRAMs' combination of an effective row cache and low latency facilitates elimination of an SRAM-based secondary cache.
- **Revolutionary alternatives** offer a complete redesign of the DRAM interface. The redesign provides high bandwidth and minimum granularity. The revolutionary devices replace the separate address and data bus of the generic DRAM with a byte-wide interface that carries address, data, and signaling information.
- Vertically or horizontally mounted **Rambus DRAMs (RDRAMs)** connect to a processor or memory controller through a physically short, 533 Mbytes/s bidirectional bus. Rambus Inc. has defined all aspects of the physical, electrical, and logical interface and has licensed a variety of DRAM and logic product vendors to produce Rambus-compatible devices. Complete specification of the Rambus Channel facilitates very high-speed systems with a minimum of design effort by system vendors. As of 1Q1996, 16- and 18-Mbit RDRAMs are available, with 8-Mbit devices sampling. RDRAM latencies for short transfers are somewhat higher than the latencies of some alternatives. However, RDRAMs facilitate small high-bandwidth memory systems; they are well suited for graphics and video controllers, unified main memories, and high-performance portable systems. Future 64-Mbit RDRAMs will offer an excellent combination of high bandwidth and moderate memory granularity to be effective main memory devices across a broad range of system types.
- The **RamLink** interface specification, developed by the IEEE Computer Society P1594.6 Working Group, defines a protocol and ring topology for communication with DRAMs and other devices over a high-speed, byte-wide interface. A derivative called **SyncLink** migrates the RamLink protocol onto a dual bus physical implementation with 1 Gbyte/s peak bandwidth. SyncLink is moving through the standardization process with the support of a consortium of DRAM companies. Regardless, no RamLink- or SyncLink-based products are expected for at least two to three years.

Generic DRAMs are becoming increasingly inadequate. Each new alternative has specific strengths and weaknesses that make it a good fit for some DRAM applications and a poor fit for others. Toward the end of the decade, when the 64-Mbit generation becomes the cost leader, all significant market segments will have shifted away from generic fast-page-mode devices. The result will be a fragmented and unsettled marketplace with a few winners and many niche architectures.

This report uses a combination of new and old tools to examine the selection of DRAM architecture for main memory applications through the year 2000. In addition to looking at the characteristics of the various devices both individually and comparatively, this report provides accurate and in depth comparative analysis of the memory systems that these new DRAMs are capable of producing. A powerful new metric—the fill frequency—is used to identify which DRAMs are simply unacceptable in particular systems marketplaces.

Acknowledgments

The authors would like to thank the many people who have directly and indirectly helped during the preparation of this report. They include:

Betty Prince of Memory Strategies International
Russ Tuck and Tom Blank of MasPar
Bob Proebsting and Bill Scharrenberg of Hyundai
Gene Cloud, Bob Fusco, and Sharron Bittick of Micron
Dave Mooring, Steve Ray, Billy Garrett, Richard Crisp,
Julie Cates, and the other reviewers at Rambus
David Bondurant and Todd Oseth of Ramtron
Rajit Shah of Mosel-Vitelic
Satoru Kobayashi and Bart Ladd of NEC
Charles Hart, Sam Chen, Doug Gates, and Sherry Hill of
Mitsubishi
Cormac O'Connell of MOSAID
Peter Cheng, D.G. Kim and Mark Ellsberry of Samsung
Mark Horowitz and Mehrdad Heshami of Stanford University
Hans Wiggers of Hewlett Packard
Jim Handy of Dataquest
Wolf-Dietrich Weber of HaL Computer Systems
Akhil Duggal of Alantec
Melanie Sanborn, Leslie Davidson, Ron Liskey, Bruce Koball,
John Snell, Jeff Deutsch, and Linley Gwennap of
MicroDesign Resources
Gail Anderson of Pine Cone Communications
Liz Longsworth of PC Computing
Ellen Clements

In addition to the above, the authors would like to thank the following people who provided information or helped review the second edition:

Craig Hampel, Julie Cates, Subodh Toprani, and Fred Ware of
Rambus
Peter Cheng and Joe Kraus of Samsung
David Gustavson of SCIZZL
Joseph McIntyre and Ron Liskey of MicroDesign Resources
Jeff Mailloux, Brett Williams, and Sharron Bittick of Micron
Mark Levy of EDN
Sam Chen, Steve Forman, and Sherry Hill of Mitsubishi
Charles Furnweger and Tina Lim of NEC
David Bondurant of Enhanced Memory Systems
Michael Jahed of Toshiba
Betty Prince of Memory Strategies International
Jim Sogas of Hitachi

About the Publisher

Founded in 1987, MicroDesign Resources is an analytical information services company focusing on the microprocessor and PC industries. MicroDesign Resources publishes *Microprocessor Report*, widely regarded as the leading independent newsletter in the field, and sponsors two conferences, the PC Tech Forum and the Microprocessor Forum. The PC Tech Forum will debut in May 1996 and will bring a sophisticated understanding of systems-level technologies together with a grasp of where the industry and the markets are headed. Now in its ninth year, the Microprocessor Forum is the most important annual gathering of microprocessor technologists in the industry. Michael Slater, President of MicroDesign Resources, is an international authority on microprocessor technology and trends. *Fortune* magazine has called Slater “the one expert to whom executives from Intel and the PowerPC consortium defer when asked to reconcile their wildly conflicting claims about the merits of their chips.” Other analysis reports in MicroDesign Resources’ Technical Library are *Selecting a High-Performance Embedded Microprocessor* and *Buyer’s Guide to DSP Processors*.

Table of Contents

Executive Summary	iii
Acknowledgments	v
About the Publisher	vii
Table of Contents	ix
List of Figures	xv
List of Tables	xix
 1. Introduction	 1
1.1 Outline	2
<i>Introductory and Tutorial Material</i>	<i>2</i>
<i>Individual Alternatives</i>	<i>3</i>
<i>Comparative and Market Analysis</i>	<i>4</i>
<i>Supporting Materials and Resources</i>	<i>4</i>
1.2 Changes Since the First Edition	5
<i>New Alternatives and Significant Events</i>	<i>5</i>
<i>Market Changes</i>	<i>6</i>
<i>Other Changes since the First Edition</i>	<i>7</i>
1.3 Which Chapters Should You Read?	7
<i>Engineers and Engineering Management</i>	<i>7</i>
<i>Marketers and Market Analysts</i>	<i>8</i>
<i>General Interest Readers</i>	<i>8</i>
1.4 Conventions in This Report	8
 2. Industry Trends and Their Implications	 11
2.1 New In The Second Edition	11
2.2 DRAM Trends	11
<i>DRAM Speed Trends</i>	<i>13</i>
<i>Technology and Economic Trends</i>	<i>15</i>
2.3 Computer System Trends	19
<i>Computer CPU Trends</i>	<i>19</i>
<i>Computer Memory System Trends</i>	<i>21</i>
<i>Computer Graphics Trends</i>	<i>25</i>
2.4 Combining DRAM and Systems Trends	28
2.5 Fill Frequency: A New DRAM Metric	31

3. DRAM Basics	41
3.1 New In The Second Edition	41
3.2 Introduction	41
3.3 What Is a DRAM?	42
<i>The Basics</i>	42
<i>DRAM Internals</i>	48
3.4 DRAM Fabrication Technology	54
3.5 Advanced DRAM Concepts	60
<i>Soft Errors</i>	60
<i>Supply Voltages</i>	60
<i>Testing and Redundancy in DRAMs</i>	62
<i>Speed-Enhancing Techniques</i>	64
3.6 Video RAMs (VRAMs)	65
4. Memory System Design	69
4.1 New In The Second Edition	69
4.2 Basic Concepts	70
<i>Single In-Line Memory Modules (SIMMs)</i>	70
<i>Memory System Organization: Granularity and Expandability</i>	76
<i>Bandwidth and Latency: Peak, Sustained, and Sustainable</i>	81
<i>Performance and Bandwidth Considerations</i>	84
<i>Parity and Error Correction Codes (ECC)</i>	89
<i>Refresh</i>	91
4.3 High-Speed Design Issues	92
<i>High-Speed Signals and Terminated Transmission Lines</i>	92
<i>Clock and Data Skew Issues</i>	96
<i>Issues Specific to Memory Systems</i>	98
4.4 Memory Systems for PCs	100
<i>x86 Machines</i>	100
<i>The Macintosh Family</i>	103
<i>Portables</i>	104
4.5 Memory Systems for Workstations	104
4.6 Memory Systems for Large Machines	107
4.7 Graphics Memory Systems	110
4.8 Unified Memory Architecture (UMA)	117
5. New DRAM Architectures	129
5.1 New In The Second Edition	129
5.2 Potential Improvements to the Existing Interface	129
<i>The Physical Interface</i>	130
<i>The Electrical Interface</i>	136
<i>The Pin-Level Interface</i>	147
<i>The Temporal Interface</i>	148
<i>The Logical Interface</i>	148
5.3 Generic Narrow DRAMs	150
5.4 Conventional Approaches	151
5.5 Evolutionary Approaches	151
5.6 Revolutionary Approaches	153

6. Conventional Alternatives	157
6.1 New In The Second Edition	157
6.2 Wide Conventional DRAMs	158
6.3 Extended Data-Out DRAMs	164
7. Evolutionary Alternatives	171
7.1 New In The Second Edition	171
7.2 Burst EDO (BEDO) DRAMs	172
<i>Physical, Electrical, Pin-Level Interface</i>	172
<i>Temporal and Logical Interface</i>	173
<i>System Design Issues</i>	178
<i>Burst EDO Fill Frequencies</i>	180
<i>History, Current Status, and Future Plans</i>	181
<i>Strengths and Weaknesses</i>	182
7.3 Enhanced DRAMs (EDRAMs)	183
<i>Physical, Electrical, and Pin-Level Interfaces</i>	184
<i>Temporal Interface</i>	186
<i>Logical Interface</i>	189
<i>EDRAM Fill Frequencies</i>	198
<i>Miscellanea</i>	199
<i>Future Developments</i>	202
<i>Strengths and Weaknesses</i>	203
7.4 Synchronous DRAMs (SDRAMs)	204
<i>Physical, Pin-Level, and Temporal Interfaces</i>	204
<i>Logical Interface</i>	212
<i>Electrical Interface</i>	229
<i>SDRAM Fill Frequency</i>	230
<i>SDRAM-Lite</i>	233
<i>History and Barriers to Acceptance</i>	235
<i>Prices, Costs, and Design Wins</i>	238
<i>Current Offerings</i>	241
<i>Strengths and Weaknesses</i>	242
7.5 Cached DRAMs (CDRAMs)	243
<i>Physical, Pin-Level, Temporal, and Electrical Interfaces</i>	245
<i>Logical Interface</i>	248
<i>The 16-Mbit CDRAM</i>	252
<i>CDRAM Fill Frequencies</i>	253
<i>Miscellanea</i>	255
<i>Strengths and Weaknesses</i>	257
8. Revolutionary Alternatives	259
8.1 New In The Second Edition	259
8.2 Rambus DRAMs (RDRAMs)	260
<i>Pin-Level and Physical Interfaces</i>	261
<i>Electrical Interface</i>	267
<i>Temporal Interface</i>	271
<i>Logical Interface</i>	274

<i>Miscellanea</i>	289
<i>First Generation 4-Mbit RDRAMs</i>	295
<i>The 64-Mbit RDRAMs</i>	296
<i>RDRAM Fill Frequencies</i>	299
<i>Current Status and Business Issues</i>	300
<i>Strengths and Weaknesses</i>	305
8.3 RamLink DRAMs	306
<i>Pin-Level Interface</i>	307
<i>Physical, Electrical, and Temporal Interfaces</i>	309
<i>Logical Interface</i>	310
<i>Current Status and Future Developments</i>	317
<i>Strengths and Weaknesses</i>	319
8.4 SyncLink DRAMs (SLDRAMs)	320
<i>Bus Structure and Pin-Level Interface</i>	322
<i>Physical, Electrical, and Temporal Interfaces</i>	324
<i>Logical Interface</i>	325
<i>Changes from RamLink</i>	332
<i>SLDRAM Fill Frequency</i>	333
<i>Current Status and Business Issues</i>	334
<i>Strengths and Weaknesses</i>	335
9. Chip-Level Comparison	337
9.1 New In The Second Edition	337
9.2 Pin-Level and Physical Interfaces	338
9.3 Logical Interface	339
9.4 Electrical and Temporal Interfaces.....	345
9.5 Peak Fill Frequencies	348
9.6 Power Consumption	349
9.7 Price and Supply.....	350
10. System-Level Comparison	353
10.1 New In The Second Edition	354
10.2 High-Performance Memory System.....	355
<i>Memory Size and Granularity</i>	366
<i>Row-Cache Organizations</i>	367
<i>Access Latencies</i>	372
<i>Sustainable Bandwidths</i>	377
<i>Fill Frequencies</i>	381
<i>Miscellanea</i>	383
10.3 PC Memory System	386
<i>Memory Size and Granularity</i>	389
<i>Row-Cache Organizations</i>	396
<i>Access Latencies</i>	397
<i>Sustainable Bandwidths</i>	405
<i>Fill Frequencies</i>	408
<i>Unified Memory Architecture PCs</i>	410
<i>Miscellanea</i>	411

10.4	Memory Systems for Portable Systems.....	413
	<i>Memory Size and Board Area</i>	415
	<i>Row-Cache Organization, Bandwidth, and Access Time</i>	416
	<i>Power Considerations</i>	419
10.5	Embedded Applications.....	425
	<i>Video Games and Other High-Performance Embedded Systems</i>	425
	<i>Other Consumer Electronics and HDTV</i>	427
10.6	Other DRAM Applications	428
	<i>Frame Buffer Memories</i>	428
	<i>Mainframes, Supercomputers, and Massively Parallel Machines</i>	430
11.	Market Forces and Opportunities	433
11.1	New In The Second Edition	433
11.2	Market Segments and the DRAM Alternatives	434
	<i>DRAM and VRAM Market Segments</i>	434
	<i>Strengths and Weaknesses of the DRAM Alternatives</i>	438
	<i>Matching Market Segment Needs To DRAM Strengths</i>	440
	<i>Changes at the 64-Mbit Density</i>	450
	<i>Comparison of Peak Fill Frequencies</i>	451
11.3	DRAM Costs and Pricing	454
	<i>Cost of Goods</i>	454
	<i>DRAM Pricing</i>	457
	<i>Expected Pricing for New Alternatives</i>	463
11.4	Changing DRAM and VRAM Market Dynamics	468
11.5	The Coming Fragmentation of the DRAM Market.....	473
11.6	Barriers to Market Acceptance	475
A.	List of Acronyms	479
B.	Glossary	485
C.	List of Vendors	507
D.	References	511
E.	Reprinted Data Sheets	517
E.1	A Generic 4M×4 Fast-Page-Mode DRAM Data Sheet	517
E.2	A 1M×16 Fast-Page-Mode DRAM Data Sheet.....	537
E.3	A 4M×4 Extended Data-Out DRAM Data Sheet	577
E.4	A 1M×16 Burst EDO DRAM Data Sheet	593
E.5	A 1M×4 Enhanced DRAM Data Sheet	613
E.6	A 4M×4, 2M×8, and 1M×16 Synchronous DRAM Data Sheet	637
E.7	A 1M×16 Cached DRAM Data Sheet.....	741
E.8	A 16-Mbit and 18-Mbit Rambus DRAM Data Sheet	809
	Index	841

List of Figures

Figure 2-1	DRAM density versus time.	12
Figure 2-2	Row and column access times versus time.	14
Figure 2-3	Minimum feature size versus time.	16
Figure 2-4	Fab line capital cost versus time.	17
Figure 2-5	Memory cost versus time.	17
Figure 2-6	CPU performance versus time.	19
Figure 2-7	Cache miss ratio versus cache size.	21
Figure 2-8	Interaction effects that determine system memory size.	22
Figure 2-9	Memory size and number of DRAMs per system versus time.	29
Figure 2-10	Memory size and number of DRAMs per system versus time to 1999.	30
Figure 2-11	Traditional and modified view of the DRAM performance gap.	32
Figure 2-12	Long-term trends in DRAM and system fill frequencies.	37
Figure 3-1	DRAM storage cell.	43
Figure 3-2	Subdivision of the DRAM address.	44
Figure 3-3	Basic DRAM timing diagram.	46
Figure 3-4	DRAM internals.	49
Figure 3-5	Internal voltage levels during a data read and refresh.	51
Figure 3-6	Typical sense amplifier.	52
Figure 3-7	Physical organization of a 4M×1 DRAM.	55
Figure 3-8	Physical organization of a 512K×8 DRAM.	56
Figure 3-9	A planar DRAM cell cross-section.	58
Figure 3-10	A trench DRAM cell cross-section.	58
Figure 3-11	A stacked DRAM cell cross-section.	59
Figure 3-12	Soft error rate versus time.	61
Figure 3-13	Number of soft errors over 10 years of use versus time.	61
Figure 3-14	Typical graphics controller block diagram.	65
Figure 3-15	VRAM block diagram.	66
Figure 4-1	Standard 30-pin SIMM physical diagram.	71
Figure 4-2	Four-Mbyte byte-wide SIMM block diagram.	72
Figure 4-3	2M×36 SIMM block diagram.	73
Figure 4-4	Buffered SIMM block diagram.	74
Figure 4-5	Typical low-end memory system block diagram.	77
Figure 4-6	Ranks, banks, and sub-banks.	79
Figure 4-7	Burst transfer timing diagram.	81
Figure 4-8	Interleaved burst transfer timing diagram.	84
Figure 4-9	Interleaved memory system block diagram.	85

Figure 4-10	Physical address fields	86
Figure 4-11	Block diagram of a typical Pentium® PC.....	101
Figure 4-12	Block diagram of a high-end workstation.....	106
Figure 4-13	Low-end graphics controller block diagram	112
Figure 4-14	Mid-range graphics controller block diagram	113
Figure 4-15	Traditional untiled frame buffer	114
Figure 4-16	A tiled frame buffer makes better use of DRAM row accesses.....	115
Figure 4-17	Traditional PC graphics architecture	117
Figure 4-18	Dual-supplier unified memory architecture (UMA) PC	120
Figure 4-19	Single-supplier unified memory architecture (UMA) PC	121
Figure 5-1	DRAM package types.....	132
Figure 5-2	Timing diagram showing excessive ringing	138
Figure 5-3	Gunning Transceiver Logic (GTL) driver circuit diagram.....	140
Figure 5-4	HSTL defines a variety of electrical environments.....	142
Figure 5-5	Stub-series termination mitigates the impact of stub reflections	144
Figure 5-6	Rambus electrical interface circuit diagram	146
Figure 6-1	Block diagram of a ×16 DRAM.....	159
Figure 6-2	Package types for ×16 DRAMs	160
Figure 6-3	Horizontally versus vertically organized memory systems	161
Figure 6-4	Wide DRAM fill frequencies	164
Figure 6-5	Extended Data-Out (EDO) DRAM timing diagram.....	165
Figure 6-6	EDO DRAM fill frequencies	168
Figure 7-1	Burst EDO read timing diagram	174
Figure 7-2	Burst EDO write timing diagram	174
Figure 7-3	Burst EDO fill frequencies	180
Figure 7-4	EDRAM packages	185
Figure 7-5	EDRAM timing diagram	188
Figure 7-6	EDRAM block diagram.....	191
Figure 7-7	EDRAM timing diagram with overlapped row-cache hit and precharge.....	193
Figure 7-8	EDRAM ×8 block diagram.....	196
Figure 7-9	EDRAM burst-mode timing diagram.....	197
Figure 7-10	EDRAM fill frequencies.....	198
Figure 7-11	SDRAM packages.....	205
Figure 7-12	SDRAM basic timing diagram	207
Figure 7-13	SDRAM timing makes at-speed operation difficult	209
Figure 7-14	SDRAM block diagram	213
Figure 7-15	SDRAM addressing across families	215
Figure 7-16	SDRAM command register format	216
Figure 7-17	SDRAM timing diagram.....	219
Figure 7-18	SDRAM timing diagram showing accesses to both banks.....	220

Figure 7-19	SDRAM timing diagram showing long alternating bursts.....	221
Figure 7-20	SDRAM state diagram.....	223
Figure 7-21	SDRAM fill frequencies	231
Figure 7-22	CDRAM block diagram.....	244
Figure 7-23	CDRAM package.....	245
Figure 7-24	CDRAM output modes.....	247
Figure 7-25	CDRAM timing diagram	250
Figure 7-26	CDRAM fill frequencies.....	254
Figure 7-27	CDRAM unified memory architecture block diagram	256
Figure 8-1	RDRAM package and pin descriptions	263
Figure 8-2	Board layout of a segment of the Rambus Channel.....	262
Figure 8-3	Populated Rambus Channel.....	266
Figure 8-4	Rambus Channel high-level block diagram.....	267
Figure 8-5	Rambus Channel circuit diagram	268
Figure 8-6	Rambus Channel detailed block diagram.....	271
Figure 8-7	RDRAM block diagram.....	279
Figure 8-8	Rambus Channel Request Packet format.....	277
Figure 8-9	Rambus Channel timing showing a row-cache hit.....	279
Figure 8-10	Rambus Channel timing showing a row-cache miss.....	280
Figure 8-11	Serial address packets facilitate random octet bursts.....	282
Figure 8-12	RDRAM address mapping.....	286
Figure 8-13	RAC block diagram.....	292
Figure 8-14	RDRAM fill frequencies.....	300
Figure 8-15	RamLink system block diagram	307
Figure 8-16	RamLink slave block diagram.....	308
Figure 8-17	SyncLink system block diagram	310
Figure 8-18	RamLink packet format.....	312
Figure 8-19	New SyncLink system block diagram.....	323
Figure 8-20	SyncLink pinout.....	324
Figure 8-21	SyncLink DRAM block diagram.....	326
Figure 8-22	SyncLink packet format	327
Figure 10-1	High-performance memory system block diagram	356
Figure 10-2	High-performance memory system latencies	373
Figure 10-3	High-performance memory system initial latencies	378
Figure 10-4	High-performance memory system sustainable bandwidths	379
Figure 10-5	High-performance memory system sustainable fill frequencies	382
Figure 10-6	PC memory system block diagram.....	387
Figure 10-7	PC memory system latencies	398
Figure 10-8	PC memory system initial latencies	402
Figure 10-9	PC memory system sustainable bandwidths	407
Figure 10-10	PC memory system sustainable fill frequencies	409

Figure 10-11	The effect of memory technology on PC performance	412
Figure 11-1	Market affinities	442
Figure 11-2	Comparison of DRAM capabilities and market requirements	469
Figure 11-3	DRAM price history	458
Figure 11-4	Historical generational crossover	469
Figure 11-5	Contemporary generational crossover	470
Figure 11-6	Future generational crossover	472
Figure 11-7	DRAM market fragmentation	475

List of Tables

Table 2-1	Graphics standard's display bandwidth requirements.....	25
Table 2-2	System bandwidth and granularity requirements	34
Table 2-3	DRAM bandwidth and size capabilities	35
Table 3-1	Primary DRAM timing parameters	48
Table 4-1	Linear and sub-block ordered burst transfers.....	88
Table 4-2	Typical frame buffer sizes and display bandwidths.....	111
Table 4-3	Characteristics of different UMA architectures.....	123
Table 5-1	DRAM package types.....	135
Table 5-2	Electrical signaling standards	147
Table 5-3	Interface changes in the new DRAM alternatives	154
Table 6-1	Comparison of EDO and fast-page-mode cycle times	166
Table 7-1	Comparison of Burst EDO, EDO, and fast-page-mode timing parameters..	177
Table 7-2	Comparison of Burst EDO, EDO, and fast-page-mode in PCs.....	179
Table 7-3	EDRAM timing parameters	187
Table 7-4	EDRAM PC system response times	190
Table 7-5	SDRAM timing parameters.....	207
Table 7-6	Typical 66-MHz system timing parameters	210
Table 7-7	SDRAM access times	217
Table 7-8	SDRAM command encodings	222
Table 7-9	SDRAM-Lite feature set	234
Table 7-10	CDRAM timing parameters	246
Table 7-11	CDRAM operations.....	249
Table 8-1	RDRAM configuration registers.....	276
Table 8-2	RDRAMs' power consumption.....	287
Table 8-3	RamLink signal descriptions.....	308
Table 8-4	RamLink command field encoding.....	314
Table 8-5	SyncLink command field encoding	327
Table 8-6	SyncLink byte ordering	330
Table 9-1	Chip-level comparison	340
Table 10-1	High-performance memory system comparison	360

Table 10-2	PC memory system comparison	360
Table 10-3	PC memory system access time comparison	360
Table 10-4	Portable system comparison.....	360
Table 11-1	DRAM market requirements.	435
Table 11-2	DRAM architecture's strengths and weaknesses.....	439

4

Memory System Design

This chapter is concerned with memory systems: how DRAMs are put together in a larger unit that responds to a CPU's read and write requests. The new DRAMs are interesting as individual circuits; more important, however, are the memory systems these new devices can form. The characteristics of these memory systems determine the utility and effectiveness of the new DRAM organizations.

4.1 New In The Second Edition

The changes to this chapter between the first and second editions include minor updates to reflect the changing structure of computer systems. In particular, many of the examples have been changed to reflect Pentium-based systems.

New descriptions of the terminology of latency, bandwidth, ranks, banks, and sub-banks have been added. Section 4.2.3 defined and explains the difference between peak, sustained and sustainable bandwidths and fill frequencies. Descriptions of the different burst transfer orders have also been added.

Section 4.8 also adds a description of Unified Memory Architecture (UMA) PCs and discusses the pros and cons of the different styles of UMA system.

4.2 Basic Concepts

What memory systems do

The role of a memory system is twofold:

- Reliable and consistent data storage
- Quick responses to the read and write requests of CPUs and I/O devices

Requests come in sizes ranging from a single byte to many words.¹ The definition of *reliable data storage* varies from system to system: in some systems, an occasional dropped bit is acceptable, while other systems require data loss rates at or below once per millennia even in the face of power outages. *Consistent data storage* refers to the memory system's behavior when dealing with reads and writes to the same address, potentially from different sources. For example, the CPU may perform a write to a particular location, followed shortly by a read of the same address. For the memory system to respond with anything other than the written value would present an inconsistent, and potentially fatal, view of memory.

Memory system design is primarily constrained by physical restrictions and bandwidth requirements. However, the DRAMs themselves account for a substantial portion of most systems' cost; this portion of system cost has been substantially beyond the designer's control. Memory system designers can only impact overall system cost somewhat, by selecting one variety of DRAM over another and by trading off the cost of the non-DRAM components with the memory system performance. Frequently, the memory system designer is constrained by the need to facilitate unnaturally small memory systems to minimize the price of the entry-level system.

4.2.1 Single In-Line Memory Modules (SIMMs)

DRAMs come in a variety of packages (see Section 5.2.1). In addition, memory vendors sell a wide variety of **Single In-Line Memory Modules (SIMMs)**. SIMMs, small printed circuit boards with up to 20 DRAMs mounted on them, have 30, 72, or more signals accessible via an edge connector. SIMMs are inserted into small connectors that are soldered onto a larger printed circuit board, or motherboard. Because SIMMs are socketed, they are inherently replaceable and upgradable.

1. The definition of a word varies across the industry. Most commonly, a word is defined as 16 or 32 bits. In this report, unless otherwise specified, a word is 32 bits long.

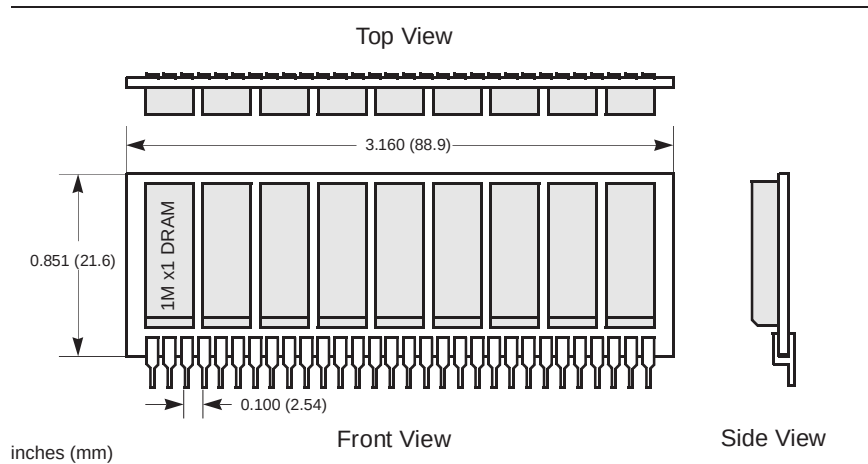


Figure 4-1 Standard 30-pin SIMMs are common in many low-end and midrange memory systems.

The most common SIMMs are $\times 8$ and $\times 9$ organizations on 3.5" long by 0.5 to 1.0" high boards, and $\times 32$ and $\times 36$ organizations on 4.25" long by 1.00–1.45" high boards (see Figure 4-1). Depths of 256K through 16M are widely available. Consequently, SIMMs come in all capacities from 1 to 40 Mbytes and more. They can have DRAMs mounted on one or both sides. The form factors and pinouts of the more common SIMMs are standardized, making them commodity parts.

Figures 4-2 through 4-4 show the internal structures of three SIMMs:

- A small, simple $\times 8$ part
- A $\times 36$ part with two banks and per-byte $\overline{\text{CAS}}$ signals
- A custom $\times 72$ buffered SIMM

The simplest SIMMs just forward the DRAM signals to the SIMM edge. SIMMs of intermediate complexity often logically partition the DRAMs into banks and slices by routing individual $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$ signals to groups of DRAMs. The $\times 36$ part shown in Figure 4-3 has two banks, activated by $\overline{\text{RAS0}}$, $\overline{\text{RAS2}}$ and $\overline{\text{RAS1}}$, $\overline{\text{RAS3}}$. The $\overline{\text{CAS}}$ signals are arranged to enable byte reads and writes. All the standardized 72-pin SIMMs contain speed and size information on four of the pins. This allows the memory controller to adjust its behavior to meet the requirements of the particular SIMM in each socket.

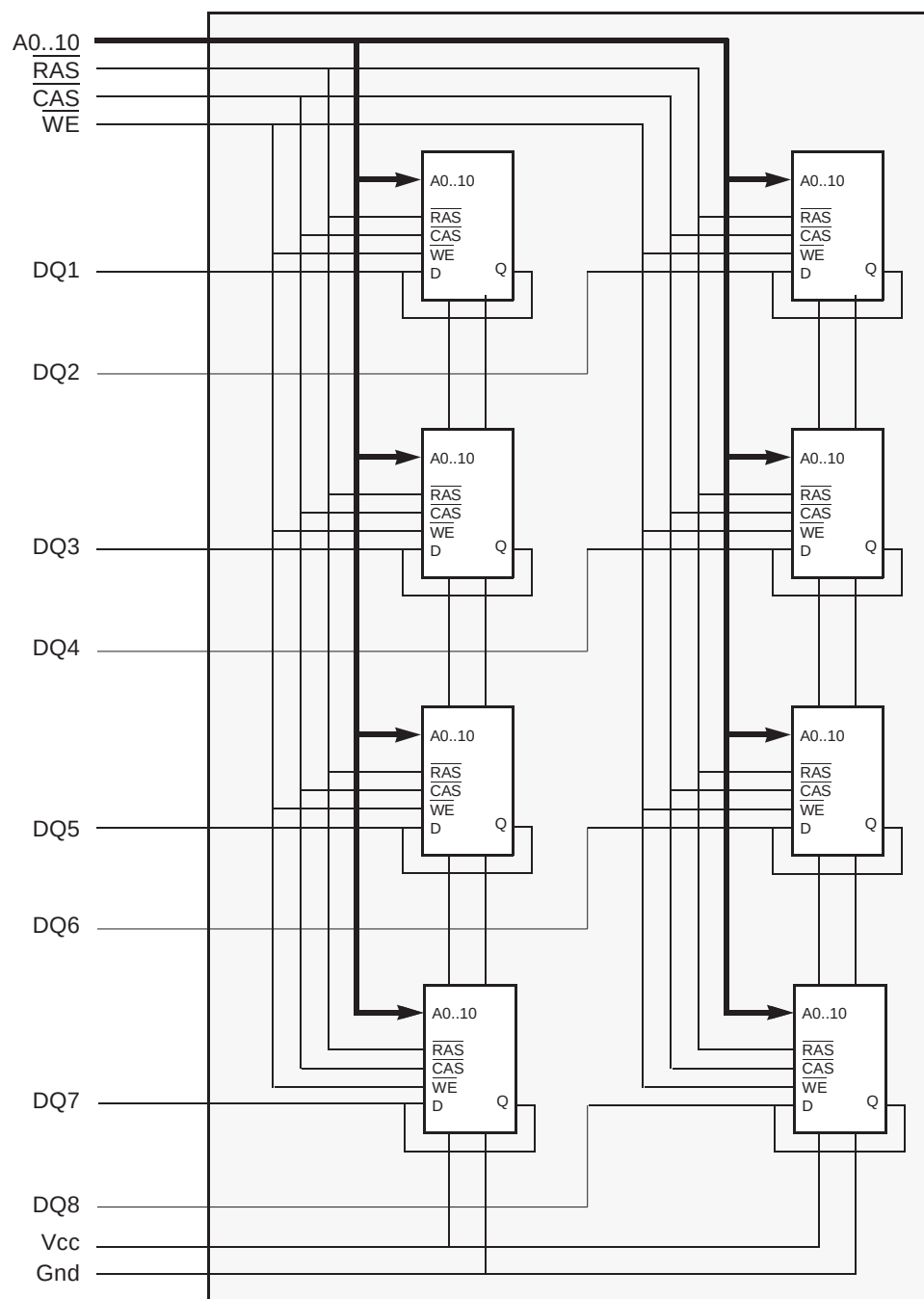


Figure 4-2 A 4-Mbyte byte-wide SIMM can be constructed from eight 4M x 1 DRAMs.

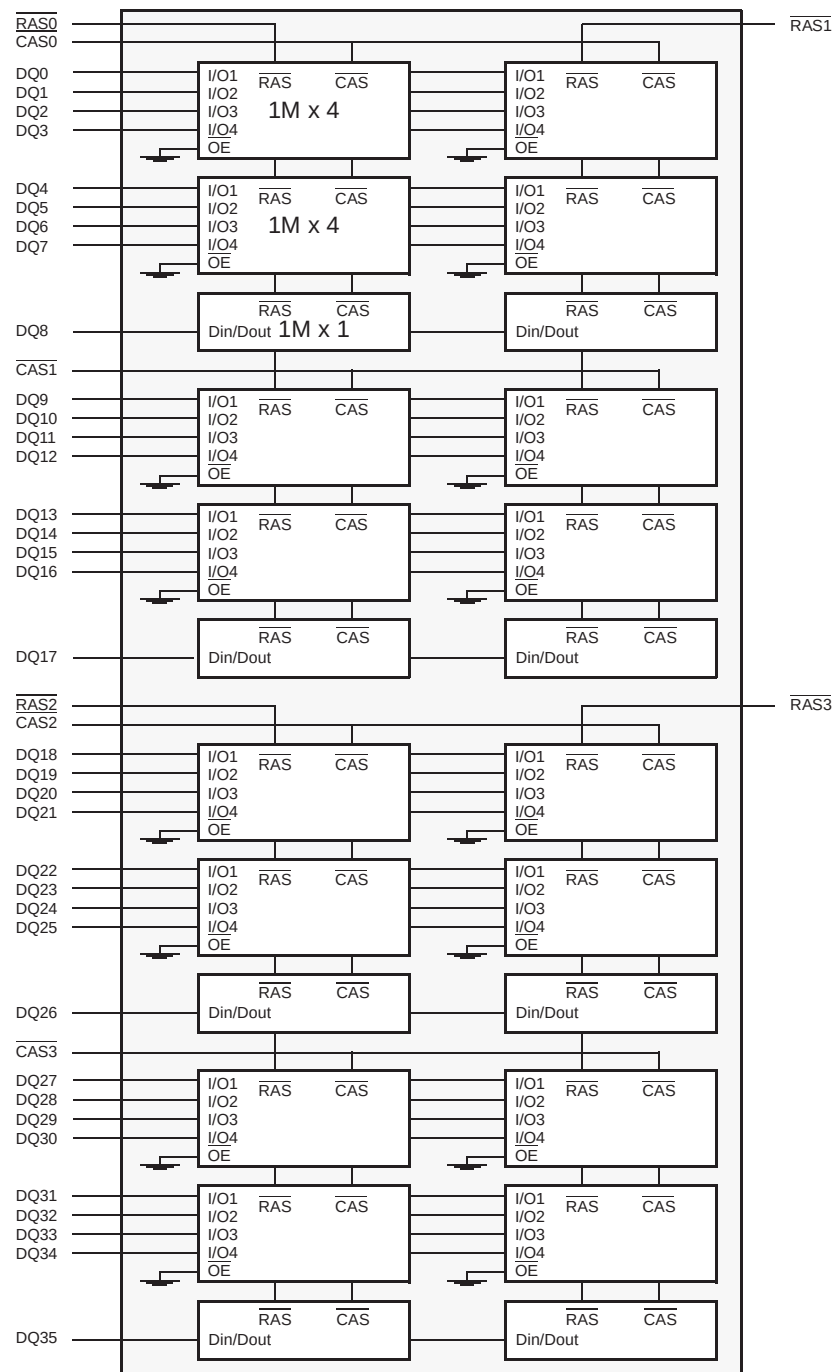


Figure 4-3 A 2M x 36 SIMM can be constructed from sixteen 4-Mbit DRAMs and eight 1-Mbit DRAMs, configured in two banks. The 10-bit address bus going to all twenty-four DRAMs is not shown.

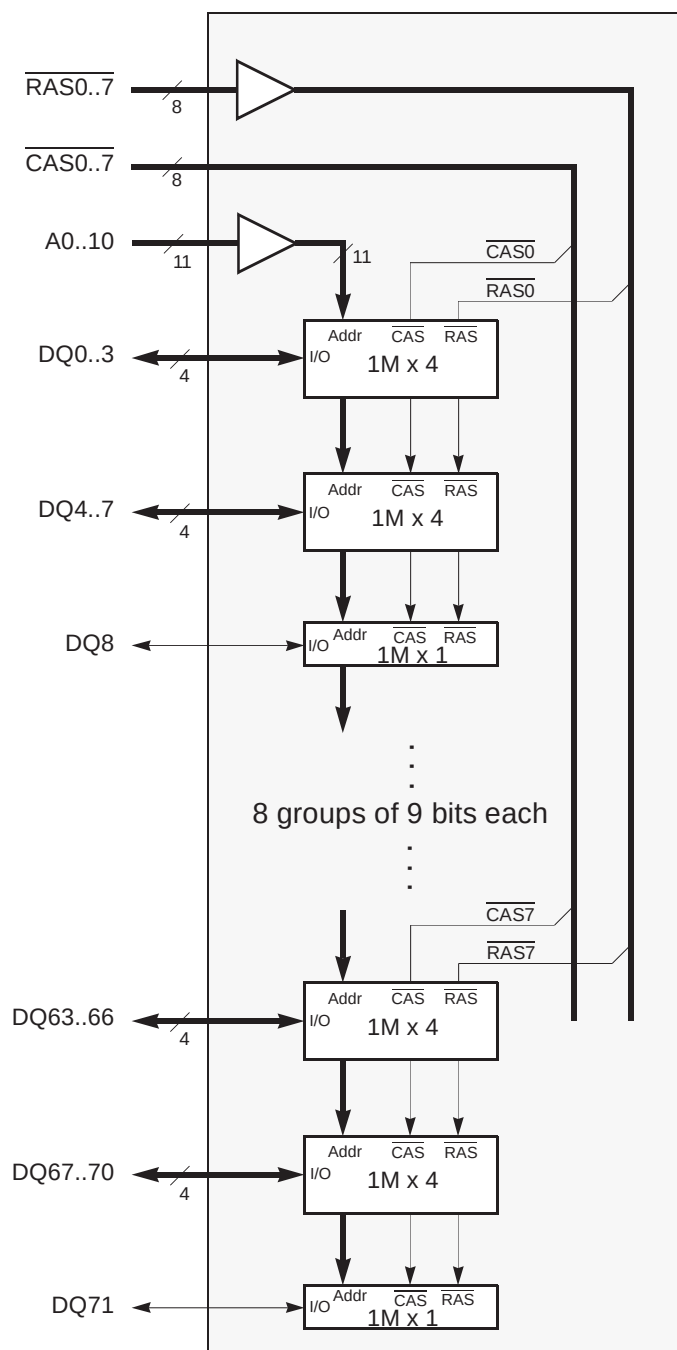


Figure 4-4 On this custom buffered x72 SIMM, the address lines and most of the control lines are locally regenerated.

SIMMs were conceived at a time when DRAMs and, consequently, memory systems, were slow. The sole goal of their developers was to facilitate cheap and physically dense memory expansion. The poor electrical characteristics of SIMMs and SIMM sockets are becoming severe impediments to the higher speed operation of memory systems. This problem will be further aggravated as the new generation of DRAMs, with their high frequency interfaces, become common. This problem may be partially alleviated by adding flow-through or registered buffers to SIMMs.

The buffered SIMM in Figure 4-4 has flow-through buffers on the address and control lines to limit the load presented by the SIMM to the memory controller. Limiting the load helps the memory system's overall performance by improving signal quality on the motherboard and decreasing the elapsed time between the memory controller outputs and the DRAM inputs. Adding buffers to the motherboard minimizes variability in the load placed on the primary control signals from the memory controller. Reduced variability in the load improves the signal quality of the motherboard signals and helps increase the frequency of reliable operation.

A number of the newer memory modules come in buffered and unbuffered versions. In many cases, the buffered versions include repeaters on all the control lines except the $\overline{\text{RAS}}$ lines. This is done because of the structure of PC memory systems (see section 4.4). In PCs, the $\overline{\text{RAS}}$ signals are used as bank-enables and so they are naturally low-fanout: if the system has only a few DRAMs per bank, buffers might actually slow the propagation of $\overline{\text{RAS}}$ to the DRAMs. On the other hand, $\overline{\text{CAS}}$, $\overline{\text{WE}}$, and the address lines are sent to all banks, and so they can have large fanouts in multibank systems. Buffering these signals improves the total time to propagate the control signals to the DRAMs.

For some of the new synchronous devices, registered memory modules are of value. These devices allow the total address and control transit time from the memory controller to the DRAMs to be split across two cycles. For large memory systems, this split can facilitate a high clock rate, and therefore a higher peak bandwidth; however, this is done at the expense of an additional cycle of latency for all operations.

In general, for all small- and medium-sized memory systems—including those found in PCs—unbuffered memory modules

typically will yield the highest operating frequency and provide the best overall performance.

New memory modules standardized

Recently, a set of new memory modules has been standardized. Instead of leads along only one side of the bottom edge, these memory modules have leads on both sides of the small printed circuit board. They are therefore called Dual In-Line Memory Modules (DIMMs). Although the widely used memory modules in the $\times 8$ through $\times 32$ widths are SIMMs, at the $\times 64$ width, all modules are DIMMs. Because the pins are on both sides of the module, the 168-pin DIMMs are only about 1 inch wider than the older $\times 32$ 72-pin SIMMs currently in use. The 168-pin $\times 64$ DIMMs are poised to become the next widely used memory module for PCs.

Another variety of DIMM achieving wide adoption is called Small-Outline Memory Modules (SOMMs). The density of contacts along the bottom edge of SOMMs is twice that of SIMMs and DIMMs. This greater density gives $\times 32$ and $\times 36$ SOMMs half the width of the traditional 72-pin SIMMs. SOMMs with DRAMs mounted on both sides are well suited to the tight packaging constraints in notebook computers and other portable applications.

4.2.2 Memory System Organization: Granularity and Expandability

Figure 4-5 is a block diagram of a typical midrange memory system. Such a system consists of a **memory controller** that 1) receives commands and addresses from the rest of the system and 2) directly or indirectly drives the address and control lines of the DRAMs. The data flowing to or from the DRAMs usually passes through bidirectional buffers that are also controlled by the memory controller. Alternatively, especially in high-performance systems, the data itself may pass through the controller on the way to or from the DRAMs.

The memory array itself is organized as a number of **banks** with the same widths but varying depths. A read operation begins when the memory controller receives a memory-read command and determines whether the address maps to one of its banks. If the bank being accessed currently has $\overline{\text{RAS}}$ held low, and if the row currently available within the DRAMs is the one that contains the requested address, the data can be quickly retrieved from the DRAMs via a CAS access. If the wrong row is

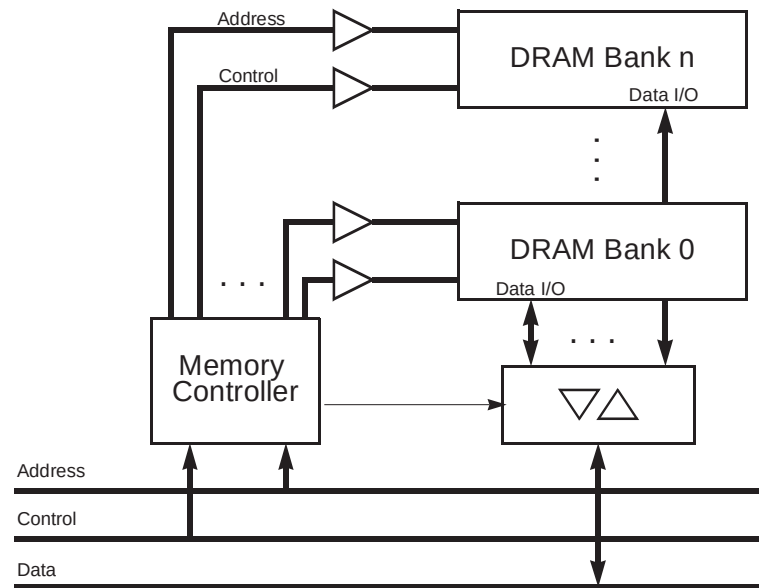


Figure 4-5 Typical memory system for a low-end to midrange computer system.

currently available, then $\overline{\text{RAS}}$ must be raised and kept high for a row precharge time before the access of the desired row can begin. Ultimately, as the data comes out of the DRAMs, the memory controller communicates to the processor that arrival of the data is imminent and directs the buffers to appropriately forward the requested data.

Writes happen similarly, except that the data to be written is usually received by the memory controller simultaneously with, or immediately after, the arrival of the address. This can be in advance of when the DRAMs want to see the data; as a result, the data is often temporarily stored or delayed in the memory controller or transceivers until the DRAMs are ready to perform the appropriate CAS write access.

Of banks and ranks

Until recently, all DRAMs logically consisted of a single homogeneous block of memory. Several of the new DRAM architectures are internally divided into banks. This has introduced a degree of confusion: the word *bank* can mean either a physical or a logical grouping of bits: groupings that can be different. To rectify this confusion, a new term, **rank**, has been introduced to refer to the physical structure.

Figure 4-6 illustrates the relationships among banks, ranks, DIMMs, and SIMMs. It shows a high-performance 64-bit memory system formed by two-way interleaving² DRAMs that have two banks internally. The DRAMs are mounted on DIMMs or SIMMs. In the figure, two different module configurations are shown: a $\times 32$ SIMM and a $\times 64$ DIMM.

A rank is defined as the collection of DRAMs that have to be present or not present together. A rank is made up of a linear array of DRAMs that are accessed and controlled in parallel. All DRAMs within a single rank are the same speed. Usually, though not always, all ranks under the control of a memory controller must be the same speed. In Figure 4-6, the memory system is interleaved, causing a rank to consist of a group of DRAMs on each side of the multiplexer. In this case, a rank is made up of a number of SIMMs or DIMMs.

A bank is the collection of bits that are accessed together to satisfy a block transaction. In Figure 4-6, a block is half the bits in one rank of DRAMs. A **sub-bank** is the collection of bits that are always accessed with the same temporal characteristics. In this case, a sub-bank is half of a bank because the memory system is two-way interleaved. A sub-bank is half the bits in the group of DRAMs—those on one side or the other of the multiplexer. When the banks of a memory system are logically divided into sub-banks, it is typical for the corresponding ranks to be physically divided into **sub-ranks**. In most cases, all of the sub-ranks in a rank must be present or not present together.

What is memory system granularity?

SIMMs and DIMMs come in a variety of interchangeable sizes. For example, a $\times 32$ SIMM constructed from two $1\text{M} \times 16$ DRAMs contains 4 Mbytes, while a $\times 32$ SIMM made with $4\text{M} \times 4$ DRAMs contains 16 Mbytes. Typically, memory controllers are able to control a wide variety of SIMMs and DIMMs. Consequently, a memory system constructed with memory modules can have a wide ranges of sizes depending on the number of ranks populated and the size of the SIMMs or DIMMs used. The **granularity** of a memory system is the amount of memory in a single rank using the smallest density SIMM, DIMM, or DRAM allowed by the memory controller. Since memory systems have an integral number of banks, the granularity of the memory system is also the smallest amount of memory that can be added to the machine.

2. See Section 4.2.4 for an explanation of interleaving.

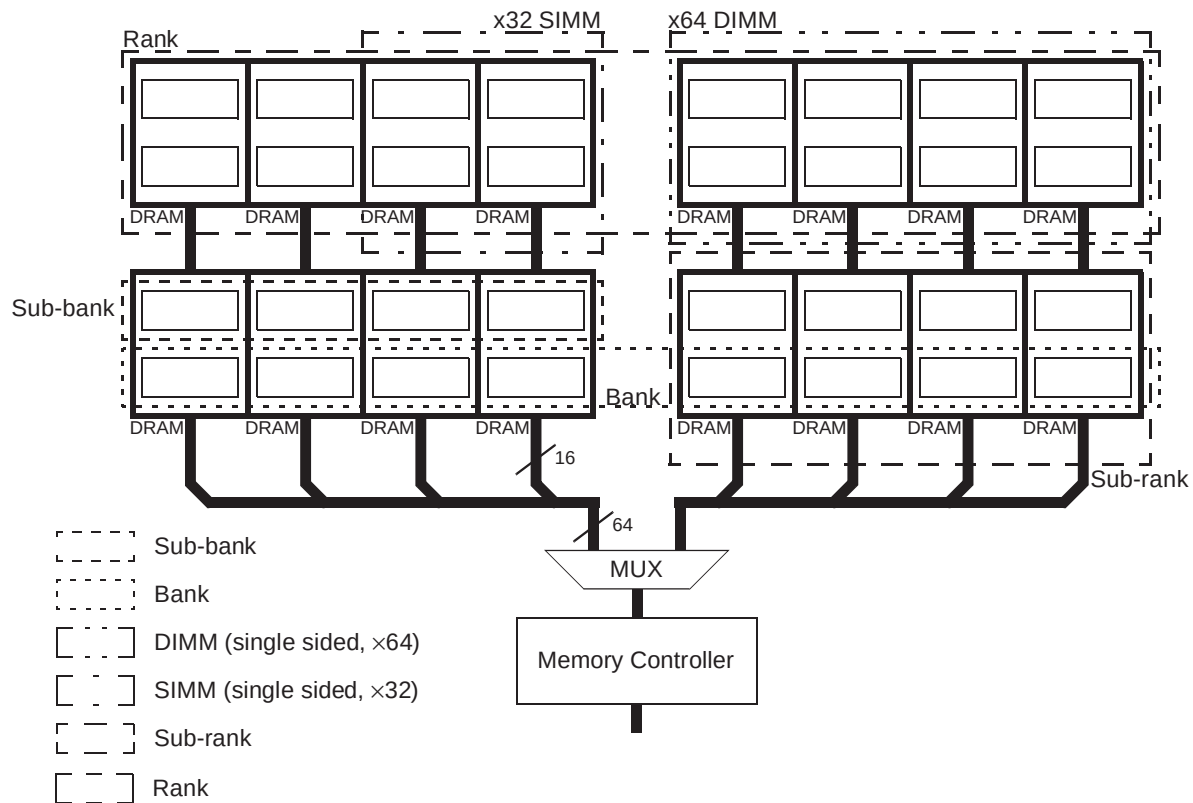


Figure 4-6 Complex memory systems are divided logically into banks and physically into ranks. DIMMs and SIMMs are ganged in parallel to form ranks, each of which may consist of one or more banks.

If the DRAMs used are not internally partitioned into distinct banks—in other words, if the DRAMs consist of a single homogeneous block of memory—then the ranks and banks are one and the same. In this case, the memory banks have the same depth as the DRAMs from which they are constructed. A 32-bit-wide memory bank made of eight $4\text{M} \times 4$ DRAMs has a depth of 4M and a width of 4 bytes, for a total of 16 Mbytes. At a given DRAM density, building a bank or rank out of fewer DRAMs that are wider and shallower makes for banks that are shallower and consequently smaller.

Another way to describe the granularity of the memory system is as the product of the memory system's width and the minimum controllable rank depth. For example, a 16-bit memory with a 256K minimum depth has a granularity of 512 Kbytes. In contrast, the granularity of a 64-bit memory built with $4\text{M} \times 4$ DRAMs is 32 Mbytes.

Since the depth of a memory rank is generally the same as the depth of the DRAMs it is made of, the granularity of a memory system is:

$$\text{Granularity} = \frac{\text{DRAM Size (bits)} \times \text{Memory Width (bytes)}}{\text{DRAM Width (bits)}}$$

Since DRAM and memory widths both increase slowly, the granularity is primarily related to DRAM density. Consequently, over the past two decades granularity has grown so much that it is often comparable to the minimum desirable memory size. If granularity grows beyond the minimum desirable memory size, then the price of an entry-level system will be inflated by the cost of the unnecessary but unavoidable extra memory.

Slowing the growth in granularity

In the face of ever rising DRAM densities, two ways exist to slow the growth in granularity:

1. Match every increase in DRAM size with a comparable increase in DRAM width.

The migration path from 4M×4 at the 16-Mbit level would be to 4M×16 at the 64-Mbit level and on to 4M×64 at the 256-Mbit level. This is practical in the short run, but soon becomes impractical as pin counts grow exponentially.

Although the desirable minimum memory size is also growing, increasing bandwidth requirements indicate a slow but steady growth in memory width. Consequently, the need to increase device width is undiminished by increasing memory system size.

2. Change the memory system organization and implementation so that the memory width is equal to the DRAM width.

Rambus DRAMs (RDRAMs) and **RamLink DRAMs**, the revolutionary alternatives discussed in Sections 8.2 and 8.3, take this approach. A high-bandwidth path is provided to every DRAM, and memory system granularity is reduced to the size of a single DRAM. In the simplest use of these interfaces, the DRAMs are not ganged in parallel. Conceptually, the memory system width is equal to the DRAM width. Though these solutions dramatically reduce granularity, memory system granularity is still linked to the exponentially growing DRAM size. Unfortunately, short of using an older generation of DRAMs or combining logic and memory on a single die, little can be done to reduce granularity below the size of a single DRAM.

4.2.3 Bandwidth and Latency: Peak, Sustained, and Sustainable

Improving bandwidth with block transfers

DRAMs share an important feature: once a row has been fetched from the storage cells into the sense amps, the entire row is available via column accesses. The sense amplifiers act as a cache capable of holding a row's worth of bits.

Caches take advantage of the **spatial locality of reference** in programs by fetching and storing a number of contiguous memory locations. For example, when the program executes instruction i , the cache also fetches instructions $i+1$, $i+2$, and $i+3$. This is worthwhile because of the significant likelihood that these instructions will be executed next.

Consequently, in most computer systems today, the vast majority of the read references to main memory are block requests. Typically, 16 or 32 bytes are fetched at a time—sometimes less in older or low-end systems, sometimes more in specialized or very high-performance systems. In Intel 486 systems, 16-byte transfers and 4-byte-wide memory systems are the norm. In such systems, each memory access becomes a row access followed by four column accesses. These accesses deliver the 16 bytes of data to the CPU in four 4-byte chunks in rapid succession (see Figure 4-7). In Pentium® systems, the processor transfers 32 bytes at a time over a 64-bit bus. Again, the transfers occur in four consecutive chunks.

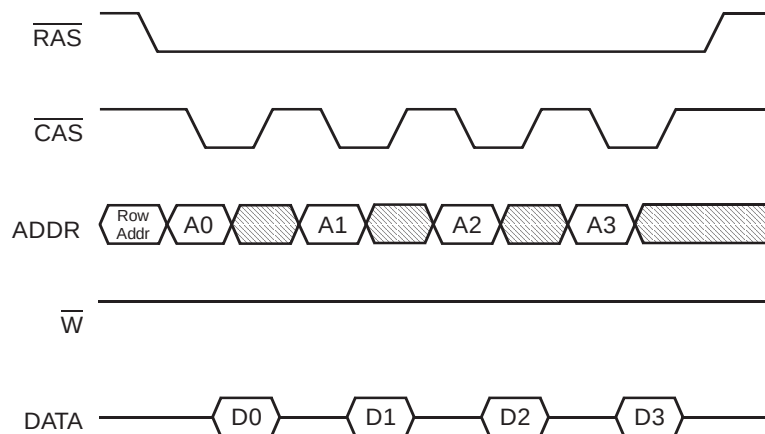


Figure 4-7 A burst of four words consists of four column accesses.

Two metrics of performance

Two primary metrics are used to evaluate DRAMs and memory systems: **bandwidth** and **latency**. Latency is the amount of time between the start of a transaction—typically a read transaction—and the return of the first piece of data. From the DRAM's perspective, if the transaction is a column read access, then the latency is the column access time, and if the transaction needs a row access, then the latency is the row access time. From the processor's perspective, though, the latency starts when the address of the request leaves the CPU. From there it has to travel to the memory controller where it is decoded and multiplexed. Then the address and the appropriate control signals are transmitted to the DRAMs. Once the DRAM outputs the first piece of required data, it must flow to and through the memory controller and then on to the CPU. The temporal overhead in getting the address to the DRAMs and the data back to the CPU can add many cycles of latency to the basic DRAM access time.

Bandwidth is the rate at which data is transferred. In fact, a number of different bandwidths together describe the characteristics of a DRAM and a memory system. A DRAM's **peak bandwidth** is the maximum rate at which data can be transferred into or out of a DRAM; constraints imposed by the memory system are not taken into consideration in calculating this metric. For a single DRAM, the peak bandwidth is equal to the width of the DRAM divided by the minimum column cycle time. A memory system's peak bandwidth is the width of the memory bus divided by the actual column cycle time. The actual column cycle time is typically greater than the DRAM's data sheet column cycle time because of design realities: signal propagation delays, synchronization delays, and the overhead of cycle-based controller design.

Actual **sustained bandwidth** in a system is typically much lower than the peak bandwidth. Several factors account for the gap. First, both row and column accesses have latencies during which no data is transferred. Also, in a real system, the overhead associated with transferring information to and from the processor—and often additional overhead within the processor—separates one memory transaction from the next. This time during which no data is being transferred decreases the actual bandwidth of the memory system further below the peak bandwidth. The sustained bandwidth obtained is highly dependent on the workload: the applications being executed. This is because the application determines the mix of row and column accesses and the amount of time between primary and secondary cache misses.

Sustainable bandwidth defined

Because the actual sustained bandwidth depends on so many factors outside the memory system (such as the application, processor, compiler, operating system, and system architecture), the sustained bandwidth is not a convenient or reliable metric for comparing DRAMs or memory systems. Instead, this report defines a new metric: the **sustainable bandwidth**. Like the sustained bandwidth, the sustainable bandwidth is more representative of DRAM and memory system performance in a system than is the peak bandwidth. However, unlike the sustained bandwidth, the sustainable bandwidth is only dependent on the memory organization, DRAM characteristics, and the relative frequency of row and column accesses.

The sustainable bandwidth of a memory system is the bandwidth deliverable by a memory system assuming an aggressive controller and an infinite supply of memory references queued up for processing. The sustainable bandwidth is calculated assuming that the memory controller can take maximum advantage of any unique features of the DRAM.

The only remaining external parameter is the mix of row and column accesses. This is a function of the application and of the DRAM's row-cache characteristics. The **worst case sustainable bandwidth** is the sustainable bandwidth assuming a pathological reference mix that forces 100% row accesses. The **best case sustainable bandwidth** assumes a trivial reference stream that is satisfied with 100% column accesses to a random mix of banks. For most DRAM organizations, the worst case scenario involves a sequence of references to a single bank. In contrast, the best case scenario typically involves references spread across multiple banks. **Good case** and **bad case sustainable bandwidth** refer to the bandwidths attainable with row column access frequencies that are on the good and bad ends, respectively, of a realistic range of row-cache miss ratios. The range is chosen to be representative of the row-cache behavior that would be encountered in a real system using the specific DRAM and memory organization.

Peak versus sustainable fill frequencies

As defined in Chapter 4, the fill frequency of a DRAM or a memory system is the ratio of an organization's bandwidth to its size. As was just noted, the peak, sustained, and sustainable bandwidths of DRAMs and memory systems vary substantially. Consequently, the peak and sustainable fill frequencies of systems and devices also differ substantially. The sustainable fill frequencies are more meaningful than the peak fill frequencies. However, sustainable fill frequencies are more difficult to calculate properly because they are dependent on all the important

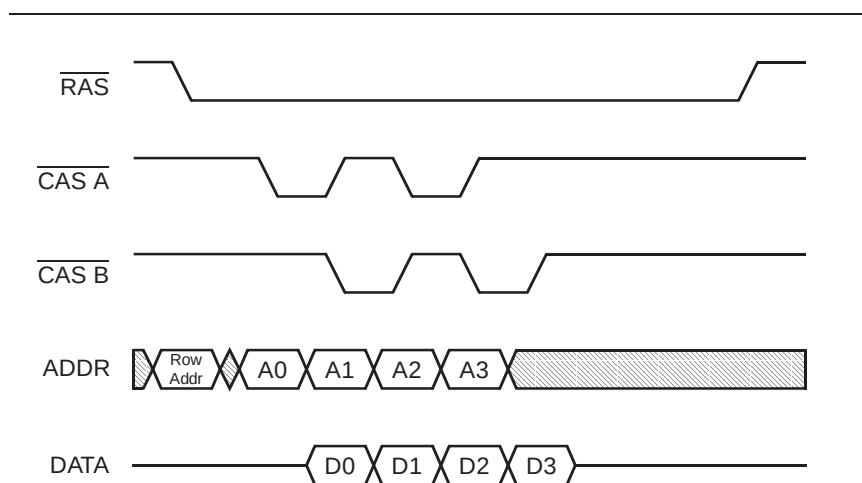


Figure 4-8 Interleaving uses two or more sub-banks to improve bandwidth.

characteristics of the DRAM and memory system—not just the peak column access cycle rate. Note that since the sustainable bandwidth is a range of bandwidths dependent on the mix of row and column accesses (i.e., the row-cache miss ratio), the **sustainable fill frequency** is also a corresponding range of fill frequencies based on the same access type mix.

4.2.4 Performance and Bandwidth Considerations

Interleaving overlaps CAS precharge time

In Figure 4-7, each access is handled entirely within one bank, even if the memory system contains more than one bank. The other banks are idle for the duration of the access. An alternative is to have more than one bank participate in each transaction. Figure 4-8 shows the same 32-byte access handled cooperatively by two banks. Both banks perform RAS accesses, then each bank does two alternating column accesses. This is potentially faster than the design shown in Figure 4-7 because each bank's CAS precharge time is in effect used by the other bank to transfer data. This alternating of two or more banks is called **interleaving**.

Unfortunately, the industry is inconsistent with respect to the terminology of interleaving. For example, Figure 4-9 shows a 32-bit memory system with four SIMMs. Each horizontal pair of SIMMs cooperates to satisfy accesses. This organization can be

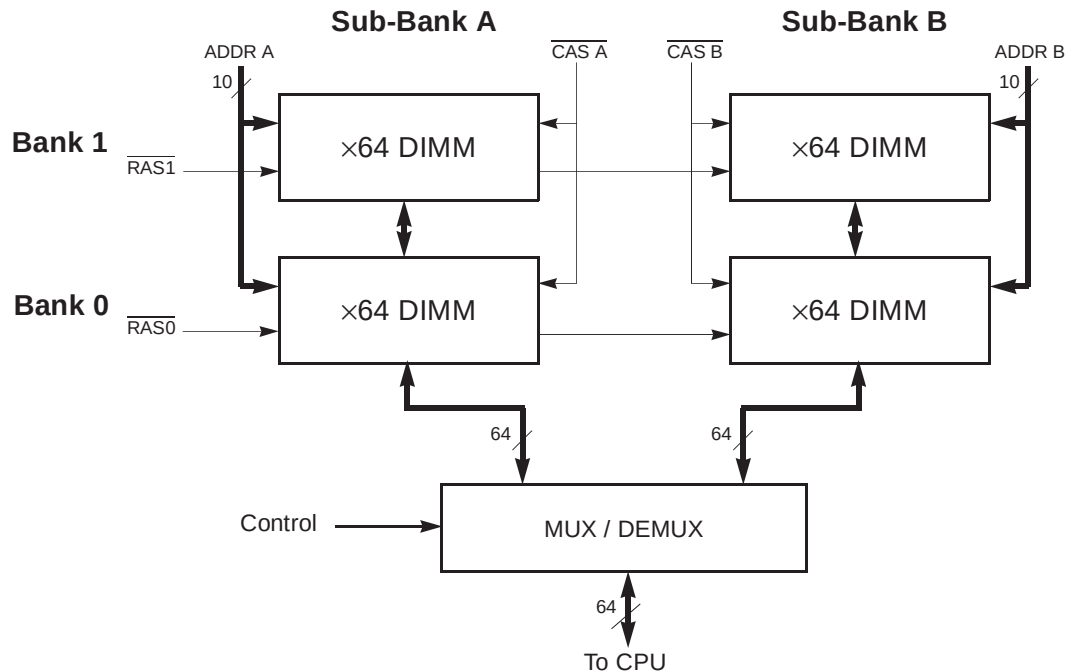


Figure 4-9 This interleaved memory system has two banks. Each bank has two sub-banks.

described as either four banks that are two-way interleaved, or two banks that are each two-way interleaved internally. A two-way interleaved bank is made up of two **sub-banks** that work together to provide data more quickly than either could do individually. The latter description is more consistent with the definition of a bank as the chunk of memory that is accessed for each memory reference; in the remainder of this report, interleaved memory systems are described in terms of banks and sub-banks.

Figure 4-10 shows the partitioning of the physical address bits in a two-way interleaved memory system. The highest order bits specify the bank being accessed; the next highest order bits specify the row address within the bank. Then come the column address bits, which are common to all interleaves. The least significant three bits indicate the interleave and byte lane of the referenced byte. For a 16-byte read access, the low-order four bits of the address would be zero, and the memory controller would have to increment the column address between the first and second column accesses to each half of the bank.

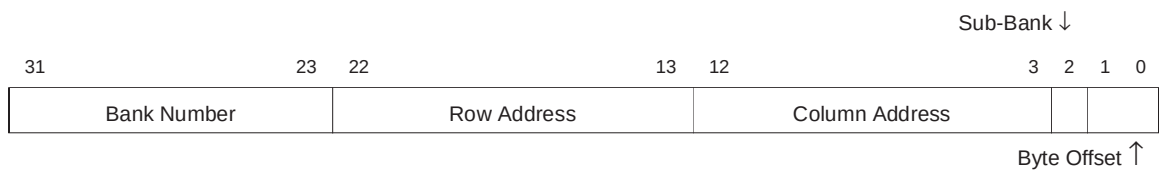


Figure 4-10 Interleaved memory systems use low-order address bits to locate words in sub-banks.

How interleaved and noninterleaved systems differ

Interleaved memory systems are primarily differentiated from noninterleaved systems by their temporally staggered column accesses. Unfortunately, although an n -way interleaved memory system has higher bandwidth than a noninterleaved system, the granularity of a memory system is increased by a factor of n . Another cost of interleaving results from the need to provide a separate address bus and control signals for each sub-bank. In high-speed interleaved memory systems, each sub-bank has its own data bus as well. The net result is that interleaving memory controllers have higher pin-counts than their generic counterparts. In lower speed systems, the sub-banks can share the same data bus. In this case the DRAMs' output-enable signals are used to switch quickly between the sub-banks. However, if the data bus is shared among the sub-banks, an interleaved memory controller can have fewer pins than a wide, noninterleaved controller of comparable bandwidth—if the shared data bus data replaces a wider nonmultiplexed bus.

Aggressive use of page mode

Memory systems can take also advantage of the spatial locality of references by aggressively using page-mode or static-column-mode DRAMs. $\overline{\text{RAS}}$ need not be raised after a memory request is satisfied. If $\overline{\text{RAS}}$ is held low between memory requests, and if the next memory request is to the same row, then the needed data can be retrieved with one or more column accesses.

Unfortunately, if $\overline{\text{RAS}}$ is low when the next memory reference arrives, and the request is to a different row, then the second access is delayed. This delay occurs because $\overline{\text{RAS}}$ must first be raised and held high for the RAS precharge time before the new row access can begin. Using page mode between accesses is therefore a gamble. If the frequency of same-row, back-to-back accesses is high enough, more time is saved than is wasted when the row address is different. The net effect is that the

mean memory access time is reduced. Generally, aggressive use of page mode is good policy only if there are several banks because instruction and data references are less likely to repeatedly interfere with one another.

Row sense amplifiers as a cache

One way to view the aggressive use of page mode is to think of the sense amps, taken together, as a **cache**: as long as $\overline{\text{RAS}}$ is held low, a portion of main memory is available more quickly than the rest. This **row cache** is a direct-mapped cache in that each location in main memory can be copied into exactly one sense amplifier. The block size of the cache is equal to the size of the memory system row: that is, the row size of each DRAM multiplied by the number of DRAMs accessed in parallel. The number of blocks in this cache is equal to the number of banks held with $\overline{\text{RAS}}$ low. For example, a 16-Mbyte memory system constructed of four $1\text{M} \times 32$ SIMMs has a row cache of 16 Kbytes, consisting of four blocks of 4 Kbytes each. In comparison to current CPU caches, this is a fairly small cache with an abnormally large block size.

Row caches exploit the spatial locality of reference in the reference stream reaching main memory. In the absence of a CPU cache, a row cache with a small number of blocks can work very well because there is a great deal of spatial locality within the instruction and data reference streams. The presence of primary and secondary CPU caches reduces the spatial locality of the reference stream reaching main memory. However, in many cases, enough of the reference stream's spatial locality remains to allow moderate, though not dramatic, improvements in system performance if page mode is used aggressively. Interestingly, if the CPU caches are write-through, a great deal more locality of reference exists in the write stream reaching main memory than in the read stream. In these circumstances, it makes sense to keep the DRAMs in page mode after a write, based on the expectation of another write to a similar address soon, but to take them out of page mode after a read.

In many respects, row caches behave similarly to CPU caches. In both CPU and row caches the miss rate is primarily determined by four factors, listed here in descending order of importance:

- Program characteristics
- Cache size
- Block size
- Set associativity

The primary benefit of increased set associativity is not so much a decrease in the miss rate as a reduction in the frequency of pathological conflicts that occasionally cause periods of abnormally high cache miss rates.

Row caches differ from CPU caches in two important ways:

- The block sizes that minimize the miss ratio are significantly higher than in CPU caches.
- The row-cache block size that maximizes performance is the same as the block size that minimizes the row-cache miss ratio.

Because CPU cache blocks are filled serially over several cycles instead of all at once—as row-cache blocks are—the CPU cache block size that maximizes performance is always smaller than the CPU cache block size that minimizes the miss ratio.

Linear and sub-block ordered bursts

There are two different orders in which data is returned to the CPU from the memory system. Unfortunately, each goes by several names. With the first transfer order—called linear or sequential burst—the data returned to the CPU starts with the addressed word, continues linearly to the end of the block, and wraps around to the beginning of the block. With the second transfer order—called sub-block order, interleaved, or Intel order—data transfer begins with the addressed word, and then proceeds with increasingly distant words until the entire block is transferred. The order that data is transferred in for the two types of bursts is shown in Table 4-1.

Start Address	Linear Ordered Bursts				Sub-block ordered Bursts			
	Burst of 1	Burst of 2	Burst of 4	Burst of 8	Burst of 1	Burst of 2	Burst of 4	Burst of 8
0	0	0-1	0-1-2-3	0-1-2-3-4-5-6-7	0	0-1	0-1-2-3	0-1-2-3-4-5-6-7
1	1	1-0	1-2-3-0	1-2-3-4-5-6-7-0	1	1-0	1-0-3-2	1-0-3-2-5-4-7-6
2			2-3-0-1	2-3-4-5-6-7-0-1			2-3-0-1	2-3-0-1-6-7-4-5
3			3-0-1-2	3-4-5-6-7-0-1-2			3-2-1-0	3-2-1-0-7-6-5-4
4				4-5-6-7-0-1-2-3				4-5-6-7-0-1-2-3
5				5-6-7-0-1-2-3-4				5-4-7-6-1-0-3-2
6				6-7-0-1-2-3-4-5				6-7-4-5-2-3-0-1
7				7-0-1-2-3-4-5-6				7-6-5-4-3-2-1-0

Table 4-1 Linear and sub-block ordered burst are identical for short burst and bursts that start at the start of a block.

The order of the data returned in the sub-block-ordered case seems almost random. However, this data order actually exhibits a number of important properties. First, the address the n th piece of data is obtained by doing the exclusive-OR of n and the starting address of the transfer. Consequently, the order does depend on the length of the burst. For example, bursts of length four are the first part of the bursts of length eight. This makes it easier to abort and subsequently restart long transfers. Second, since the first two pieces of data transferred are always an odd and even pair of words, it is much easier to build DRAMs and memory systems that are half-speed and double-width internally. Sub-block ordering has been used in all Intel processors since the 486 and in some workstations as well.

It is important to note that the two burst orders are identical for short bursts and for all bursts with a zero start address. Overall, the choice of burst type is a matter of convenience and politics, and not a matter of great significance from a performance perspective.

4.2.5 Parity and Error Correction Codes (ECC)

Generally, computer memory systems are supposed to faithfully retain the information entrusted to them. In many applications, even a small chance of losing a single bit is unacceptable. Given that DRAMs occasionally lose information to alpha particles and other transient errors, memory system engineers add some redundancy in order to reduce the probability of bad data being used accidentally.

In addition to the straightforward cost of additional memory bits that must be purchased to provide parity and error protection, hidden costs stem from the additional logic and time in the memory controller and/or processor needed to set and check the redundant bits. More significantly, when the parity or error correction code is formed over 16-, 32-, or 64-bit words, then memory writes that are smaller than the word size must be performed as read-modify-write cycles so that all the redundant bits reflect the changed portion of a memory word. This naturally takes longer than a simple write operation.

Two factors affect the acceptable level of error protection in a computer system: the probability of an error and the likely result of an error. When a computer gets the wrong answer or crashes due to a bad bit, the resulting problems depend greatly on the application. The cost of a flatly wrong answer is

potentially much greater in a computer controlling a nuclear reactor, flying a space shuttle, or shuffling money around than in one running a video game. The probability of an error also influences the acceptable level of protection. An expected bit failure rate of one per 10,000 years is probably acceptable in almost any application, while one error per day is probably unacceptable even to the most casual of PC users.

Unprotected memory systems are increasingly common

Three levels of protection from transient errors are commonly used:

No Protection:

In the extreme, if no redundancy is added to the memory system, any error that does occur is undetectable. The possible consequences of using a bad instruction or datum range from no discernible effect, through an application or system crash, to an unsuspected wrong answer.

Parity Protection:

The addition of a single parity bit per word of memory allows the detection of any single-bit error within each word. However, correcting the error once it has been detected is impossible since all single-bit errors within a word exhibit the same symptom.

Single-Error Correcting, Double-Error Detecting (SECDED) Codes:

By adding seven redundant bits per 32-bit word, or eight redundant bits per 64-bit word, it is possible to detect and correct any single-bit error to a word and to detect, but not correct, any two-bit error to a single word.

Given the steady and dramatic decline in soft error rates in DRAMs over time (see Section 3.5.1), unprotected memory systems are becoming more and more acceptable for a wide variety of non-mission critical applications and even those that are mission critical. The limited safeguard afforded by parity protection is becoming irrelevant: either data integrity is sufficiently important to warrant error correction, or the soft error rate is low enough that the system failure rate due to memory errors is well below the failure rate due to other causes. Already, the parity protection that is standard in PCs is not primarily protecting the user from soft errors in the DRAMs. More often, parity protects the user from other sources of intermittent memory failures. These sources include user-installed memory that is fractionally too slow or is poorly seated in its

socket and overly aggressive motherboard designs that intermittently and inadvertently run DRAMs or other memory system components beyond their rated specifications.

4.2.6 Refresh

DRAMs must have their contents refreshed periodically. Almost all DRAMs must receive a refresh operation on average every 15 μ s to ensure that all information is retained under worst case operating conditions. Section 3.3.1 describes the two primary types of refresh cycles offered by generic DRAMs.

In deciding how DRAMs are to be refreshed, the memory system designer addresses the following questions:

- How should available bandwidth be divided between refresh and normal operations?
- Which type of refresh operation should be used?
- Can the processor generate a memory reference while a refresh operation is taking place? If so, how is this handled?

The possible responses to these questions involve choosing one of two basic approaches to DRAM refresh operations:

- Static allocation of a fixed proportion of the memory bandwidth to refresh. Memory references are multiplexed with refresh operations according to the fixed allocation. This strategy results in a fixed but nonminimum memory response time.
- Dynamic allocation of all memory bandwidth to either memory references or refresh operations as required. Such organizations have greater average bandwidth available to the processor at the expense of decreased predictability. The response time for a memory reference will be longer than usual if it arrives at the memory subsystem while a refresh operation is occurring.

To minimize the amount of memory bandwidth used for refresh, it is common to refresh all banks within a memory subsystem simultaneously. Unlike a normal memory reference in which only one bank performs a RAS operation at a time, in a multi-bank refresh all banks perform RAS operations together. To avoid an unacceptable amount of power supply noise generated by a large number of DRAMs refreshing simultaneously, it is common to slightly stagger the refresh operations of all banks

in the system. The drawback is that when staggered, an entire refresh operation takes somewhat longer than if all banks are refreshed in synchrony. However, the combined benefits of reliable operation and reduced demand on the power supply greatly outweigh this drawback.

4.3 High-Speed Design Issues

An inevitable consequence of the constant improvement in IC fabrication technologies is an increase in the maximum operating frequency of ICs. As intrachip speeds and bandwidths improve, there is constant pressure to increase the speed and bandwidth of the interchip connections. However, the physical laws that govern the maximum rate of signal and information flow come into play much more forcefully in the interchip case because of the longer distances to be traversed. This section highlights some of the impediments and fundamental limitations that come into play when high-speed DRAMs are combined to form high-performance memory systems.³

4.3.1 High-Speed Signals and Terminated Transmission Lines

The speed of an electrical signal traveling down a wire is determined by the physical properties of the wire and its surrounding material. For traces on PC boards, the propagation delay⁴ of a signal traveling down a wire is between 100 and 500 ps/in, depending on the physical dimensions of the trace and its neighbors, the printed circuit board material, and the number and characteristics of loads on the trace (ICs, connectors, etcetera).

Electrical signals have rise and fall times: the time the signal takes to change from the voltage representing one logic level to that representing the other.⁵ When the time necessary for a signal to traverse a wire is much shorter than the rise time of the signal, then the wire and all the components along it act as a

3. A more complete, yet thoroughly approachable, presentation of the complexities of high-speed design can be found in H. Johnson and M. Graham, *High-Speed Digital Design: A Handbook of Black Magic*, Prentice Hall, 1993.
4. It is convenient to express the speed of signal propagation as a delay per unit length. A delay of 100 ps/in is the same as a speed of 256×10^6 m/s.
5. Rise and fall times are generally defined as the elapsed time from the 10 to 90% points: that is, from the point in time that the signal has already made 10% of the total transition from the voltage representing one logic level to the voltage representing the other logic level, to the point in time when it has completed 90% of the transition.

single lumped (capacitive) load. This is the normal situation in low-speed systems with slowly changing signals. The lumped capacitive load is easily characterized, and signals are well behaved in that transitions are smooth and predictable.

Any of three circumstances can invalidate the basic assumption of long rise times relative to signal propagation delays:

- Traces that are longer than a few inches
- Traces that have unusually slow propagation delays due to heavy loading or unusual construction
- Signals that transition in less than a few nanoseconds

The total propagation delay of a signal from point A to point B is related to both the transition time and the propagation delay on the wire proper. As systems have become faster, there is an increasing need to cut the total propagation delay of interchip communication by pushing rise and fall times to ever lower levels—levels that cause the lumped model of wires to break down.

When the transition time is less than the propagation delay on a wire, the wire behaves as a transmission line. Signals enter the transmission line at a driver and propagate down the wire as a wave front. An appropriate physical analogy is of a long, taut string that is plucked close to one end. The wave caused by the plucking travels down the string; if the far end is held firmly, the wave reflects back to the source. Voltage waves reflect off the ends of wires in an analogous fashion. The magnitude and polarity of the reflected wave depend on the characteristics of the load at the ends of wire relative to the electrical characteristics of the wire itself. In the simplest case, the voltage wave will bounce off the two ends of the wire in ever smaller amounts until a steady state value is reached.

Reflections can originate not only at the ends of the wire but also wherever there is an electrical load on the wire or a change in the wire's physical (and corresponding electrical) properties. Examples of electrical discontinuities that can cause reflections include connectors, ICs, vias from one level of a multilayer PCB to another, and, perhaps most importantly, stubs. Stubs are short sections of wire or trace branching off from a main trace to one or more chips. If the stub is short, it is effectively invisible to the voltage wave, and the wavefront behaves as if the loads were on the main trace. However, if the stub is too long, it becomes a source of muted reflections for signals traveling down the main trace.

**Signal reflections
cause ringing and
delay**

Reflections, though not inherently bad, become a problem when they interfere with one another enough to degrade signal quality or to delay the reception of a signal at its intended destination. Figure 5-2 on page 138 illustrates the ringing that can be caused by interference between reflected wavefronts. Ringing not only delays the point at which the signal can be reliably interpreted at the destination, but, when excessive, can physically harm the input electronics.

One way that reflections can be eliminated is by placing an appropriately sized resistor at the end of the trace. This is called parallel, or end, termination of the trace. When the termination resistor matches the characteristic impedance (i.e. resistance) of the trace, then the magnitude of the reflection vanishes to zero. In this case, the effective signal propagation delay is determined by the characteristics of the trace and the length of the wire. Calculation of the delay is not complicated by the reflections. Reflections from all the intermediate discontinuities along the trace are minimized or eliminated by:

- Making the trace's electrical characteristics as uniform as possible along its entire length
- Concentrating the load at the terminated end of the trace or distributing the load evenly along the trace's length

Another form of termination is source termination. In this scheme, a resistor equal to the characteristic impedance of the trace is placed between the driver and the transmission line. The far end of the trace is left unterminated. When the driver switches state, a half-amplitude wave-front propagates to the unterminated far end of the trace. There it reflects back toward the driver and, in the process, the signal attains its full amplitude. When the reflected wave reaches the driver, it is absorbed by the source terminating resistor, and no further reflections occur.

For single-source, single-destination traces, both source and parallel termination are acceptable. In both cases, the receiver at the far end of the trace sees a full amplitude signal in minimum time. Parallel termination causes some additional power to be dissipated in the termination resistor. The magnitude of this dissipation depends on 1) the termination voltage, 2) the voltage levels representing logic 0 and logic 1, and 3) the characteristic impedance of the trace.

Source termination also involves a complication. Receivers that are not at the far end of the trace from the driver first see a half-amplitude wave traveling toward the far end. Later on, after the incident wave has traveled to the unterminated far end and reflected back, their input voltage completes its transition. The half-voltage state may be insufficient to be recognized as a transition, and so the effective time to communicate with these intermediate receivers is longer than with parallel termination. In some cases, this half-transitioned level can cause excessive power to be dissipated in the receiver's input buffer.

The use of **terminated transmission lines**—uniform and uniformly loaded traces terminated through an appropriate resistor to either the driver or a termination supply—is a fundamental prerequisite to the design of very high-speed systems.

Bidirectional buses pose special challenges

This discussion has focused on the case of a wire driven at one end toward one or more loads at the other end. However, bidirectional buses, with many potential sources and destinations of information, are common in computer systems. These buses pose special challenges because they are typically:

- Physically long
- Heavily and unevenly loaded
- Critical to high-speed operation

Because drivers can be anywhere along the traces in bidirectional buses, signals propagate in both directions. If reflections are to be prevented, the bus must be terminated at both ends. Consequently, the DC resistance of the bus is equal to half the characteristic impedance of the trace, and the power needed to drive the bus is doubled. Source termination is inappropriate for multisource buses.

In non-transmission line environments, the total capacitive load on a bus often limits the speed of operation. The bus can sometimes be effectively partitioned through the use of low-resistance NMOS transistors. These transistors selectively connect or disconnect major portions of the overall net when specific communication needs warrant doing so. If the maximum load connected to the driver at one time is dramatically reduced, the time needed to transition the bus from one logic state to the other is also reduced. These NMOS transistors can also be used with high-current TTL drivers to construct 3:1 address

multiplexers that drive the heavy loads found in memory arrays faster than any monolithic TTL multiplexers.

4.3.2 Clock and Data Skew Issues

Unfortunately, getting high-quality electrical signals from point A to point B is only part of the problem. Designers of high-speed systems must also cope with the issue of elapsed time.

Clocks are among a system's most important signals. Clocks define the start and stop of each machine cycle; consequently, the beginning and end of all of interchip communication is defined relative to one or more clock edges. However, like all electrical signals, clocks have rise and fall times and propagation delays. If the clock that defines the start of a cycle does not transition at all ICs at the same time, then the time available within a cycle for communication between ICs is compromised. The maximum difference in time between the arrival of a clock at different parts of a system is called **clock skew**, and various techniques are employed to minimize it:

- Distributing multiple copies of a clock so that one signal does not have to reach all ICs on a board or in a system
- Ensuring that all clock traces are the same length and similarly loaded, thus minimizing differences in the clock propagation delays to all parts of the system
- Using **Phase-Locked Loops (PLLs)** in ICs to minimize the delay from a clock transition at an IC's input pin to the clock transition throughout the chip

Skew can exist between data signals as well. Buses normally carry signals in parallel: the bits traveling down the bus together are temporally aligned—that is, sent and received at the same time. However, if one or more of the traces is longer or more heavily loaded than the rest, the bits on those traces will arrive later than the bits on the other traces. This difference effectively constrains the time at which the bus as a whole can be latched. In the extreme, the difference in time between the arrival of the fastest and slowest bits can impact the minimum cycle time of the system.

Even in a system in which all clock and data skew has been eliminated, signal propagation delays can cause headaches, especially with regard to bidirectional buses. Most commonly, data is set up at the output pins prior to the start of the clock

cycle in which the data is to be communicated to another IC. At the appropriate edge of the clock, the data flows through the transmitting chip's output buffers. The data then travels down the traces to the receiving IC or ICs. The data must reach the destination devices prior to a setup time before the end of the cycle. At the end of the cycle, the data is latched at the input buffers of the receiving chip(s). Regardless of the direction of information flow, this scheme allows one full cycle (less any clock skew) for an output driver, an input receiver, and propagation along the trace. If the trace is exceptionally long or heavily loaded, or if the output driver is slow to deliver data onto the trace, then the cycle time of the entire system must be increased to compensate.

Coping with slow communication in high-speed systems

One common reason for the output buffer being slow to deliver data is that the data is being generated in the same cycle that it is being communicated. In this case, less than one full cycle is allocated to actual communication between the chips. If the time from the clock to the data being valid on the output pin is too long, and the cycle time of the system is being constrained, then the clock at the receiver can be delayed relative to the clock at the transmitter. One drawback is that the time given to communication in one direction comes directly out of the time available for communication in the other. Also, it can be hard to manage and control many clocks in a system.

The obvious solution to this problem of long propagation times is to orchestrate communication at some lower frequency than the system's primary frequency. This is the idea behind the proliferation of processors with frequency-doubled, -tripled, and -quadrupled clocks internally. To prevent low-speed external communication from restraining high-speed internal computation, the processor's external interface is run at one-half, one-third, or one-quarter the rate of the CPU core. Thus, two, three, or four processor cycles are available to generate, propagate, and receive external signals.

The idea of shifting the send and receive clocks in time to provide more time for signal propagation can be taken further. If the clock at the receiver is shifted one-half cycle out of phase from the clock at the transmitter, symmetry can be restored. If one-and-one-half cycles are used for communication in each direction, then three full cycles, instead of the normal two, are allocated to there-and-back communication. This increased number of cycles naturally increases the latency of operations being performed in the remote IC. This practice is not without its hazards, though. In burst transfers in which a new piece of

data is sent on every cycle, the driver conceptually begins sending the second chunk of information before the first chunk is latched at the destination. If the difference between the driver's data-valid-out time and data-hold time is too great, then the first piece of information may disappear before it can be latched. Under such circumstances, phase shifting the send and receive clocks with respect to each other is of limited utility and the only recourses are to:

- Send data every other cycle, as with a clock-doubled CPU
- Restrict the operating frequency of the system
- Explicitly pipeline the communication by adding latching between the source and destination ICs.

4.3.3 Issues Specific to Memory Systems

As DRAMs move to higher operating frequencies, some situations that commonly arise in the design of high-performance memory systems will begin to confront designers of mainstream memory systems.

The common structure for memory systems today is a wide bus with address, data, and control signals in parallel on a PC board. On this wide bus are SIMM sockets, one or more of which are populated with conventional SIMMs of various configurations. The sockets need not all contain the same type of SIMM. This physical structure is not at all conducive to high-speed operation for the following reasons:

- The motherboard bus is long and nonuniformly loaded. A large maximum load means short worst case transition times and long propagation delays.
- The address and control traces on the SIMMs are heavily loaded and long enough to be reflecting and resonating stubs that contribute to poor signal quality.
- DRAMs, which are quite slow to turn on and off, limit the frequency with which multiple sub-banks sharing the main bus can be interleaved.

One technique that can effectively control the reflections caused by the stubs on the SIMMs is to place small resistors between the main bus and the stubs in series with the connectors. These resistors dampen the effect of the reflections on a motherboard

trace and help maintain good signal quality throughout the memory system.

In a synchronous memory system, the signal propagation times alone make the use of unbuffered SIMMs untenable at very high frequencies. But the use of buffered SIMMs cause new problems. The buffers add delay, and, more significantly, variability in the delays of address, control, and clock signals: this variability translates directly into clock and data skew. The extraordinary amount of clock skew introduced by standard per-SIMM buffers can be eliminated by using Phase-Locked Loops (PLL) on every SIMM. However, using PLLs adds cost and requires additional start-up time after a power down.

The delay introduced by address and control buffers adds significantly to the total propagation time from the memory controller to the synchronous DRAMs,⁶ making single-cycle, end-to-end communication difficult: latching buffers may be necessary for operating frequencies above 66 MHz. The latches may facilitate high operating frequencies, but of course they also increase the latency of all operations by one cycle. A comparable set of buffers may also be needed on the data lines to preserve good signal quality on the motherboard bus and maintain high-frequency operation.

Ideally, the buffers would be on the motherboard instead of on the SIMM in order to minimize the variability of loading on the motherboard bus. However, placing the buffers on the motherboard increases the motherboard's cost and area. The alternative is to place the buffers on the SIMMs, and mandate that all SIMM sockets be filled with either real SIMMs, or dummy SIMMs with loading comparable to that of real SIMMs.

The physical design of memory systems is very important in attaining high-speed operation. Regular placement of DRAMs or SIMMs ensures consistent trace lengths and uniform loading. Also, since the current consumption of DRAMs is very bursty—DRAMs draw a lot of current when doing row and column accesses but little current in between accesses—and because DRAMs are relatively sensitive to power supply noise, careful placement of adequate bypass capacitors throughout the memory array is very important. Bypass capacitors are small capacitors, connected to the power and ground planes of the printed circuit board, that supply the transient current needs of

6. Recall that *synchronous DRAM* refers to any DRAM with a synchronous interface, rather than to a *Synchronous DRAM*, which is a JEDEC-standardized DRAM.

the DRAMs. Inadequate bypassing, or selection of the wrong capacitors, can lead to unreliable operation. Many capacitors designed for low-frequency applications are primarily inductive at high frequencies. Staggering the refresh operations of individual banks and sub-banks is also very important to the maintenance of high-quality supplies.

4.4 Memory Systems for PCs

4.4.1 x86 Machines

Cost is by far the most important aspect of memory system design for PCs. PCs are also sold on the basis of configuration and, increasingly, service. Processor type and frequency are generally the only performance-related design considerations, especially for low-end and mid-range products. Furthermore, in the presence of primary and secondary caches, the memory system's leverage on system-level performance is very small. Memory system design changes that improve performance at the expense of increased cost are seldom worthwhile.

Figure 4-11 shows a block diagram of a typical Intel Pentium®-based desktop PC. The processor bus connects the processor, the secondary cache and two system control ASICs. The System Controller ASIC and the Datapath ASIC separate the processor address and data buses from the memory and I/O buses. The DRAMs are logically and electrically quite close to the processor. Consequently, the overhead in the cache miss penalty, above and beyond the basic DRAM access time, is minimal. The secondary cache control is either incorporated into one of the system control ICs or into a separate dedicated device.

Low-end and midrange systems are often built with sockets for two banks; each bank is constructed of two $\times 32$ SIMMs. Some PCs are now being built with $\times 64$ DIMMs, and many **chip sets** allow a mix of socket types. Commonly, bank depths of 256K–4M are allowed; the wide range of allowed bank depths facilitates a correspondingly wide range of memory sizes: 2–128 Mbytes. With $1\text{M} \times 4$ DRAMs, the minimum memory size (granularity) is 8 Mbytes.

To provide greater flexibility to the system designer and end user, chip sets are programmable with respect to several of the

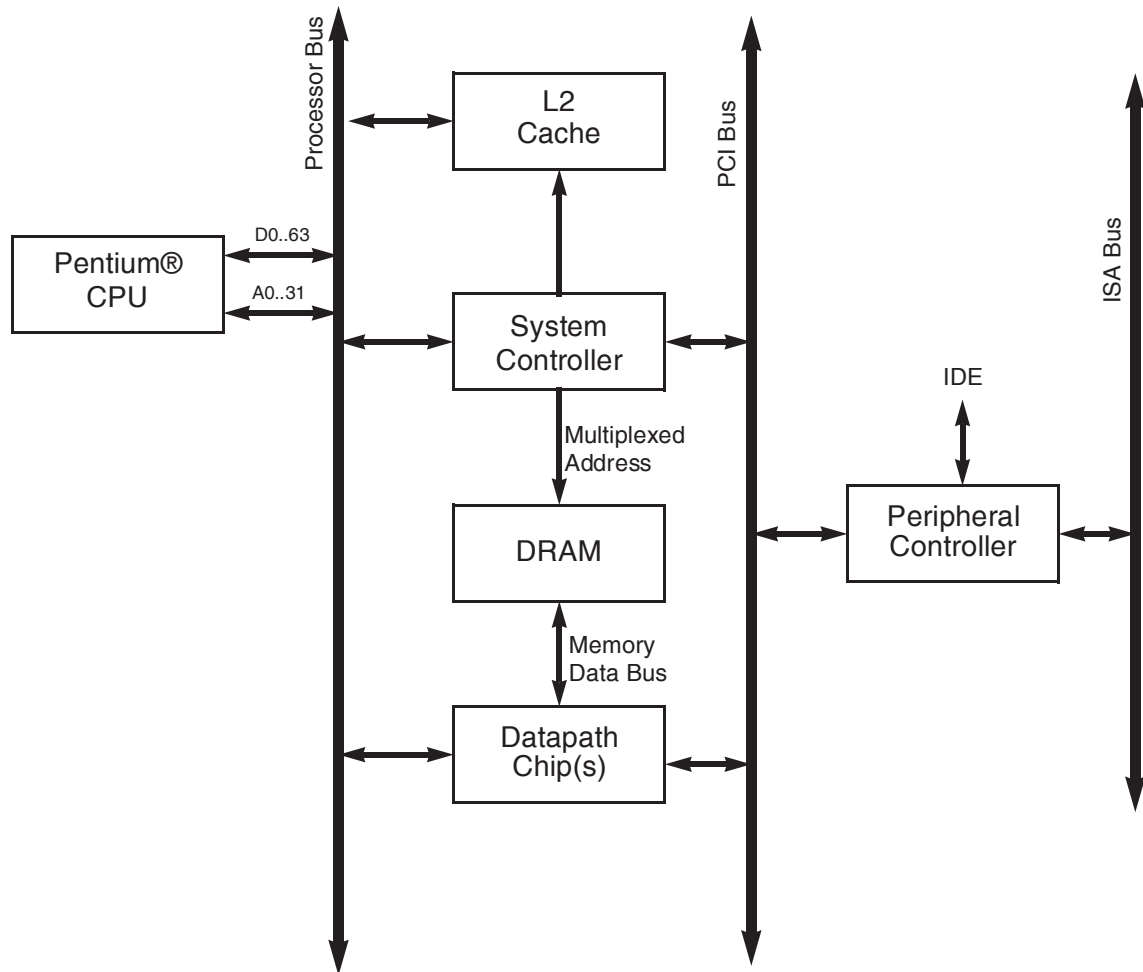


Figure 4-11 Block diagram of a typical Pentium®-based PC.

major DRAM timing parameters. Many PC chip sets support interleaved accesses when two compatible banks are installed. They can also perform hidden and/or staggered refreshes.

Almost all new PCs have at least one level of caching. In Intel 486-based systems, the primary cache is a **write-through cache**: all processor-generated data writes are written to the cache and immediately to main memory. The external secondary cache, if there is one, can be either write-through or write-back. In a **write-back (copy-back) cache**, written data is propagated to main memory only when a cache line that has absorbed a write is replaced. Such a line is called a dirty line. If only write-through caches are present in the system, the

majority of the references reaching main memory are small writes of one to four bytes. In contrast, when there is a write-back cache, almost all references reaching memory are block operations, and between one-third and one-half of these are writes. The Pentium® processor has a separate instruction cache and a write-back data cache on-chip.

Typically, the chip set provides individual $\overline{\text{RAS}}$ signals for each rank supported. Chip sets generally support up to four, six, or eight ranks, though most low- and midrange PCs only provide sockets for two ranks. $\overline{\text{WE}}$ signals are shared by all ranks but are sometimes duplicated on the chip set to reduce the fanout on the motherboard. Four or eight $\overline{\text{CAS}}$ signals are shared by all ranks—one $\overline{\text{CAS}}$ signal for each byte lane in the memory system.⁷ This matrix of $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$ signals allows for full-speed byte writes. During a byte write, only one of the four or eight $\overline{\text{CAS}}$ signals is asserted.

A serious consequence of this orthogonal matrix of $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$ lines is that at most one bank can be active (i.e., open and available to column accesses) at a time. Since the DRAM chip-selects are not used in PCs—narrow SIMMs and some early DRAMs do not provide them—if more than one $\overline{\text{RAS}}$ is low at a time, then all open banks would perform the column operation indicated by the shared $\overline{\text{CAS}}$ lines. Because only one bank can be open at a time, the DRAMs' row cache cannot be very effective, since only a fraction of it can hold data at a time. Consequently, most chip-set DRAM controllers precharge the accessed bank immediately after every transaction. In most cases, the precharge is complete by the time the next memory access arrives at, and is recognized by, the chip set. Consequently, the main memory access time is almost always equal to the row access time of the DRAMs plus the inevitable chip-set and bus transfer overhead.

Until recently, most PC chip sets did not try to optimize the L2 cache-miss timing. The chip sets took several cycles to recognize an L2 miss, decode and multiplex the address, and initiate a DRAM row access. This substantial overhead—together with the processor's internal L1 miss overhead—matched the DRAM access time in importance. More recently, though, chip-set designers have been optimizing the main memory controller to minimize the number of wasted cycles in the case of an L2 cache miss.

7. In 486-class machines, the system and memory buses are 32 bits wide, so four $\overline{\text{CAS}}$ lines are needed. In Pentium and Pentium Pro systems, the system memory buses are 64 bits wide, so eight $\overline{\text{CAS}}$ signals are used.

Fill frequency requirements

In early 1995, before the release of Windows 95, the acceptable granularity of low-end PCs was 4 Mbytes, especially on the 486-class systems. The acceptable peak bandwidth for these systems was around 100 Mbytes/s, making for a market-required fill frequency of around 25 Hz. High-end Pentium systems had 8-Mbyte granularities and peak bandwidths of around 180 Mbytes/s. Consequently, their fill frequency market requirements are around 22 Hz.

However, by the end of 1995, with Windows 95 firmly entrenched as the default operating system on new x86 PCs, granularities jumped by a factor of two: to 8 Mbytes at the low end and 16 Mbytes at the high end. At the low end, the peak bandwidth market requirement also doubled as the 32-bit memory systems of the 486-class machines were replaced by the 64-bit memory systems of Pentium systems—consequently, the fill frequency requirement changed little. At the high end, much of the industry shifted from fast-page-mode DRAMs to EDO DRAMs, effectively raising the peak bandwidth requirement to about 260 Mbytes/s. The two shifts—from fast-page mode to EDO and from 8- to 16-Mbyte granularity—combined to drop the high-end fill frequency requirement to about 16 Hz.

The required fill frequencies for PC memory systems have declined from the range of 30–50 Hz a decade ago to about 15–25 Hz recently (see Section 2.5). The ranges are fairly broad: peak bandwidth requirements are very weak because most computer purchasers do not directly care about bandwidth. The primary reason for the decline has been the increasing use of caches in the PCs to reduce the CPU's demands on the memory subsystem. However, with the advent of streamed multimedia data types and significant non-CPU compute requirements, bandwidth and fill frequency requirements are likely to creep up again in the coming years.

4.4.2 The Macintosh Family

Significant differences between the memory systems of Apple Computer's Macintosh family and those of the x86-based PC compatibles reflect Apple's flexibility in design—a flexibility that stems from Apple's control over the Macintosh product line. For example, though most of the recent Macintosh computers have used standard 30-pin SIMMs, the Macintosh IIfx and Macintosh IIsi used unusual 64-pin SIMMs. Apple also routinely solders the basic memory configuration onto the motherboard, providing sockets only for expansion memory. Finally, Apple

normally does not provide parity protection for main memory. Only two of Apple's machines—the Macintosh IIci and Macintosh IIfx—allow the user to attain parity protection through the installation of $\times 9$ SIMMs instead of $\times 8$ s.⁸

Some of Apple's latest PowerPC systems were among the first to use the 8-byte 168-pin DIMMs.

4.4.3 Portables

The important design criteria in the portable PC domain are similar to those in the desktop arena. The main difference is the PC domain's emphasis on low power consumption. Most portable systems use special low-power DRAMs with long refresh periods and/or lower operating currents. In addition, to reduce the power consumption per row access, these machines tend to use wide $\times 16$ DRAMs (see Section 6.2). Most notebook and laptop machines take proprietary memory modules instead of generic SIMMs, primarily because of the very tight physical constraints inside the machines. The latest standard memory modules, Small-Outline Memory Modules (SOMMs), give portables the same flexibility that SIMMs provide in the desktop market, but with a higher packing density. Many of the smallest machines use PCMCIA card slots as a mechanism for memory expansion.

Interestingly, some of the early Apple PowerBooks used Pseudo-Static RAMs (PSRAMs) instead of regular DRAMs. PSRAMs are DRAMs with all the refresh control circuitry on chip. They have the important advantages of very low power consumption and entirely autonomous refresh. Consequently, when the system is idle, the memory controller does not have to periodically initiate refresh cycles—it can go to sleep with the rest of the system.

4.5 Memory Systems for Workstations

Workstations and servers are different from PCs in three important respects that impact their respective memory systems.

8. *Designing Cards and Drivers for the Macintosh Family*, Third edition, Apple Computer Inc., 1992.

First, a typically configured workstation has four to sixteen times more memory than a midrange PC. Second, workstations' greater processing power and higher I/O rates imply substantially higher main memory bandwidth requirements. Finally, performance plays a more significant role in influencing workstation buyers. Therefore, adding some cost to the workstation memory subsystem in order to improve system-level performance by a few percentage points may be justified.

The primary consequence of the larger bandwidth in workstations is that workstation and server memories tend to be wider and are more commonly interleaved to get maximum performance. Both these trends increase the granularity of the system, but fortunately that is generally not a problem because of increased memory sizes.

Workstations and workstation servers span an important organizational boundary. Like PCs, low-end workstations are essentially single-board computers. The processor or processors, the memory, and the bulk of the graphics and I/O capability are on one printed circuit board. Many trade-offs and choices in this domain are fundamentally similar to those in the PC arena. However, at the high end of the workstation market, the systems follow the traditional minicomputer model with processor cards, autonomous memory cards, and I/O cards interconnected by a backplane bus (see Figure 4-12).

To improve performance, workstation designers increasingly use buffered SIMMs. By facilitating shorter column cycle times, buffered SIMMs yield memory systems with higher bandwidths. Since the address buffers are on the SIMMs themselves, the signal traces distributing the address and control signals from the memory controller are shorter and lack long, heavily loaded stubs on the SIMMs. Elimination of these stubs significantly improves signal quality and reduces overall signal propagation time to the DRAMs.

Multiprocessors increase memory traffic

Multiprocessors are rising dramatically in popularity in the workstation arena. The primary impact of multiprocessing on the memory subsystem is increased memory traffic. The increased number of processors means a linear increase in baseline memory traffic. The amount of additional bus and memory traffic can be substantial when processors share data and cooperate on a task.

Aside from increased bandwidth requirements, the impact of multiprocessing on the memory subsystem varies greatly

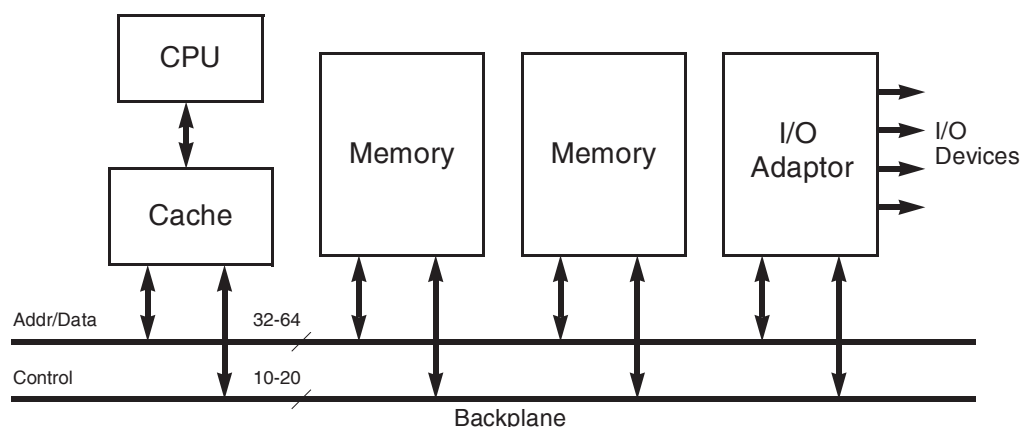


Figure 4-12 Many high-end workstations have CPU(s), memory, and I/O blocks connected on separate cards connected by a backplane.

depending on the type of computer system. The impact is limited in the case of small-scale multiprocessors with snoopy-bus-based **cache-coherence** protocols. The impact is more significant when the cache-coherence problem is solved by keeping directories at main memory. In the latter case, the memory subsystem is charged with keeping track of which memory blocks are currently held in which caches. Memory plays an active role in the sharing of data by mediating between two or more processors bidding for exclusive access to shared data. The need to play this role adds complexity to the memory controller and mandates a secondary bank of RAMs for storing the directory of coherency information. This directory generally adds 5–15% overhead to the size of main memory.

Increased memory traffic in multiprocessors can be combated through the use of a distributed memory architecture. In this architecture, main memory is broken up into smaller chunks and physically distributed throughout the system. Typically, each chunk is associated with a processor. As the system grows, more nodes—each consisting of a processor, some memory, and a network interconnection—are added. With some operating system collusion, most of a processor's memory references will be to its local memory even though there is a shared global address

space. The total amount of network traffic is therefore limited to shared data and possibly I/O.

Workstations use error protection

Most workstations use error detecting and correcting codes in their memory systems to protect against soft and some hard errors. Use of a single-error correcting/double-error detecting (SECCDED) code is an explicit requirement for many workstation market segments.

Finally, workstations are characterized by secondary caches that are larger than those found in PCs. The decreased miss rate of these large caches helps reduce the demands on the memory system. However, large caches also decrease the performance impact of main memory. This reduced performance leverage and the longer cache blocks found in the large secondary caches (typically 32–128 bytes) combine to decrease the emphasis on low memory latency and to increase the importance of high-bandwidth solutions.

Fill frequency requirements

Compared with high-end PCs, workstations have greater bandwidth requirements but dramatically relaxed (i.e., increased) granularity requirements. For example, minimum memory configurations of 64–256 Mbytes are common in this market segment. The net result is that the fill frequency requirements for workstation memory systems are currently in the vicinity of 10–20 Hz. These lower fill frequencies explain why the workstation market adopted the 4M×4 fast-page-mode DRAM, while the PC industry waited for the 1M×16 devices to be cost-competitive.

4.6 Memory Systems for Large Machines

Three classes of large machines are noteworthy: regular mainframes, vector supercomputers, and massively parallel machines. Common to these machines are very large memory sizes and massive amounts of memory bandwidth available throughout the machines. Multigigabyte primary and secondary memories are commonly supported with many gigabytes per second of aggregate bandwidth.

Mainframes

Thus the primary difference between a mainframe and a high-end workstation or server is not so much processing power as

I/O capability. Over the last half-decade, the dramatic performance lead that mainframes once held over microprocessors has vanished. However, no uniprocessor microprocessor-based system exists that can keep a terabyte disk farm meaningfully utilized.⁹

The greater I/O capability of mainframes affects the memory subsystem in two ways: main memory size is increased and memory bandwidth is larger. Mainframe main memories typically start at several hundred megabytes on a minimally configured system and run as large as several gigabytes. Frequently, main memory is logically and physically partitioned into two separate units: a main store and a secondary store. The main store, which is the smaller of the two, is sometimes implemented with SRAMs; the larger, slower secondary store is always constructed of DRAMs. Unlike a cache, which is not visible to software, the operating system manages both the main and the secondary stores.

With 4-Mbit DRAMs, 1 Gbyte of secondary store takes 2,048 ICs. If mounted on 128 SIMMs, this same memory could be placed on a single large PC board. Consequently, though memory sizes have continued to grow over the years, the physical volume of mainframe memory systems has shrunk: the appropriate measure of mainframe memory has changed from cubic meters to cubic feet.

Increased bandwidth requirements are satisfied in three ways:

- Through the use of static RAM where economics permit
- Through greater interleaving of memory banks
- Through wider memory organizations.

Eight- and sixteen-way interleaving is common in machines of this class, as are 512- and 1,024-bit memory buses.

Vector supercomputers

Vector supercomputers make unique demands on their memory systems. As a consequence of the very large problem sizes that these computers work on, even large caches are of little value. Therefore, performance depends heavily on the ability of the memory system to provide multiple streams of vector data simultaneously at very high bandwidths. Each processor is capable of absorbing two double-precision operands and

9. In fact, many recent medium to large machines use many microprocessors, operating in concert with a powerful I/O system, to effectively manage very large data sets.

generating a double-precision result every cycle. Chaining functional units together increases the number of operands per cycle that the memory unit must provide.

Supercomputer memory systems are very heavily interleaved—up to 512-way interleaved—with multiple paths from the subbanks to the memory controller and on to the processors. The whole subsystem is engineered to minimize the number of resource conflicts between otherwise independent read and write accesses, so that vector processors can operate at full speed the majority of the time. For most vector supercomputers, many cycles of latency are required to start up a vector operation. The emphasis for memory system design is clearly on high bandwidth and low contention for memory. The trade-off between the use of SRAM and DRAM for main memory is driven mainly by economics, performance, and, to a lesser extent, by the physical implications of a lower density SRAM memory system. All midrange to high-end supercomputers have used SRAMs in their main memories. For the lower performance, entry-level machines, designers have occasionally chosen DRAMs to improve affordability.

Massively parallel machines

In many respects, massively parallel processors (MPPs) represent the other end of the spectrum from vector supercomputers. In an MPP, memory is not a single large, complex subsystem. Instead, memory is divided into small chunks and is distributed throughout the machine; it is associated with either individual processors or small groups of processors. The amount of memory under the control of each autonomous memory controller varies significantly according to the system architecture. One class of MPP uses very large numbers of simple processors, small clusters of which share memory systems consisting of a few DRAMs each. With this class of MPP, the problem is not getting massive amounts of bandwidth out of a large number of DRAMs, but squeezing as much bandwidth as possible out of a very small number of RAMs while retaining a relatively simple memory controller.

Fill frequency requirements

The fill frequency requirements of large systems are difficult to categorize because of the wide spectrum of machine architectures and application domains. Furthermore, with the partitioning of main memory into primary and secondary stores, a hierarchy of fill frequency requirements is introduced within a single machine category.

That said, generally, many large systems rely on very large main memories to completely hold the problem or problems

being worked on. In contrast, PCs and—to a much lesser extent—workstations rely on virtual memory to provide the illusion of large physical memory. Consequently, the granularity requirements in large systems are substantially larger than those in other system classes; the granularities are larger even relative to the increased bandwidth requirements. As a result, the fill frequency requirements for the main memories of large general purpose machines are typically lower than those of workstations and PCs.

4.7 Graphics Memory Systems

The visual display of information in many forms is an increasingly important component of what computers do. Graphics controllers are responsible for 1) maintaining an image of the screen's current contents; 2) accepting commands that describe at some level of abstraction changes to the appearance of the screen; and 3) interpreting those commands by changing the controller's internal image of the screen. Also, the contents of the internal image must be continuously translated into the electrical signals needed to drive the physical display. All graphics controllers have a small graphics memory system that encodes the screen's current appearance.

When a one-to-one correspondence exists between locations in this local memory and the appearance of individual pixels on the screen, the memory is said to contain a **frame buffer**. Depending on the number of colors that can be displayed at once, between one and thirty-two bits are used to describe the appearance of each pixel. One bit per pixel implies a monochrome display; eight bits per pixel implies a choice among 256 colors for each pixel; and twenty-four bits per pixel, called *true color*, allows each pixel to take on any of 16.7 million colors.

Part of the graphics controller's job is to regularly scan the entire contents of the frame buffer and drive the video monitor correspondingly. This task takes up a portion of the video memory's total available bandwidth. Table 4-2 lists both the peak and the average amount of bandwidth needed to refresh the screen for a variety of video display configurations. The peak bandwidth is the rate at which data must be fed into the RAMDAC while the CRT's electron beam is drawing the visible portion of the screen. The average bandwidth is computed by multiplying the frame buffer size by the refresh rate. The

average bandwidth with compression column is discussed in Section 4.8. The average bandwidth and the peak bandwidth differ because pixels are not drawn during the horizontal and vertical retrace periods. Peak bandwidths are 50–100% higher than average bandwidths—though the exact ratio depends on the display configuration.

Low-end graphics controllers use generic DRAMs

Figure 4-13 shows a coarse block diagram of a graphics controller that uses generic DRAMs to hold the frame buffer. The graphics controller uses the single port on the DRAM to make changes to the display contents and to retrieve the frame buffer for display on the screen. Regular generic DRAMs work acceptably as video memories for low-performance graphics with small resolutions and color depths because the total bandwidth required of the frame buffer memory is less than the 80–100 Mbytes/s that can be obtained from a $\times 32$ memory system. Some of the more recent graphics controller chips can make use of $\times 64$ memory systems to generate even more bandwidth. Recently, midrange graphics applications have been using high-speed DRAMs to generate more bandwidth than is possible with regular generic DRAMs, but without the cost and design effort of a VRAM-based solution. Other recent trends are the integration of the RAMDACs onto the same device as the

Name	Resolution	Bits per Pixel	Colors	Frame Buffer Size	Refresh Rate	Peak Display Bandwidth	Average Display Bandwidth	Average Display Bandwidth w/ Compression
VGA	640 $\times$ 480	8	256	300 Kbytes	60 Hz	33 Mbytes/s	18 Mbytes/s	1.2-1.8 Mbytes/s
	640 $\times$ 480	16	64K	600 Kbytes	60 Hz	66 Mbytes/s	36 Mbytes/s	2.4-3.6 Mbytes/s
SVGA	800 $\times$ 600	8	256	480 Kbytes	60 Hz	52 Mbytes/s	29 Mbytes/s	1.9-2.9 Mbytes/s
	800 $\times$ 600	16	64K	960 Kbytes	70 Hz	122 Mbytes/s	67 Mbytes/s	4.4-6.7 Mbytes/s
	1,024 $\times$ 768	8	256	786 Kbytes	70 Hz	100 Mbytes/s	55 Mbytes/s	3.6-5.5 Mbytes/s
	1,024 $\times$ 768	16	64K	1.6 Mbytes	70 Hz	200 Mbytes/s	110 Mbytes/s	7.3-11 Mbytes/s
	1,024 $\times$ 768	24	16M	2.4 Mbytes	72 Hz	300 Mbytes/s	170 Mbytes/s	11-17 Mbytes/s
	1,280 $\times$ 1,024	8	256	1.3 Mbytes	72 Hz	172 Mbytes/s	94 Mbytes/s	6-10 Mbytes/s
	1,280 $\times$ 1,024	16	64K	2.6 Mbytes	72 Hz	344 Mbytes/s	189 Mbytes/s	12-19 Mbytes/s
	1,280 $\times$ 1,024	24	16M	3.9 Mbytes	72 Hz	516 Mbytes/s	283 Mbytes/s	19-28 Mbytes/s
	1,600 $\times$ 1,200	24	16M	5.8 Mbytes	72 Hz	756 Mbytes/s	415 Mbytes/s	27-41 Mbytes/s
HDTV	1,920 $\times$ 1,080	24	16M	6.2 Mbytes	72 Hz	816 Mbytes/s	448 Mbytes/s	N/A

Table 4-2 The display bandwidth grows with the screen resolution, color depth, and refresh rate.

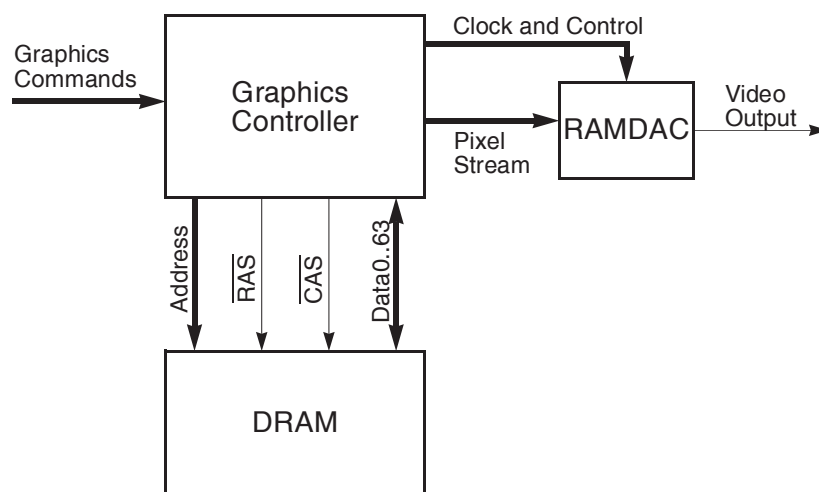


Figure 4-13 Low-end graphics controllers use generic DRAMs.

graphics controllers and the addition of more GUI acceleration into the graphics controller to help relieve the CPU of some of the more mundane display management tasks.

High-end graphics need VRAMs

When higher resolution, more colors, or better video performance is desired, standard DRAMs are no longer adequate for the task. For these systems, **Video RAMs (VRAMs)** are used. As described in Section 3.6, VRAMs are two-port devices designed for use in graphics controllers. In addition to an augmented standard DRAM interface, they have a high-speed serial interface tailored for dumping the frame buffer to the screen. Figure 4-14 shows a block diagram of a graphics controller using VRAMs.

When still higher performance is required, as in the case of real-time 3-D object rendering, even more complex hardware structures are put in place to speed the interpretation of high-level graphics commands from the processor. For example, a Z-buffer can be added to the frame buffer to improve 3-D performance. A Z-buffer holds depth information associated with each pixel to help the controller decide whether some new object being drawn is in front of or behind the object already drawn at that location. Another feature that improves high-performance 3-D is double-buffering. Double-buffering provides one copy of the frame

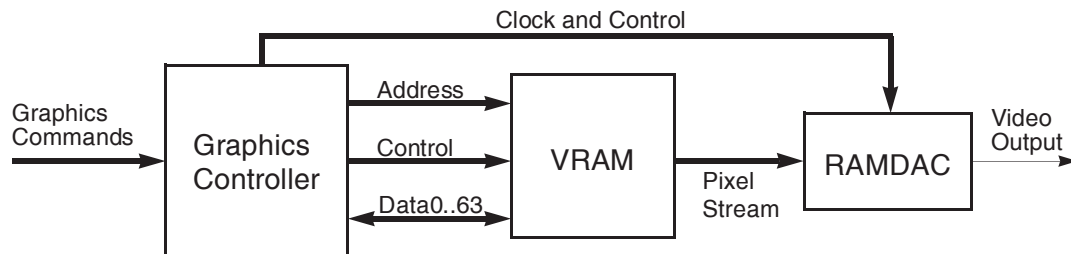


Figure 4-14 Midrange graphics controllers use two-ported VRAMs to increase the frame buffer bandwidth.

buffer for displaying to the screen and a second copy for rendering a scene. Once a scene is fully drawn into the second copy, the two buffers exchange roles, and the display hardware dumps the completed scene to the screen. Two frame buffers can be used in virtual reality applications; the two buffers are used to generate separate images for the left and right eyes, so that the user perceives depth properly.

Tiling improves DRAM utilization

An increasingly important question in the design of frame buffers is the way in which pixels on the screen are mapped onto bytes in the physical memory (i.e., in the frame buffer). One method is natural and intuitive: the screen consists of a rectangular collection of rows (i.e., scan lines), while DRAMs are a rectangular collection of rows of bits. It seems natural then to map scan lines onto DRAM rows. This arrangement, shown in Figure 4-15, was used for many years.

This straightforward mapping is very efficient for screen refresh. A single row access brings in a big chunk or all of a scan line into the row cache, from which it can be dumped to the RAMDAC with column accesses. Unfortunately, though, this arrangement is not very efficient for rendering. All graphical objects except for horizontal lines cross multiple scan lines. Drawing a vertical line requires writing only a few pixels—as few as one—on each scan line. Thus, drawing a vertical line becomes a sequence of row accesses with a single column write access following each. This request sequence has the lowest possible efficiency and yields the lowest sustained bandwidth.

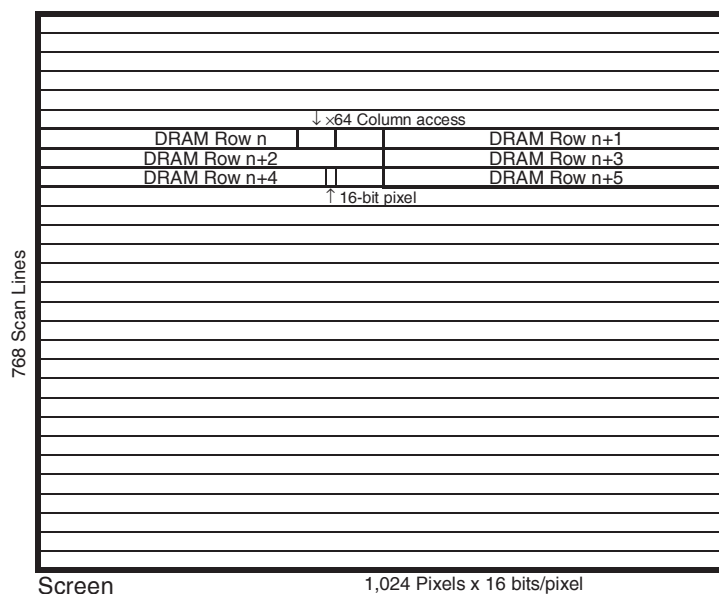


Figure 4-15 In a traditional, untiled frame buffer, each DRAM row is mapped onto a single scan line. In this example, each DRAM row represents a portion of the screen that is one scan line high and 512 pixels across.

Although the traditional, untiled method was used for many years, a tiled frame buffer offers significantly improved performance. In a tiled frame buffer, rectangular regions of pixels are mapped onto DRAM rows. These regions, or tiles, are typically up to eight scan lines tall, and 256 pixels across—plus or minus a factor of two in both directions. The size and best aspect ratio for the tiles depends on the logical layout of the DRAMs being used (i.e., the size of their row cache), the screen resolution, and the color depth.

With a tiled frame buffer, the screen refresh and best case rendering (i.e., when drawing horizontal lines) performance is not noticeably worse than with the untiled mapping. However, the worst case rendering performance is very much better. When drawing a long, thin vertical line into tiles eight scan lines high, each row access is followed by eight column accesses. The worst case sustained bandwidth with a tiled frame buffer is about three times greater than with an untiled one. The average sustained bandwidth is generally improved by 50–100% with the introduction of tiling. The magnitude of the improvement

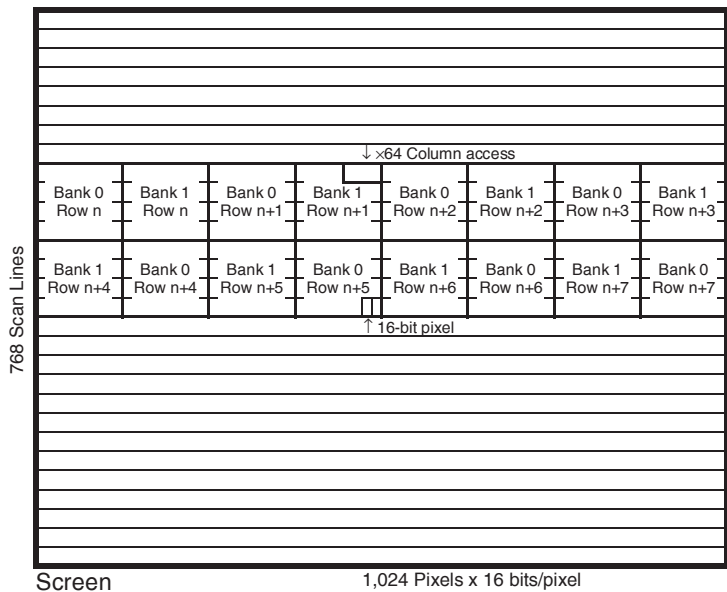


Figure 4-16 A tiled frame buffer has rectangular regions—here 4 pixels by 128 pixels—mapped onto DRAM rows.

depends on the DRAMs, the tiles, the screen resolution, and the objects being rendered.

A final optimization is possible when the frame buffer memory system has several banks. If horizontally and vertically adjacent tiles are placed in different banks, then one bank can perform a row access while another performs column accesses. This technique can hide a substantial amount of the total row access time, further increasing the sustained rendering performance. This tiled frame buffer structure is depicted in Figure 4-16.

Tiling a frame buffer has one drawback: mapping from the x-y screen coordinates to a DRAM row and column address is more complicated than with a untiled frame buffer. However, GUI accelerators need to support a variety of unusual screen resolutions, and the additional complexity in their address generation logic needed to support tiling is insignificant.

Multimedia more demanding of PC graphics

The growth of multimedia applications increases demands on PC graphics controllers. For example, the information content of an NTSC video signal is about 7.5 Mbytes/s, assuming 8-bit pixels. The ability to process and render a video signal onto a portion of the screen requires at least that much free frame

buffer bandwidth plus enough controller complexity to map the NTSC signal into a reduced pixel and color space.

The trend in the design of small portable systems is toward a unified memory system. Rather than having two separate physical memories—one for the main memory and one for the graphics memory—the logical memories are combined into a single physical unit. Unified Memory Architecture (UMA) memory systems are described in detail in Section 4.8.

High fill frequency requirements

Frame buffers—and graphics memory systems in general—have the highest fill frequency requirements of any of the main DRAM application domains. This is because the size and granularity of these memory systems are small—typically 1–4 Mbytes—and the peak and sustained bandwidth requirements are extreme. In 1995, at the low end of the GUI accelerator spectrum, the acceptable granularity was 1 Mbyte, while the peak bandwidth requirement of 100 Mbytes/s derives from a 32-bit memory width and a 25 MHz operating frequency for fast-page-mode DRAMs. These yield a peak fill frequency requirement of 100 Hz. At the high end of the performance spectrum, the minimum granularity is increased to 2 Mbytes, but the peak bandwidth requirement is increased to approximately 400 Mbytes/s: the result is a required peak fill frequency on the order of 200 Hz.

The bandwidth requirements for high-performance 3-D graphics systems are generally substantially higher than the requirements in the 2-D arena. This increase in the peak bandwidth requirement more than offsets the increase in granularity due to double-buffering and the addition of a Z-buffer. Consequently, the ratio of the bandwidth to minimum memory size—i.e., the peak fill frequency requirement—for 3-D graphics is larger by a substantial margin than the ratio for all other significant memory system types.

The high fill frequency requirements of graphics memory systems suggest why the graphics world is still using DRAMs that are 4-Mbit and smaller while the main memory world has shifted to the more cost-effective 16-Mbit devices. As illustrated in Section 2.5, none of the generic 16-Mbit devices' fill frequencies are adequate to meet the graphics market requirements.

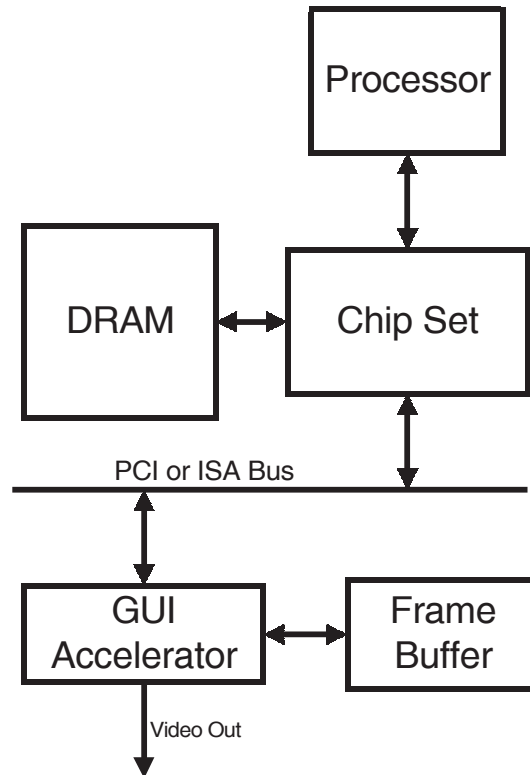


Figure 4-17 The traditional PC graphics architecture.

4.8 Unified Memory Architecture (UMA)

In the traditional PC architecture, the graphics controller is physically and logically separate from the central portion of the system: it sits out on the ISA or PCI bus, on a separate daughtercard inserted into a socket on the motherboard. Consequently, most PCs contain two separate DRAM memories: 1) a main memory array, constructed of SIMMs and controlled by the chip set, and 2) a frame buffer memory, usually soldered down or socketed on the graphics adapter card. Figure 4-17 illustrates this traditional PC system architecture.

The central idea behind **Unified Memory Architecture (UMA)** systems is to house these two logical structures (the main memory and the frame buffer) in a single memory array on the motherboard. The goal is to reduce total cost, increase

functionality, or, preferably, do both. The single memory array is shared and divided between the two logical memories so that the sum of the size of the main memory and the size of the frame buffer is equal to the total amount of memory physically present.

**New
cost-performance
trade-offs**

A switch to UMA reduces total cost. Total memory cost is reduced because the graphics memories used to make frame buffers are more expensive per bit than main memory DRAMs. Graphics memories are either lower density or have special functionality (as in VRAMs) to provide the bandwidth and granularity (or equivalently, the fill frequency) needed by graphics applications. Replacing graphics memories with cheaper main memory devices reduces total cost.

Alternatively, the frame buffer bits can be eliminated altogether. Instead of having 16 Mbytes of main memory and 2 Mbytes of frame buffer memory, a system could have 16 Mbytes of total memory, conceptually divided into 14 Mbytes of main memory and 2 Mbytes of frame buffer. The consumer, seeing 16 Mbytes in the advertisement, may not be fully cognizant that some of those bits are not available for general purpose computing.

Reserving some memory for the frame buffer results in some performance loss compared to a system with a separate frame buffer. Obviously, the magnitude of this performance loss depends on both the total amount of memory and the amount of memory taken up by the frame buffer. As the amount of main memory left for the operating system and applications decreases, the number of page faults increases and the user spends more time waiting on the disk. If 1 Mbyte is taken for the frame buffer from a cached 8-Mbyte system, the total performance loss is 5–10%. Stealing 2 of 8 Mbytes of memory for the frame buffer is very noticeable. However, when the total memory size is 16 Mbytes of memory, the performance loss from reserving 2 Mbytes for the frame buffer is again less than 10%. Without a secondary cache, the performance loss associated with the smaller main memory size is increased. In this case, taking 1 Mbyte from an 8-Mbyte system results in a 10–15% total performance degradation.

Of course, the magnitude of the performance loss is highly dependent on the application, the problem size, and the operating system. In some circumstances, the performance loss due to the loss of memory available to the operating system and application will be much greater than the amounts indicated

here. Similarly, in some situations the loss of main memory will not have a noticeable impact on end-user performance.

In a UMA system, contention over the single memory resource is another source of performance loss. In particular, if the memory array is busy performing graphics operations when a processor cache miss occurs, the processor may stall longer than it would in a traditional architecture machine. Similarly, if the graphics engine is precluded from accessing the memory because the processor is using the memory a lot, then graphics display will not be as snappy. However, queuing theory analyses indicate that so long as the amount of a shared resource—memory bandwidth in this case—is more than marginally greater than the total demand, the average delay from contention should be quite small.

Additional cost savings are effected through eliminating sockets, daughter PC boards, and—in some cases—one or more of the ICs needed to implement the graphics adapter.

One nice aspect of UMA architectures is that only the memory actually used for the frame buffer is purchased. In a traditional graphics daughter card, the graphics memory is binary-sized¹⁰ and must be large enough to hold the frame buffer for the highest supported resolution and color depth. When a lower resolution is used, a portion of the graphics memory system goes unused. For example, a 4-Mbyte frame buffer memory array is required by a graphics card that supports $1,024 \times 768$ displays and 24-bit pixels (see Table 4-2). However, whenever the user is in 8-bit VGA mode, only 300 Kbytes of that 4 Mbytes is used; 3.7 Mbytes are unused or "wasted." With a UMA system, the system can be booted in VGA mode with a small frame buffer. This leaves the maximum amount of memory for main memory. Ideally, display resolution could be changed dynamically and the logical boundary between main memory and the frame buffer could be changed on the fly while the system is operating.

In some circumstances, UMA architectures can improve performance over the traditional split architectures. For example:

- When the unified memory system is of a higher performance level than is typical for PC main memories, the average response time to CPU memory requests can improve.
- When some or all of the displayed images originate in main memory—for example, as in a decompressed video signal or

10. The number of bytes is an integral power of 2.

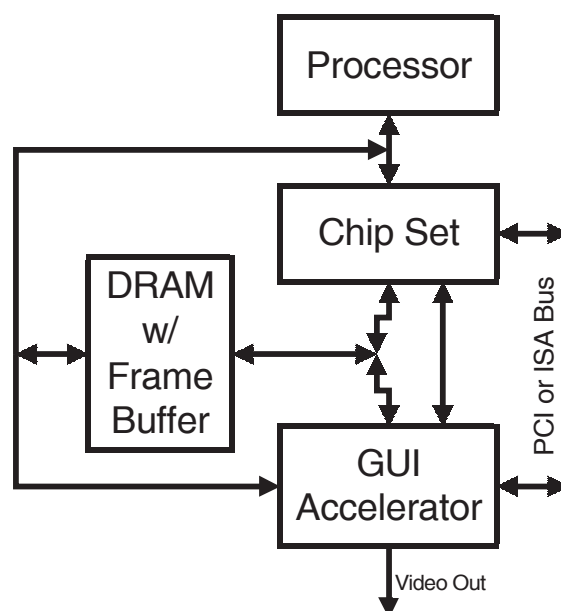


Figure 4-18 A loosely-coupled UMA system.

a texture-mapped scene—then those bits need not traverse the slow and sometimes overcommitted PCI or ISA bus.

Two styles of UMA systems

There are two styles of UMA machines: loosely-coupled (or dual-supplier) and tightly-coupled (or single-supplier). These two styles are shown in Figures 4-18 and 4-19, respectively.

A loosely-coupled (dual-supplier) UMA system contains the same ASICs as a traditional PC contains. However, these ASICs are interconnected differently. A GUI accelerator chip—possibly made by a different vendor than the chip set supplier—shares control of the unified memory array with that chip set.

As of 1Q1996, a three-pin interface connecting the chip set and the GUI accelerator is being standardized by the Video Electronics Standard Association (VESA). The standard provides for the GUI accelerator to request access to the memory array. Having two request priorities reflects the necessity of keeping screen refresh data flowing to the RAMDAC. When a request is granted, the chip set tri-states all the memory array control signals, and the GUI accelerator takes control until it either completes the task or is preempted by the chip set. Several

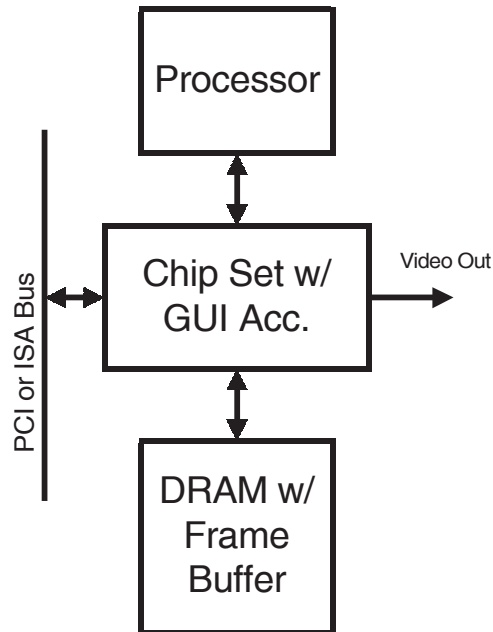


Figure 4-19 A tightly-coupled UMA system.

companies are promoting a slightly extended interface. The addition of another signal from the GUI accelerator to the chip set facilitates three levels of request priority. This allows an urgent need for screen refresh data (i.e., a nearly empty refresh data FIFO) to be differentiated from a nonurgent need (i.e., the refresh data FIFO is still one-half full).

An important part of the VESA standard are the software protocols that allow the system software modules controlling the two logically separate functional units—the BIOS for the GUI accelerator and the BIOS for the chip set—to communicate and share information about the size, type, and speed of the memory and cooperate on the allocation of physical bits to the two logical arrays.

Advantages and disadvantages of loosely-coupled UMA systems

Loosely coupled UMA systems have both advantages and disadvantages relative to the standard PC architecture. A main advantage is that a range of PC products can be developed by inserting different GUI accelerators into a socket on a standard motherboard. This differentiation is similar to that currently common in the product lines of the large PC suppliers: a

low-end system with smaller memory and disk has a cheaper, lower performance graphics subsystem, while the high-end product with the same processor¹¹ (and motherboard, chassis, and everything else) has more memory, a larger disk, a better CD-ROM, and a better graphics controller and monitor. The ability to efficiently and cheaply create a range of PC products is vital in the current PC marketplace.

The dual-supplier architecture enables a company with a single competency or specialization to offer products for UMA systems. This benefits users by increasing the range of product choice and by increasing competition.

A disadvantage of the loosely-coupled UMA architecture is that graphics data flowing from the CPU to the GUI accelerator must still traverse the PCI bus. Because only one $\overline{\text{RAS}}$ signal is shared between the two units, the GUI accelerator does not necessarily have access to all of memory.

A second disadvantage is the loosely-coupled UMA model's assumption that all banks are closed (i.e., $\overline{\text{RAS}}$ is raised) whenever control is transferred between the two controllers. This assumption is not a problem with traditional PCs and DRAMs: because at most one bank can be kept open at a time (see Section 4.4), little is gained by keeping banks open across the transition of control. However, with several new DRAM architectures, many banks can be kept open simultaneously. If these new DRAMs are to be used effectively, the banks must be kept open over long periods of time.

A third disadvantage of this organization is that there is necessarily additional loading on the DRAM control signals. The additional loading is probably not problematic with current fast-page-mode DRAMs. However, with the higher operating frequencies common among the new DRAM architectures, it is more important to keep the loading on the DRAM control lines to a minimum. Also, some new memory architectures do not easily facilitate having two masters share control of memory.

Perhaps the greatest disadvantage of the loosely-coupled UMA architecture relative to the standard PC architecture is that it is harder to develop a range of chip products that facilitate a range of system-level products.

11. High-end systems often have a higher clock-frequency version of the same processor.

Tightly-coupled UMA systems

In a tightly-coupled (single-supplier) UMA system, both the chip set and the GUI accelerator are integrated into a single ASIC or group of ASICs: they are no longer two physically separate units. In order to design such a chip set, a company must have competency in both graphics accelerator design and chip-set design—or two companies must cooperate on design and distribution.

One advantage of tightly-coupled UMAs is that as levels of integration and pin counts increase, the total number of chips in the system can decline. This leads to board area and power savings as well as the obvious savings in the bill-of-materials. In addition, a single memory controller enables easier management of high-performance DRAMs to get the most out of multiple banks and pipelined or concurrent transfers. Furthermore, with a single controller, the loading on the DRAM address and control signals is minimized, and so the operating frequency is maximized.

UMA architectures compared

Table 4-3 contrasts the strengths and weaknesses of the three basic PC architectures.

Characteristic	Traditional PC	Loosely-Coupled UMA	Tightly-Coupled UMA
Cost	Two memory arrays Two controllers	One memory array Two controllers	One memory array One larger controller
Performance impact	Baseline	Tolerable	Tolerable
Supported resolutions	Limited by size of frame buffer	Not limited	Not limited
Partitioning difficulties	None	Negligible	Severe in 64-bit system w/o BGA packaging
Ability to choose chip set and GUI accelerator	Completely independent	Independent but linked	One constrained choice
Main memory to frame buffer transfers	Via PCI	Via PCI	Memory to memory
Simultaneous main memory and frame buffer operation	Yes	No	No
Optimizing memory controller(s)	Yes; Independent	No	Yes
DRAM control signal loading	Typical for main memory	Increased	Typical for main memory

Table 4-3 The two forms of UMA architecture for the PC have different strengths and weaknesses.

High peak bandwidth requirements

The average bandwidth requirements placed on unified memory systems are essentially equal to the sum of the average main memory and graphics requirements. The true peak bandwidth requirement is actually the maximum of the individual main memory and graphics peak bandwidth requirements. However, since peak and average bandwidths are related, the realistic

peak bandwidth requirements lie between the maximum of the two individual components and their sum.

As with any system, bandwidth requirements are soft. As bandwidth demands approach the total available bandwidth, memory utilization increases, average access time increases, and system performance degrades. As the performance of the system degrades, bandwidth demands also decrease until the supply and demand for bandwidth are balanced. This degradation of performance occurs smoothly and gradually unless the supply of bandwidth is grossly inadequate.

It is also true that many applications are quite bimodal in their behavior: they go through periods of heavy computation alternating with periods of heavy display activity. For these applications, both the amount of interference between the CPU and the graphics demands on the unified memory system would be minimal because the CPU and GUI accelerators would not need their peak bandwidth requirements simultaneously.

As noted in Section 4.7, the bandwidth requirements from the graphics subsystem vary linearly with display resolution and color depth. The average screen refresh bandwidth—as a function of display resolution and frequency—is shown in Table 4-2. For resolutions above 800×600 with 16 bits of color, the amount of bandwidth needed just to dump the contents of the frame buffer to the RAMDAC is daunting. If screen refresh takes up too much of the total available sustainable bandwidth, then too little is left for the drawing of graphics and for the CPU. The CPU and GUI accelerator then compete for the remaining available bandwidth, and both CPU and graphics performance start to degrade. Unfortunately, screen refresh must have the highest priority when competing for a limited amount of total bandwidth: if the supply of pixels to the RAMDAC falters—if garbage is even occasionally drawn to the screen—user perception of display quality deteriorates rapidly.

**Frame buffer
compression
dramatically reduces
bandwidth
requirements**

Fortunately, the actual information content of the pixel stream to the RAMDAC is dramatically less than its bandwidth. Consequently, the pixel stream and, by extrapolation, the frame buffer, are highly susceptible to compression. Compression of the frame buffer can dramatically reduce the bandwidth needed for screen refresh. The last column of Table 4-2 on page 111 shows the amount of frame buffer bandwidth needed if the frame buffer contents are compressed by a factor of 10 to 15, as is typical in many applications.

Frame buffers are highly compressible because of two independent phenomena:

1. Groups of horizontally and vertically adjacent pixels commonly have the same or nearly the same color.
2. Most—if not all—of the screen is typically unchanged from one screen refresh to the next.

The first fact means that the information in the frame buffer is highly redundant and can be compressed by a wide variety of general purpose and customized compression algorithms. The second fact means that once a small region of the frame buffer is compressed, the compressed version will most probably be reused many times before that portion of the frame buffer is redrawn.

To illustrate how frame buffer compression can work, one possible algorithm is described below:

1. The frame buffer is conceptually divided into many regions, each most likely being a portion of a scan line. Each region is compressed as a unit. For each region, some storage is needed in addition to the uncompressed frame buffer. This separate storage, for all the regions together, constitutes a compressed version of the frame buffer.
2. If the pixel in the uncompressed frame buffer is changed by the GUI accelerator, then the compressed version of the region to which the pixel belongs is marked as invalid or old.
3. When performing a screen refresh of a region, the compressed version is checked. If it is invalid, then the uncompressed frame buffer is read and sent to the RAMDAC. In the process, it is compressed by one or more compression algorithms. The compressed result is placed in the compressed frame buffer and marked as valid.
4. If, when performing a screen refresh of a region, the compressed version is checked and found valid, then the compressed version is read out from the compressed frame buffer, decompressed, and sent to the RAMDAC.

The overhead associated with such a scheme is the cost of maintaining the compressed frame buffer. This cost is in the form of memory (the size of the compressed memory), the bandwidth needed to maintain the compressed frame buffer, and the gates needed to implement the compression and decompression

algorithms. With appropriately sized regions, all three cost components can be kept small.

One potential problem associated with keeping a compressed copy of the frame buffer is the bandwidth needed to invalidate the compressed representation of the regions every time a pixel is modified. However, since pixel writes are very spatially clustered, caching of recent invalidations works very well to reduce this burden.

As mentioned earlier, frame buffer compression works because screen contents are typically very locally uniform and generally very stable. Business applications in particular have very low rates of pixel change: as the user types in a word processor or spreadsheet, only the pixels representing the words or numbers being modified are changing. But even for applications such as games that involve a lot of dynamic screen activity, much of the screen is essentially static from one scene change to another.

**Frame buffer
compression less
effective with video
and 3-D graphics**

Frame buffer compression works less well for some applications and application domains. One such application is video. A typical video stream writes every pixel in the video window at a rate of 30 Hz. Even though many of the pixel values are unchanged from one video frame to another, recognizing this fact requires either comparing the new pixel values with the old or computing a hash function on the new pixels in a region and comparing that to the hash value stored in the compressed frame buffer. In the absence of these techniques, a full screen video window would write every pixel every other screen refresh period or so. Such a consistent and high rate of pixel writing greatly reduces the effectiveness of frame buffer compression.

Frame buffer compression is dramatically less effective with 3-D graphics. Unlike 2-D applications, 3-D applications involve smoothly shaded polygons. Furthermore, continuous movement of the background is common. These two characteristics mean: 1) compressing the frame buffer is harder since there are gradients of color across much of the screen, 2) many more pixels are written every screen refresh interval, and 3) many more pixels change value from one screen refresh to the next.

However, the ineffectiveness of frame buffer compression in a 3-D environment is a moot point: the high memory bandwidth demands of 3-D graphics make it unlikely that midrange to high-performance 3-D systems will use a UMA architecture in the first place.

UMA to enter the market at the low end

As long as total memory bandwidth is constrained by the use of fast-page-mode and EDO DRAMs, UMA architecture will be viable only in low-end PCs. This is because low-end PCs have reduced display resolutions and color depths compared with their midrange and high-end cousins. The low-end marketplace is also the domain where UMA's cost savings are the most significant, the performance penalty is least important to the consumer, and the total bandwidth requirements are lowest. Over time—as higher bandwidth memories become more available and cost-effective—UMA machines are likely to migrate up into the middle of the PC product range. The high end's specialized graphics requirements mean UMA is unlikely to penetrate this market segment in the next few years.

Microsoft actively discourages UMA systems

But the biggest barrier inhibiting the introduction of UMA systems is Microsoft. In 4Q1995 Microsoft began strongly discouraging the PC system vendors from introducing UMA-based systems. The main reason for Microsoft's stance was the performance degradation due to the reduction in main memory size. Windows 95 is tuned for 8 Mbytes of main memory—not 7 or 7.3 Mbytes. But also of significance is Windows 95's ability to adapt to changes of screen resolution on the fly, and its inability to deal with changes in main memory size without a full reboot. Unfortunately, in UMA systems these two go together since the size of main memory and the size of the frame buffer sum to a constant. Microsoft is also very concerned about the success of the PC as a game platform. Games using the DirectDraw and Direct3D APIs make full use of the available off-screen frame buffer store to improve rendering performance. In UMA systems, the tendency is to keep off-screen frame buffer storage to a minimum in order to maximize the size of main memory. The bottom line is that at least in North America, the significant adoption of UMA in the PC marketplace will wait until Microsoft gives its blessing.

Intermediate fill frequencies

UMA memory systems lie between traditional main memory and frame buffer memories in terms of design difficulty. That is because UMA's bandwidth requirements are essentially the sum of the main memory and frame buffer requirements while the granularity requirements are comparable to those of main memories for traditional PCs. Consequently, peak fill frequencies for low-end UMA machines are in the range of 70–100 Hz: higher than the 20–30 Hz of traditional PCs, but less than the 100–300 Hz of typical graphics adapters.

Bus Cntl	Bus Data								
	8	7	6	5	4	3	2	1	0
Start	Op0	Addr9..2							
Op1	Op3	Addr17..10							
OpX1	Addr26..18								
Op2	Addr35..27								
OpX2	Rsrv	Count6,4,2			Rsrv				
Close/Rsrv	Rsrv	Count7,5,3			Count1,0		Addr1,0		

Figure 8-8 A Rambus Request Packet takes 12 ns on the bus.

The most noteworthy aspect of the Request Packet is the unusual placement of the bits within the various bytes. Several reasons exist for this particular placement of the fields. First, it is important to keep related information physically adjacent within the DRAMs. For example, by interleaving count bits the way they are, a full 8-bit count field can be constructed within the RDRAM without having to bus any bits across a large distance. A second reason for the unusual field placement is history: first-generation 4.5-Mbit RDRAMs did not support OpX[2..1], Addr[1..0], or Count[1..0].⁹

Once a slave recognizes a read or write to its address space, its first job is to acknowledge receipt of the request. This is done with a 2-bit Acknowledge Packet. The slave places this packet on the BusCntl line at a fixed time after the end of the Request Packet. As with most of the fixed delays inherent in the protocol, the duration of the delay is specified in the Delay register. Some of the delay values must be programmed identically in all slaves, while others can vary within established windows from device to device on one Rambus Channel.

The responding slave has two choices with respect to the Acknowledge Packet: it can positively (Ack) or negatively (Nack) acknowledge the request. An Ack tells the master that the slave is currently able to handle the request. In the case of a read, the data follows forthwith. A Nack tells the master that the slave is currently unable to complete the request, and that the master should reissue the request later.

9. See Section 8.2.6 for further discussion of 4-Mbit RDRAMs.