Chapter 5
Platform Memory Technology

In this chapter we continue our examination of the platform context in which contemporary microprocessors function. Here we focus on memory technology, an absolutely critical technology for high-performance platforms. Our study will begin with a survey of the memory types most common to new platforms (Fast Page Mode, Extended Data Out and Synchronous DRAM). After a brief introduction to emerging memory types (Direct Rambus and SLDRAM), we then proceed to examine memory controllers for main memory in some detail. The memory controller material will discuss the differences in support for the asynchronous and synchronous memory types, consider the consequences of multiple memory banks, and detail memory controller low-level functions. Such functions include the state machine control of memory timing signals, address processing, data routing and staging, and programmable configuration of DRAM type and size.

ROAD MAP OF CHAPTER 5

Section	Audience
All major headings in the chapter	All

The following more detailed subsections:
 	
Basic Memory Technologies
    Asynchronous DRAM
    Synchronous DRAM
	Students and those unfamiliar with the internal distinctions between Fast Page Mode, Extended Data Out, and Synchronous DRAM.

ROAD MAP OF CHAPTER 5

Section	Audience
Emerging Memory Technologies Rambus Synchronous-Link DRAM (SLDRAM)	Students and those unfamiliar with emerging packet protocol and terminated transmission line based memory interconnect.
North-Bridge Memory Controller Overview FPM, EDO, and SDRAM Support Consequences of Multiple Banks of DRAM Memory Controller Functions	Students and those unfamiliar with memory controller internal operations.

Basic Memory Technologies

DRAMs have traditionally used clockless interfaces. Such asynchronous DRAMs were used in all PC platforms until only recently, when high-end platforms began using DRAMs having clocked interfaces. Having a clocked interface simplifies the control of the DRAM while making it easier to optimize the performance of accesses with main memory. Improvements in main-memory performance are necessary to support continued increases in transfers rates between main memory and PCI-Bus peripherals and between main memory and the AGP. Improved main-memory data-rates are also necessary to reduce latencies on cache line replacements and thereby support continued increases in instruction fetch and data transfer rates of the processor. Consequently, synchronous DRAMs are now generally supplanting the use of asynchronous DRAMs in new platforms. However, the industry is still in a state of transition between these two types of DRAM, and present memory controllers support both types.

Asynchronous DRAM

Page Mode, Fast Page Mode, and Extended Data Out DRAM are minor variations of asynchronous DRAM. These devices consist of little more than a core memory array with row and column selects generated by address decoders. Figure 5.1 is a typical asynchronous DRAM block diagram. Figure 5.2 is an abstract view of the DRAM analog Core that makes up the memory array block.

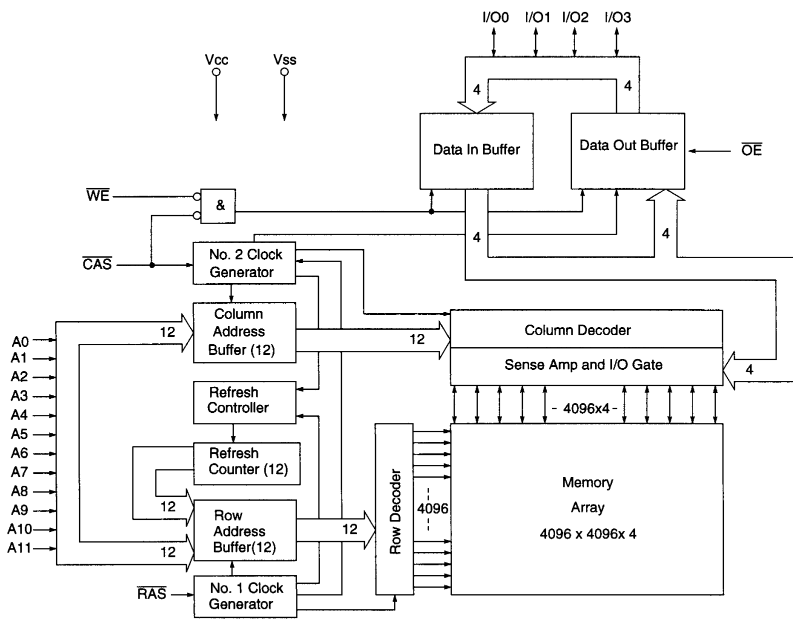


Figure 5.1 ASYNCHRONOUS DRAM BLOCK DIAGRAM

Reprinted by permission from *16M x 4 12/12 EDO DRAM*, Copyright 1996, by International Business Machines Corporation.

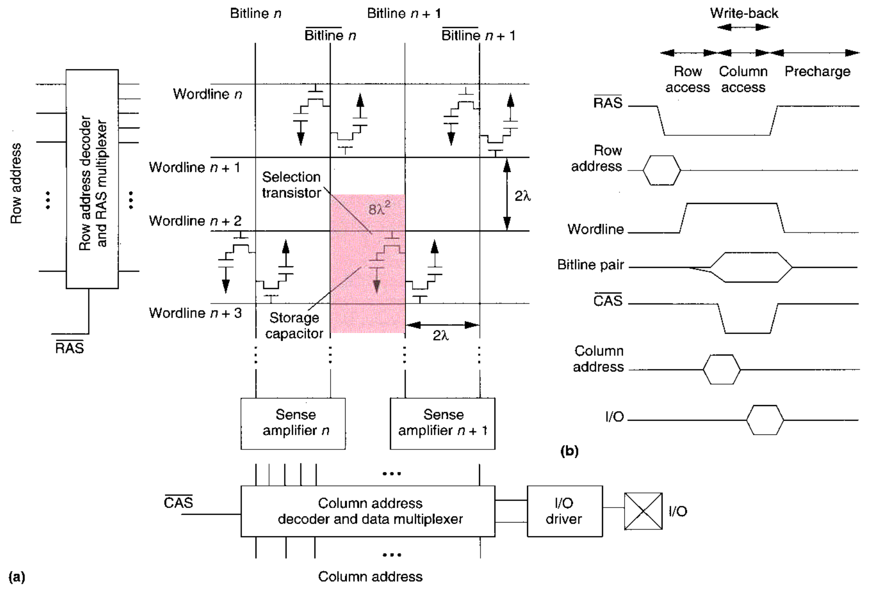


Figure 5.2 ABSTRACT VIEW OF DRAM ANALOG CORE

Copyright 1997, *IEEE Micro*, Nov/Dec 1997, used with permission.

Idiosyncrasies of the DRAM Core

All Dynamic RAM have a basic core of analog-esque functional circuits. Central to this core is a rectangular array of capacitor-based storage cells in which the stored charge must be periodically refreshed. DRAM chips have traditionally parsed the overall memory address into "row" and "column" components, and multiplexed these components over a reduced pin interface. The row address and column address normally correspond to higher order and lower order address bits, respectively. The row address is often thought of as representing a page within the memory, while the column address represents locations within the page. *Row- and column-address strobes*, generally known as *RAS!* and *CAS!*, respectively, are active-low control signals used to latch the row and column addresses within the DRAM for accessing the internal rectangular storage matrix. Write-enable, WE!, is an active-low control signal used to indicate that data is to be written into the addressed location. If WE! is deasserted, data is read from the addressed location. A data input/output signal, MD (also I/O or DQ), conveys the data input to and output from the DRAM for writes and reads, respectively.

Row- and column- address strobes

RAS! and CAS!

During a row access interval, assertion of RAS! activates one *wordline* in the array corresponding to the decoded row address. Since the activated wordline is merely the unencoded row address, the activated wordline and its associated storage cells correspond to a currently addressed page in the memory. Wordline activation enables every storage cell in the addressed page to be coupled to its adjacent *bitline*. During row access, the coupled storage cells perturb the bitlines from specially precharged voltage levels. Due to charge sharing between the necessarily small capacitance of the minimum-geometry storage cells and the large parasitic capacitance of the lengthy bitlines, the impact of the stored charges on the bitline voltages is small. During a column access interval, assertion of CAS! activates a *sense amp* on each bitline to quickly produce a voltage level suitable for output to other circuits. The column address is decoded to select with a multiplexer which column's data is coupled to the memory's I/O.

wordline

bitline

As used in DRAM, a sense amp is a dynamic logic circuit that has two terminals coupled to cross-coupled circuitry. Either its two terminals are explicitly coupled to differential signal lines (as is done in Figure 5.2 on page 429), or one terminal is explicitly coupled to a signal line and the other implicitly coupled to a reference voltage. During a *precharge* time interval, corresponding to the deassertion of RAS!, every sense amp and its associated bitline is *precharged* to voltage levels ideal to their operation during the subsequent row access and column access intervals. This bitline evaluation by the sense amp during the column access also serves to *refresh* (restore, or write-back) the charge in the storage cell. Since all bitlines in a row are precharged and evaluated simultaneously, any access within a page acts to refresh all storage cells within the page.

Page Mode DRAM

The overall *cycle time* for back-to-back memory cycles is the *access time* (the combined row and column access intervals) plus the precharge interval. It is possible in certain DRAM to accesses multiple locations anywhere within the same page by changing the column address (by strobing CAS!) while leaving the row address unchanged (and keeping RAS! asserted). Such accesses merely select a different, but already evaluated, bitline for coupling to the output drivers. Thus, same page accesses do not require the precharge and row access intervals. Consequently, the cycle time of such *page-mode accesses* is reduced. The capability to exploit page-mode accesses requires the memory controller to alter dynamically its RAS! and CAS! generation based on access patterns. A *Page Mode (PM) DRAM* is a DRAM that permits page-mode accesses, which has been a baseline requirement in PC platforms since page-mode support became common in highly integrated memory controllers.

cycle time
access time

page-mode accesses

Fast Page Mode DRAM

In both original PM DRAM and later Fast Page Mode (FPM) DRAM, CAS! activation serves to latch the column address, enable sense amp evaluation, and enable the data output buffer. The difference between these parts lies in when column address decoding is begun. PM parts do not couple the multiplexed row/column address inputs to the column decoders until CAS! is activated. In FPM parts, a transparent (type D) latch clocked by CAS! couples the output of the address multiplexer to the column address decoders. The output of the row/column address multiplexer flows through the latch, with CAS! deasserted, and its value is captured, when CAS! is asserted. As a result, the necessary delay for column address set-up begins from stable and valid column addresses, and is overlapped with the end of the row access interval. Thus, column address decode begins earlier than the later CAS! activation of the earlier PM parts, which reduces the column access interval, and hence the cycle time and the page-mode cycle time.

Extended Data Out DRAM

The relationship of the output drivers and the CAS! signal is a key difference between FPM DRAM and *Extended Data Out (EDO) DRAM*. In FPM DRAM, the column data multiplexer is coupled directly to the bidirectional transceiver buffers that drive the I/O lines. On reads, the output drivers are only active during active CAS! and CAS! must therefore be kept active from assertion until the end of the memory cycle. Because the column address decoders and column data multiplexer include dynamic circuitry that is precharged between CAS! assertions, there is a minimum interval during which CAS! must be held inactive. Because CAS! must be held active in FPM parts until the end of the memory cycle, in order to enable the output drivers, the minimum CAS! precharge time can delay CAS! assertion for subsequent accesses.

In EDO DRAM, an output transparent latch clocked by the compliment of CAS! couples the column data multiplexer to the I/O buffers. The output of the column data multiplexer flows through the latch, with CAS! asserted, and its value is captured, when CAS! is deasserted. The output drivers are designed to go off *only when both CAS! and RAS! are deasserted*. Bringing CAS! inactive by itself, does not turn off the output drives and the data remains valid until the next falling edge of CAS!. The ability to return CAS! inactive while continuing to transfer data enables the CAS! precharge for the next column access to begin earlier and to overlap with the valid data output for the previous column access. As a result, EDO cycle timing is generally one CPU CLK faster than the cycle timing for FPM DRAM. In addition, the output data is generally valid for larger portion of the memory cycle than with FPM parts.

REPORT ON CD-ROM

To learn more about EDO see the IBM Application Note, *EDO (Hyper Page Mode)*, August 27, 1996, on the companion CD-ROM.

Accessing FPM and EDO DRAM

For asynchronous DRAM, the DRAM's clockless analog core is essentially directly accessed. The FPM and EDO inputs may directly (or with minor buffering or inversion) act as clock strobes of internal latches (for the row and column addresses), act as unlatched inputs to combinational functions (address decode for row select), control pass-transistor signal routing and power-activation, act as level-sensitive enables for gating other control signals, or activate dynamic circuit operations (such as column sense-amp evaluation or pre-charge). RAS!, CAS!, the multiplexed row and column address components, and WE!, are output from the memory controller to the DRAM. Memory data (MD) is input or output from the memory controller, for writes and reads, respectively. Figure 5.3 illustrates these signals.

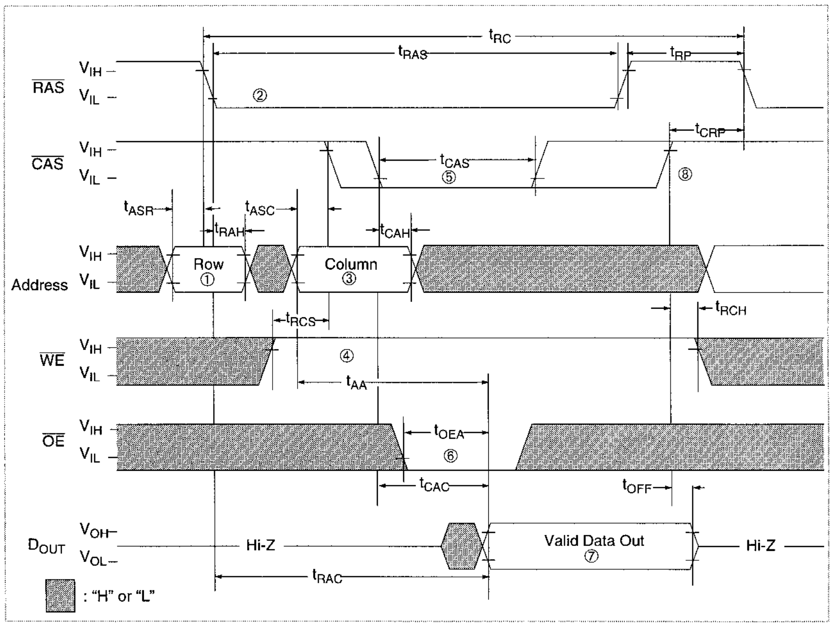


Figure 5.3 SIMPLIFIED ASYNCHRONOUS DRAM READ TIMING

Reprinted by permission from *Understanding DRAM Operation*, copyright 1996, by International Business Machines Corporation.

Synchronous Main Memory Access

Does use of asynchronous DRAM in PC platforms mean that the main memory is being operated asynchronously? The answer depends on where you look and which definition of asynchronous[74] you use. With asynchronous DRAM, there is no explicit clock coupled to the memory chips, the DRAM access and cycle timings are multiples of the CPU clock, the DRAM access timings vary depending on the history of accesses, and the initiation and control sequencing of DRAM accesses must be dynamically modified due to various time-out intervals. The above factors might suggest to some that the main memory is operated asynchronously.

Other factors suggest that asynchronous DRAM are operated synchronously in PC platforms, in spite of the foregoing. The memory controller's RAS! and CAS! strobes are generated synchronous to the CPU clock, and their transitions are used to effectuate the internal operation of the DRAM. Addresses and optional write data and write enables are set up and held synchronous with the CPU clock for a pre-programmed and predetermined number of CPU clock periods. Likewise, read data is always clocked into the memory controller synchronous with the CPU clock at

[74] See the Protocol section in Chapter 4.

the end of a preprogrammed and predetermined number of CPU clock periods. Finally, in spite of its demanding idiosyncrasies, asynchronous DRAM always operates in a fashion that is completely predictable by the memory controller, so there is no need for a handshake mechanism for the DRAM to dynamically alter the predetermined DRAM timing.

ARTICLES ON CD-ROM

To learn more about Asynchronous DRAM, see the following IBM documents on the companion CD-ROM:

- *16M x 4 12/12 EDO DRAM Data Sheet.*
- *Understanding DRAM Operation.*

SYNCHRONOUS DRAM

Synchronous DRAM (SDRAM) interposes a modest but higher-level encoded and clocked interface between the memory controller and the core, which in SDRAM is generally organized and operated as two or more interleaved arrays. The SDRAM-clocked interface internally generates the low-level unencoded direct control of the core as part of its execution of any of a rich repertoire of commands available to the memory controller. The SDRAM interface causes a minor increase in latency on acceses to random off-page locations compared to the FPM and EDO parts. In return, however, the clocked command interface permits system optimizations through access pipelining, long highly efficient burst transfers, and the shifting of responsibility for DRAM-specific low-level control details to the DRAM and away from the memory controller.

In contrast to asynchronous DRAM, SDRAM isolates the analog core behind a clocked interface. Generally, an SDRAM core is similar in technology to the FPM and EDO parts, but employs two or more internal interleaved arrays instead of one. SDRAM inputs and outputs are isolated from the inner core via clocked latches, and the control inputs are encoded (have defined combinations) to represent a rich variety of commands that the control interface translates into internal signals that control the core. The control interface generates the internal control signals using its own internal state machine controller. It is the SDRAM interface logic's responsibility to meet the setup and hold times of all the SDRAM basic core control signals. The interface carries out commanded accesses, including internal generation of addresses for burst transfers as required. SDRAM can perform burst transfers up to a page in size without signaling after command initiation. If necessary, burst transfers can be aborted prior to completion. The interface performs access pipelining in that it can accept the starting address and command for a second access while executing a first access command. The refresh function can be automatically or semi-automatically performed by on-chip circuitry ancillary to the interface logic.

Reports on CD-ROM

A wealth of information on SDRAM is contained in the IBM 64Mb Synchronous DRAM Data Sheet, included on the CD-ROM. In particular, see:

- A block diagram, on p. 6, which illustrates this part's internal 4-bank organization;
- A functional description of each I/O pin, on p. 4;
- Definition of control fields for the operating mode and other control registers, on p. 9;
- A truth table for the encoded command functions, on pp. 30-32;
- A truth table for the internal state machine, on pp. 33-36; and
- 19 timing diagrams of various operations, on pp. 44-62 (a list of these appears on p. 43).

Emerging Memory Technologies

Rambus and SLDRAM are emerging memory design technologies that incorporate a more sophisticated system-level perspective that tightly couples memory and memory controller operation. These technologies incorporate many advanced technologies, including clock forwarding to eliminate clock skew, special attention to packaging and board design to minimize transmission line effects in optimized voltage signaling, highly efficient packetized command protocols, split-transaction operation, and sending and receiving data on both clock edges.

Articles on CD-ROM

To learn about the factors motivating these and other DRAM technologies, see "Trends in Semiconductor Memories," by Yasunao Katayama, *IEEE Micro*, 1997, included on the companion CD-ROM.

Rambus

Rambus Channel

Rambus Interface

Rambus DRAM (RDRAM)

Rambus is a technically sophisticated, multi-disciplined, proprietary approach to implementing highly optimized memory systems for board-level platforms. Rambus systems consist of masters and slaves coupled to a *Rambus Channel*, each via its own *Rambus Interface*. Masters can be microprocessors, accelerators, memory controllers, or other core logic or ASIC devices. The prototypical master is a memory controller. Slaves can include any of the obvious memory types such as DRAMS, SRAMs, and Flash EPROM, or other devices such as RAMDACs. The prototypical slave is a *Rambus DRAM (RDRAM)*, which couples a Rambus Interface to one or more banks constructed from conventional DRAM cores. Figure 5.4 shows a simplified view of Rambus DRAM, coupled via a Rambus Interface to a Rambus Channel. Rambus memory systems offer a fundamentally different approach to platform design. Memory controllers are designed to the Rambus Channel specs, not for specific DRAMs. RDRAMs in turn are expected to meet Rambus Channel specs for timing and pinout.

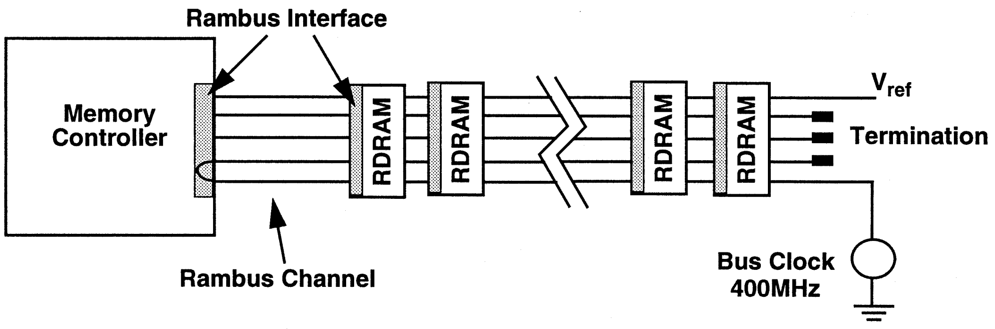


Figure 5.4 Primary Elements of a Rambus-Based System

Copyright 1997, Courtesy Rambus, Inc.

Direct Rambus Channel

In low-end platforms in particular, RDRAMs provide high bandwidth using small packages and pin count relative to other DRAM technologies. In the present version of Rambus, known as Direct Rambus, RDRAMs are coupled to the *Direct Rambus Channel* via a two-byte width of 16 or 18 bits. The Direct Rambus Channel and data transfers are made on both edges of the Bus Clock. As a result, RDRAMs have a burst transfer rate in MB/sec that is four times the clock rate in MHz. The present clock rate is 400MHz, yielding a peak data transfer rate of 1.6 GBps.

Features and Philosophy

The Rambus Channel is at the center of Rambus-based designs. The channel efficiently communicates between masters and slaves at high data transfer rates using a split-transaction packet protocol that bundles control, addressing, and data into transaction packets that are physically transported over a narrow bus. Controlled-impedance transmission line techniques are used to achieve predictable bus delays. Sophisticated synchronous clocking methods guarantee minimal skew. Primary Channels are designed to support up to thirty-two RDRAMs. A bridge called an RTransceiver may be used to increase the effective number of slaves. Up to ten Secondary Channels (each supporting up to thirty-two RDRAMs) may be coupled to a Primary Channel via such bridges. Thus a Primary Channel can accommodate up to 320 RDRAMs through the use of bridged Secondary Channels.

The Rambus Channel also has an associated philosophy that includes a design philosophy and a design and implementation discipline. Rigid adherence to the Rambus Channel philosophy ensures consistent and reliable high bandwidth in systems making use of memory controllers and DRAM from multiple vendors. A key aspect of the Rambus philosophy is that the channel's data width is intended to be held narrow regardless of memory population. Bandwidth is instead scaled by use of multiple channels or through increasing the clock frequency. The architecture of the basic channel is intended to be relatively constant over multiple generations. Similarly, system designers are discouraged from making independent modifications to the bus architecture.

Details of the Rambus Channel

The Rambus Channel achieves its high bandwidth partly by signaling over high-quality transmission lines. The topology of the channel connections, shown in Figure 5.5, plays a key role in achieving quality characteristics. The master is located at one end of the channel. At the other end of the channel, terminators are used, matched to the same characteristic impedance as the signal traces. Slaves are placed only between the master and the terminators. A disciplined implementation also contributes to the transmission line quality. The Rambus Channel is intentionally a dense/narrow bus, having only thirty-five active signals. Figure 5.1 details all signal types on the Rambus Channel. All signals are carried via controlled impedance traces, free of discontinuities. Short, dense packaging with minimized parasitics is employed. All trace loading is uniform in distribution and matched in value.

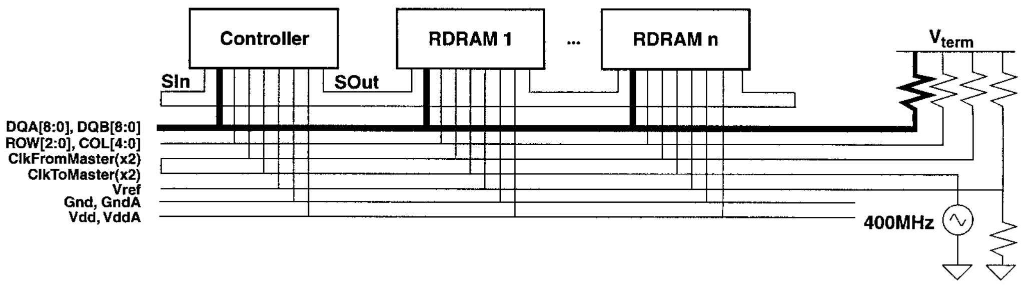


Figure 5.5 Topology Details of the Rambus Channel

Copyright 1997, Courtesy Rambus, Inc.

Table 5.1 Rambus Channel Pins and Signal Types

Pin	Signal Type and Notes
DQA, DQB (16 to 18bits)	"active" RSL (low-voltage Rambus Signaling Logic); data
ROW	"active" RSL; row commands
COL	"active" RSL; column commands
ClkToMaster (and compliment)	"active" RSL; differential sync for data from slaves
ClkFromMaster (and compliment)	"active" RSL; differential sync for data from master
SIn and SOut	"active" TTL; address configuration and refresh
Vref	Voltage reference for all RSL signals
Gnd, Vdd	power
GndA, VddA	"analog" power for PLLs

Rambus Signaling Logic (RSL)

The proprietary active-low *Rambus Signaling Logic (RSL)* defines the signaling protocol of the Rambus Channel. As shown in Figure 5.6, low voltage swings (800-mv) are differentially sensed about a bused Voltage Reference signal (Vref) of approximately 1.4V. Logic 0 corresponds to the voltage of the termination resistors (Vterm!), which is roughly 1.8V. Logic 1 corresponds to a VOL of approximately 1.0V. The differential sensing provides common-mode noise immunity, while the low voltage swings result in reduced EMI and less ground bounce.

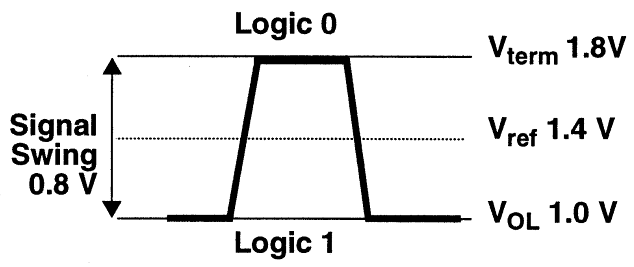


Figure 5.6 RAMBUS CHANNEL SIGNALING

Copyright 1997, Courtesy Rambus, Inc.

The Rambus Channel achieves virtually skewless data clocking by propagating clocks and data in parallel, down matched transmission lines. A clock signal known as ClkToMaster originates from a clock generator on the terminator side of the Channel. ClkToMaster is propagated down a transmission line toward the master end of the Channel. Slaves use the ClkToMaster signal, referred internally to the Slave as the TxClk, to synchronize data being sent to a master. The ClkToMaster is looped back at the master and propagates anew down a separate transmission line as a signal known as the ClkFromMaster. Masters use the ClkFromMaster signal to synchronize data being sent to slaves. Slaves use the ClkFromMaster signal, referred internally to the Slave as the RxClk, to synchronize data being sent from a master. Each master or slave on the Channel has separate PLLs for extracting internal clock signals synchronized to the ClkToMaster and the ClkFromMaster.

The Rambus Interface

Integral to all masters and slaves is the Rambus Interface, the I/O logic and circuitry which enables coupling with the Rambus Channel. Rambus makes available to licensees a reference Rambus Interface module, known as the *Rambus ASIC Cell (RAC)*. Figure 5.7 illustrates a generic Rambus Interface between a Rambus Channel and a Rambus-based memory controller. The interface module shown presents the memory controller with a simplified 128-bit (with 144-bit "9th-bit byte" option) interface operating at 100MHz. Whether based on the RAC, or based on a custom design, the Rambus Interface must perform a number of specific functions. Electrically, it performs signaling conversion between the RSL voltage levels and the voltage levels required by the CMOS-core of the attached device. The PLLs, which synchronize the transfer of data with the off-chip Rambus Channel propagated clocks, also reside in the Rambus Interface. Logically, the Rambus Interface generates for masters, and decodes for

Rambus ASIC Cell (RAC)

slaves, the request packets that are key to the master/slave packet protocol. Likewise, the acknowledge protocol is implemented here, and includes the proper handling of multiple byte transfers. For memory array slaves, the Rambus Interface also selectively couples the addressed memory location from the array sense amp latch to the channel.

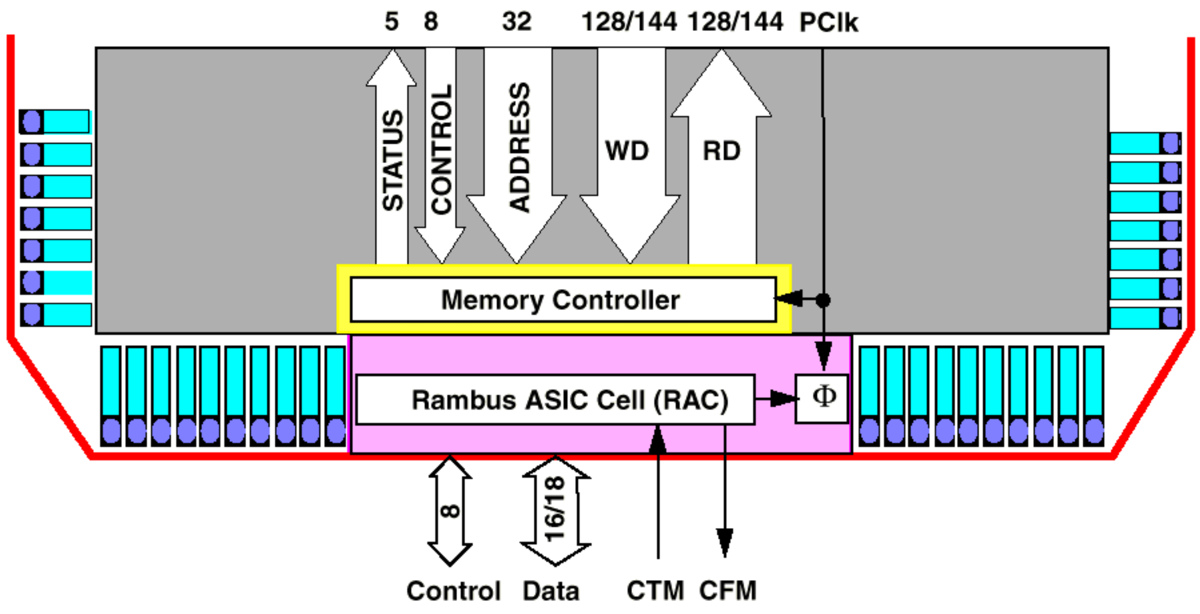


Figure 5.7 Example Rambus Interface with Memory Controller

Copyright 1997, Courtesy Rambus, Inc.

Other Rambus Architectural Features

Rambus devices support a register space having both programmable control registers and read-only status registers. Read-only registers provide such information as device type, size, and manufacturer's ID. Programmable control registers enable system designer configuration of address mapping, power management, refresh, and transaction timing. Rambus devices also are linked together via a serial daisy-chain using the SIn and SOut pins. This link is used to configure the base memory address of each slave during initialization, and to initiate system refresh requests during power down.

Pros and Cons of Rambus-based Systems

A principal factor in the appeal of designing a platform to use Rambus is its characteristic high bandwidth in small capacity and small pinout memory systems. A high ratio of bandwidth-to-memory capacity reduces the need to widen the memory system with additional memory solely to achieve a desired level of memory bandwidth. Such additional memory may increase the memory capacity and the associated cost of the base system configuration beyond what is otherwise necessary. A high ratio of bandwidth-to-memory capacity also reduces the need for special memory architectures, which might include SRAM caches or VRAM, depending on the application. Rambus's high ratio of bandwidth-to-memory capacity in small capacity systems makes it particularly well suited to dedicated video memories and low-end *Unified Memory Architecture (UMA)* platforms, in which the video memory is just a region within the main memory and not a dedicated subsystem.

Unified Memory Architecture (UMA)

Another factor in the appeal of Rambus relates its narrow bus width. Rambus's high ratio of bandwidth-to-pinout results in smaller packaging and routing without the need to resort to special packaging technologies, such as Multi-Chip Modules (MCMs). The low pin count associated with the narrow bus also permits a Rambus-based memory controller to be integrated more readily with other logic. Thus as processing technology advances, systems using Rambus have an advantage over other system approaches in combining system functions into fewer chips. Also, due to the byte-wide organization, memory can be added to a Rambus system with finer granularity (a smaller increment in capacity) and correspondingly lower cost, compared to a more typical 4- or 8-byte wide memory bus organization.

In contrast to the foregoing, Rambus's appeal diminishes with increases to the slave population of the Rambus Channel. Since the bandwidth of the Rambus Channel does not scale with the number of RDRAMs, there is a decrease in the ratio of bandwidth-to-memory capacity as the number of RDRAMs increases. Furthermore, as the number of masters and slaves increases, there is increased likelihood for contention for the single narrow bus. Finally, while not an issue with only a few masters and slaves, larger systems must deal with the cumulative power dissipation from the multiple PLLs in each chip's Rambus Interface.

Additional information about Rambus technology and the company is accessible at http://www.rambus.com/.

Article on CD-ROM

To learn more about Direct DRAM, see the article "Direct Rambus Technology: The New Main Memory Standard," by Richard Crisp, *IEEE Micro*, 1997, on the companion CD-ROM.

TECHNICAL PRESENTATION ON CD-ROM

	The following presentation gives an overview of the original Rambus architecture: *Rambus and Direct DRAM,* by Bennett Smith.

SYNCHRONOUS-LINK DRAM (SLDRAM)

Synchronous-Link DRAM (SLDRAM), like Rambus, is a high-performance interface for efficiently coupling DRAM with memory controllers. Unlike Rambus, SLDRAM is an open standard (IEEE Standard P1596.7), developed by a consortium of memory manufacturers that promote its use. This is in contrast to the proprietary Rambus interface, which requires the payment of licensing or royalty fees, or other compensation, for its use in either DRAMs or memory controllers.

Additional information about SLDRAM technology and the SLDRAM organization is accessible at http://www.sldram.com/.

SLDRAM has a clocked handshakeless interface, terminated low-voltage signaling, and a packet protocol. It uses burst-mode transfers, returns data on both clock edges, and is designed to make effective use of memories having multiple internal banks. SLDRAM uses a unidirectional *CommandLink* bus for command and address packets and a bi-directional *DataLink* bus for read and write data packets. SLDRAM is a parallel interface variant of *RamLink (IEEE Standard 1596.4)* a point-to-point memory interface standard. RamLink, in turn, is a memory specific reduced command set variant of the *Scalable Coherent Interface* protocol *(SCI, IEEE Standard P1596)*.

ARTICLE ON CD-ROM

	A detailed overview of the SLDRAM architecture an operation is provided in the Peter Gillingham's and Bill Vogley's article "SLDRAM: High-Performance, Open-Standard Memory," *IEEE Micro*, November/December 1997, which is included on the companion CD-ROM.

Reports on CD-ROM

We have included two reports about SLDRAM on the companion CD-ROM. The first item is an SLDRAM data sheet. The second item is an SLDRAM white paper.

- *4M x 18 SLDRAM Pipelined, Eight Bank, 2.5V Operation*.
- *SLDRAM Architectural and Functional Overview*, Peter Gillingham, MOSAID Technologies Inc.

Standard on CD-ROM

Included on the CD-ROM is the *Draft Standard for a High-Speed Memory Interface (SyncLink)*, IEEE P1596.7-199X. This is the proposed standard for the SLDRAM Bus.

North-Bridge Memory Controller

Byte Addressing: In PC platforms, objects being accessed are conventionally referenced by specifying the byte address of the least significant byte of the object. This is so, regardless of the objects' width, the data transfer width, or the width of datapaths that subsequently process the referenced object.

The control of main memory is fundamental to the operation of any computer system. Main memory can be viewed as an array of locations having unique addresses. Instructions and data for both system software and applications are loaded into and executed from the main memory. The main memory needs to support basic data transactions. Requesters may perform transactions on specified main memory locations that read (fetch data from), write (store data to), and read-modify-write (a non-interruptible sequence of fetching data, manipulating the fetched data, and storing the manipulated data back to the original location).

The data transfer rate and latency between main memory and the cache subsystem or the CPU is crucial to system performance. In high-performance computers, the memory controller is preferably located in close physical proximity to the CPU, caches, and a high-bandwidth peripheral bus, due to its close interaction with these subsystems.

Overview

In PC platforms the memory controller is a major functional block within the North-Bridge. Figure 5.8 on page 445 is a simplified block diagram of an AMD-640 System Controller, an illustrative North-Bridge (note the DRAM Control block and I/O signals in the lower right corner of the figure). The memory controller receives transactions requesting data transfers between the main memory and the request source. The main memory is implemented with multiple banks of DRAM, which share a common data bus and multiplexed address bus (and other common signals), but have separate bank enables. The data, address, and control interface to the DRAM banks is a major port of the North-Bridge. Figure 5.9 on page 445 illustrates the control interface of an example configuration of the AMD-640 using six banks of SDRAM.

Articles on CD-ROM

Throughout this section we will provide memory controller implementation examples based on a commercially available North-Bridge, the AMD-640 System Controller. The data sheet for this part is included on the companion CD-ROM. The reader is encouraged to review the data sheet for additional information not directly included in this chapter. In particular, see:

- Section 4.3, *DRAM Interface Signals*, on pp. 4-8 and 4-9, which gives a short description of each signal;
- Section 5.3, *DRAM Memory Controller*, on pp. 5-15 through 5-29, which discusses operation and provides timing diagrams. Separate subsections examine Mixing Memory, Error Correction Code, DRAM Refresh, Shadow DRAM, EDO DRAM, and Synchronous DRAM; and
- Section 7.5, *DRAM Control Registers*, on pp. 7-16 through 7-31, which provides full details for controlling all aspects of the memory controller.

Design Note

Sources of Main Memory Traffic

Requests for main memory may come from the shared CPU/Cache-bus, the PCI-Bus, or the AGP. These are the three major ports on the North-Bridge, besides the memory controller. The cache controller on the North-Bridge additionally has a control-only port to the cache.

Traffic between the CPU and main memory results from write-through cache configurations, and from accesses to non-cacheable areas of memory. Traffic between the cache and main memory results from cache replacements brought about by cache misses. In write-back cache configurations, cache lines being replaced from the cache must be written back to main memory from the cache, if any data has been modified. The replacement lines in the instruction and data caches are subsequently filled from main memory. Traffic between AGP and main memory results from texture and other image traffic to and from the video memory and from AGP reads of control information left by the processor.

Traffic directly between the PCI Bus and main memory includes at least two types of bus masters transfers. The first type of PCI-Bus/main-memory transfers are OS initiated transfers that swap program data and code between mass storage and main memory as part of normal virtual memory management. The second type of PCI-Bus/main-memory transfers are explicitly initiated transfers by application or system software to move data items or complete files between work areas in memory and mass storage or other I/O devices.

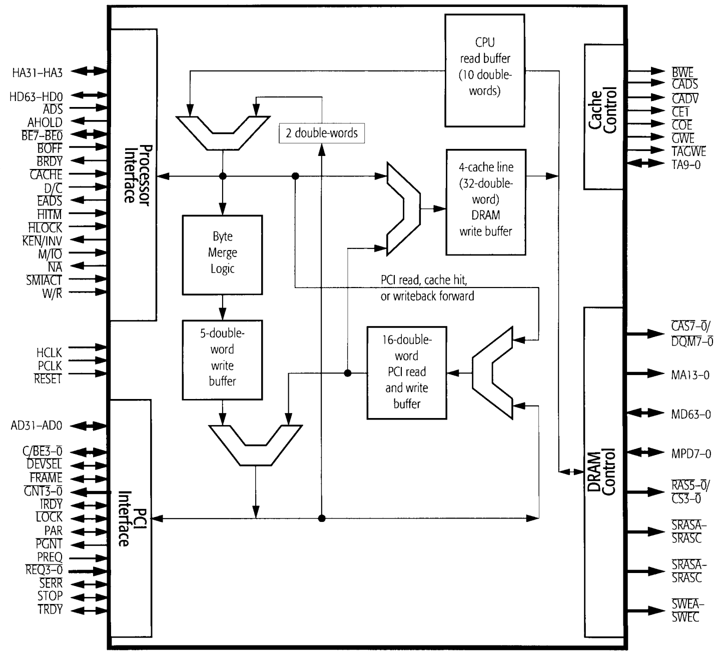


Figure 5.8 The AMD-640 System Controller (A North-Bridge)

Copyright 1997/1998, Advanced Micro Devices Inc., used with permission.

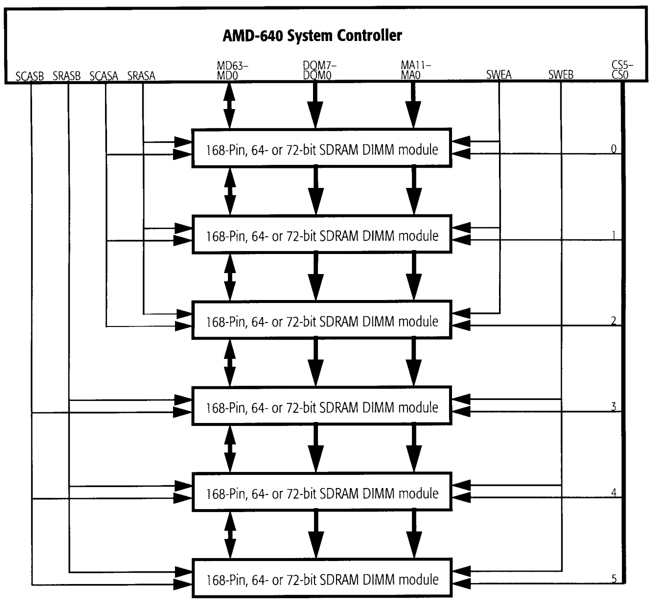

AMD-640 System Controller
SCASB
SRASB
SCASA
SRASA
MD63–
MD0
DQM7–
DQM0
MA11–
MA0
SWEA
SWEB
CS5–
CS0
168-Pin, 64- or 72-bit SDRAM DIMM module
168-Pin, 64- or 72-bit SDRAM DIMM module
168-Pin, 64- or 72-bit SDRAM DIMM module
168-Pin, 64- or 72-bit SDRAM DIMM module
168-Pin, 64- or 72-bit SDRAM DIMM module
168-Pin, 64- or 72-bit SDRAM DIMM module
0
1
2
3
4
5

Figure 5.9 All SDRAM Bank Configuration of the AMD-640

Copyright 1997/1998, Advanced Micro Devices Inc., used with permission.

Requests are processed in a state-machine controller that controls all aspects of the memory controller's behavior, including address processing, data staging, data routing, and the activation and timing of all interfaces. The state-machine controller controls the sequencing and timing for these operations to transfer data between the requesting port and specified main memory addresses, which correspond to predetermined unique locations within the multiple banks of DRAM. Address processing logic maps the transaction addresses to bank (or bank-pair), row, column, and byte-select information, which is used to select the desired bank (or bank-pair); a particular *quartet* (for banks) or *octet* (for bank-pairs); and one or more desired bytes out of the byte-quad (for banks) or octet (for bank-pairs). Data staging is used to buffer all ports (CPU, cache, PCI, AGP and DRAM), reduce latencies via prefetching and posted writes, and enable the operation on each port to generally proceed stall-free, concurrently, and independent of each other. Data routing selectively couples data between the port buffers to accomplish the desired data transfers.

A quartet, and an octet, are a group of 4 and 8 bytes, respectively.

Design Note

Configuration Register Access in the AMD-640

The AMD-640 has a number of configuration registers. These registers are accessed in the PCI-Bus configuration space, using configuration mechanism #1, described in the *PCI Local Bus Specification Revision 2.1*. Loosely speaking, the desired register is identified using an I/O write operation to a 4-byte wide I/O location, and the configuration data is passed using another I/O write to the next subsequent 4-byte I/O locations. This chapter abstracts this complexity and refers to writing the configuration data into a particular numbered configuration register. Each register is referred to by the prefix CR, for Configuration Register, a particular "offset" address in hex, denoted by the letter "h," and at least one bit within the register, specified in brackets. If more than one bit is involved, a bit range will be indicated within the brackets by an upper bit location, a colon, and a lower bit location. As an example, the upcoming usage "CR59h<2:0>" is a reference to the configuration register at offset 59hex, bits 2 through 0, inclusive. It is to be understood that all configuration registers must actually be accessed via the PCI configuration mechanism #1.

While the gross nature of the state-machine controller is fixed, the fine-grain behavior of the state-machine controller, and hence the memory controller as a whole, is programmable via a large number of configuration registers. These configuration registers are accessible to initialization BIOS routines via the PCI-Bus configuration address space. Figure 5.10 lists the DRAM configuration registers in the AMD-640.

Offset	Cache Control	Default	Recommended		Access
			Setting	Result	
58h	DRAM Configuration Register #1	40h	44h	10 bit Col	RW
59h	DRAM Configuration Register #2	05h	03h	banks 0-3 populated	RW
5Ah	DRAM Bank 0 Ending [HA29-22]	01h	10h	64M–02 for 8 Meg	RW
5Bh	DRAM Bank 1 Ending [HA29-22]	01h	20h	64M–04 for 8 Meg	RW
5Ch	DRAM Bank 2 Ending [HA29-22]	01h	30h	64M–06 for 8 Meg	RW
5Dh	DRAM Bank 3 Ending [HA29-22]	01h	40h	64M–08 for 8 Meg	RW
5Eh	DRAM Bank 4 Ending [HA29-22]	01h	50h	64M–08 for no RAM	RW
5Fh	DRAM Bank 5 Ending [HA29-22]	01h	60h	64M–08 for no RAM	RW
60h	DRAM Type	00h	00h 05h	Fast Page Mode banks 0–3 EDO mode	RW
61h	Shadow RAM Control Register #1	00h	CAh	Video BIOS	RW
62h	Shadow RAM Control Register #2	00h	00h	disable	RW
63h	Shadow RAM Control Register #3	00h	22h	main BIOS	RW
64h	DRAM Timing	ABh	FFh 4h 57h	slowest initially 60 nsec EDO 60 nsec FP	RW
65h	DRAM Control Register #1	00h	A4h	Page open Fast decode Latch delay	RW
66h	DRAM Control Register #2	00h	00h		RW
67h	32-Bit DRAM Width Control Register	00h	00h	64 bit DRAM	RW
69h–68h	Reserved	–	–	–	–
6Ah	DRAM Refresh Counter	00h	43h	15 μsec	RW
6Bh	DRAM Refresh Control Register	00h	80h	CBR	RW
6Ch	SDRAM Control Register	00h	00h	–	RW
6Dh	DRAM Drive Strength Control Register	00h	4Fh	24 ma drive	RW
6Eh	ECC Control Register	00h	00h	–	RW
6Fh	ECC Status Register	00h	00h	–	RO

Figure 5.10 DRAM Control Registers in the AMD-640 System Controller

In PC platforms, the memory controller's design is heavily influenced by both technical and competitive requirements including: the use of asynchronous and synchronous DRAM, the organization of the DRAM into multiple banks, strict compatibility with the PC memory architecture, configuration flexibility, ease of system design, and aggressive performance and pricing. High performance is achieved via support for wide transfers and high clock rates and supporting system-level optimizations through improved bus transfer efficiency, reduced access latencies, and enabling system bus concurrencies. Configuration flexibility and ease of system design require that the memory controller hide most of its sophistication from the system designer, platform vendor, and especially the end user.

FPM, EDO, and SDRAM Support

Current memory controllers must be designed to support Fast-Page-Mode DRAM, Extended-Data-Out DRAM, and Synchronous DRAM. SDRAM likely will supplant FPM and EDO DRAM in the near future, due to SDRAM's potential for superior system cost-performance. SDRAM represents a fundamentally new approach to DRAM main memory control that permits focus on system performance improvements rather than details of low-level timing. Presently, however, continued compatibility with FPM and EDO parts is important for mainstream PC platforms. Thus, the platform's memory controller must provide for the combination of the many demanding characteristics of FPM, EDO, and SDRAM. Future memory controllers will likely also support one or both of Rambus and SLDRAM.

All of these DRAM organizations multiplex row and column addresses over shared memory address pins and have multiple CPU-clock latencies for accesses to the first data item at random locations. Use of these parts requires the memory controller to carry out low-level unencoded direct control of the core.

A memory controller for asynchronous DRAM must manage and track a large number of dynamic conditions to ensure the core's proper operation. The timing parameters and their sequential relationship illustrated in the earlier simplified read-timing diagram represent but a subset of the overall DRAM timing parameters that the memory controller must comply with. The controller must implement all aspects of the timing behavior of the core's control signals and the controller's configuration must be programmed accordingly. This includes the coarse-grain inter-signal dependencies as well as the fine-grain cycle-by-cycle setup and hold-time behavior for each signal. Since the external control signals that drive the asynchronous DRAM are not captured via clocked latches, the memory controller must keep the control inputs constant while internal operations propagate through the core, optional data ouputs become valid, and the data is captured by the memory controller. The memory controller is responsible for the refresh function and tracking other time-outs affecting how accesses are implemented. To exploit page-mode timing, an on-going comparison of column addresses must be made to determine if the reduced access timing may be used. Continued CAS! signaling is then required to obtain subsequent data items within a page after the first access.

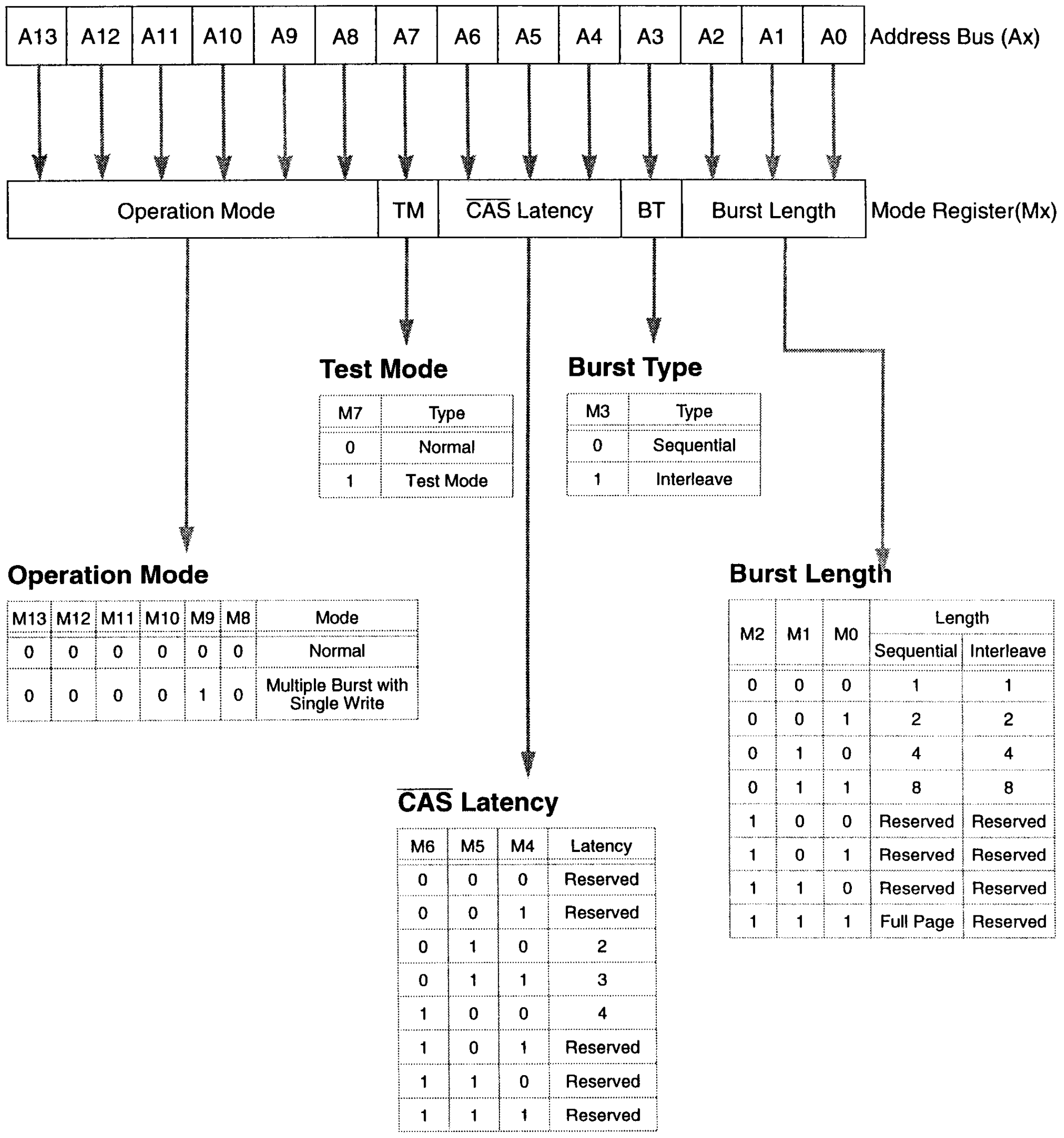
A memory controller for SDRAM must translate transaction requests for main memory into the encoded commands that the SDRAM expects. In contrast to a memory controller for asynchronous DRAM, it need only hold an encoded command valid until it is latched by the SDRAM at the end of the clock cycle. The memory controller can then transmit the next command to the SDRAM, or merely go to the idle state (by transmitting a no-operation command), while the SDRAM performs the first specified operation without further assistance from the memory controller. In conjunction with the SDRAM's internal interleaved bank organization, burst transfers, and by tracking current and past accesses, pipelining of memory accesses permits the memory controller to frequently overlap the beginning of the second access with the end of the first access. The system can thus generally hide precharge intervals caused by accesses to new random page locations. A memory controller for SDRAM must be configured to account for the access latencies (in number of cycles) of the particular SDRAM being used, but such coarse-grain timing is much more straightforward to implement than the demanding fine-grain timing required for the core control signals generated by a controller for asynchronous DRAM. A memory controller for SDRAM must also program the SDRAM upon system initialization to configure the desired behavior from the multiple available operating modes (including burst length, burst increment sequence, and read latency). As illustrated in Figure 5.11, the operating mode is programmed via address bits sent with the Mode Set Command.

Consequences of Multiple Banks of DRAM

PC platform main memory is organized into multiple banks or multiple bank-pairs. This bank-focused organization permeates the memory controller's design and operation. The use of multiple banks permits increased performance through *interleaved*, or overlapping, memory access cycles to more than one bank. Also, selective mapping of row addressing bits, sequential pages in the linear memory address space maybe sequentially alternately distributed to each memory bank. Such *interleaved memory pages* increases the number of page-mode devices generally available to all execution threads. Design for the optional use of multiple banks also allows for memory expansion. To facilitate field and end user upgrades of memory, each DRAM bank is generally permitted to be of different type, density, pin configuration, and speed.

interleaved

interleaved memory pages



Test Mode

M7	Type
0	Normal
1	Test Mode

Burst Type

M3	Type
0	Sequential
1	Interleave

Operation Mode

M13	M12	M11	M10	M9	M8	Mode
0	0	0	0	0	0	Normal
0	0	0	0	1	0	Multiple Burst with Single Write

Burst Length

M2	M1	M0	Length	
			Sequential	Interleave
0	0	0	1	1
0	0	1	2	2
0	1	0	4	4
0	1	1	8	8
1	0	0	Reserved	Reserved
1	0	1	Reserved	Reserved
1	1	0	Reserved	Reserved
1	1	1	Full Page	Reserved

$\overline{\text{CAS}}$ Latency

M6	M5	M4	Latency
0	0	0	Reserved
0	0	1	Reserved
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	Reserved
1	1	0	Reserved
1	1	1	Reserved

Figure 5.11 SDRAM Operating Modes

Reprinted by permission from *64Mb Synchronous DRAM*, copyright 1997, by International Business Machines Corporation.

The platform's memory controller must accommodate design issues that arise from the use of multiple banks in main memory. A first issue is that since it is desirable to dynamically optimize memory cycle times based on the behavior of recent access, and since the behavior of recent accesses is generally unique for each bank, the memory controller must maintain information about the state of each bank and control the timing of the DRAM memory cycles on a bank-by-bank basis.

bank-by-bank timing control

A second issue is that future memory upgrades need to be performed with little regard to matching the parameters of the original DRAM chips with those of the upgrade. It is unreasonable to require an originally installed bank of DRAM to be discarded or physically relocated. Also, since many naïve end users may perform such upgrades, such configuration flexibility greatly increases their success rate and reduces service complaints to the platform vendor. Thus, banks may be of a variety of memory capacities, having a respective variety of row/column address bit widths, and occupying various extents of the memory address space. Additionally, some platform chipsets permit further combinations.

accommodation of different bank capacities and widths

Bank-pairing

The above discussion must be revisited in view of the possible use of bank-pairs. *Bank-pairing* optionally permits two, 4-byte-wide, identical banks to function in unison as one 8-byte-wide bank-pair. This permits DRAM conforming to the older 72-pin SIMM packaging standard to provide double the bandwidth (at increased power consumption) over their use as only 4-byte-wide banks. Unless power consumption is a primary concern, such bank-pairing would normally be enabled. Figure 5.12 illustrates how six banks of 4-byte-wide FPM or EDO DRAM are configured as three, 8-byte-wide, bank-pairs in one example configuration of the AMD-640 System Controller.

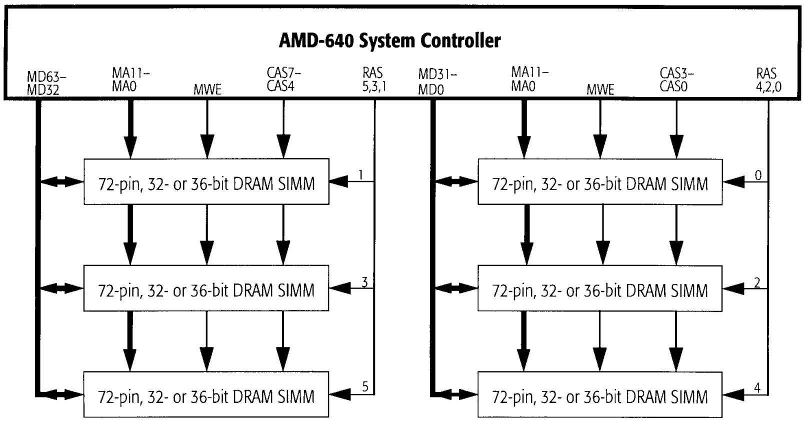


Figure 5.12 BANK-PAIRING

Copyright 1997/1998, Advanced Micro Devices Inc., used with permission.

However, when optional bank-pairing is made available, additional constraints may be imposed on bank use. For example, a unified control configuration may be used for each group of two banks, even though the banks within the group may not be operationally paired. This simplifies the logic design, but imposes the constraint that the DRAM in each bank of a bank-pair must be identical. Of course if bank-pairing is enabled, which it normally is for anything but a mobile application, the two banks should be identical anyway. The major problem of the identical bank constraint, is that while it is possible to populate only one bank of a bank-pair initially, and later correctly populate the remaining bank of the bank-pair, this presumes a level of future recollection, diligence, and care, that may in practice be difficult to achieve.

Design Note

Memory Bank Configurations in the AMD-640

In the memory controller of the AMD-640 North-Bridge, there are six 4-byte-wide banks, which must be populated in contiguous order. The last bank populated is set in CR59h<2:0>*. Banks 0/1, 2/3, and 4/5 are considered bank-pairs. To simplify the logic design, these bank-pairs use a unified control configuration, and as a result, the DRAM in each bank of a bank-pair must be identical.

Bank-pairs consisting of 4-byte-wide FPM or EDO DRAM may be optionally configured for 8-byte-wide simultaneous accesses or separate 4-byte-wide access. Bank-pairing for SDRAM is for simplified control purposes only. The doubling of access width possible for the FPM and EDO parts is not possible for SDRAM parts. When using 8-byte-wide SDRAM, each of such banks may be optionally configured for 8-byte-wide simultaneous accesses or separate 4-byte-wide access.

While the banks within a bank-pair must be identical, different bank-pairs may still be of different data bit widths. (e.g., a first bank-pair may consist of two 4-byte-wide EDO DRAM (which may or may not be operated as a single 8-byte-wide bank-pair) while a second bank-pair may be two 8-byte-wide SDRAM.) Also, each bank-pair might be of arbitrary DRAM technology (e.g., it might be permissible to have a first bank-pair of FP and second bank-pair using EDO.)

This section refers numerous times to banks or access widths that are termed to be either 4 bytes wide or 8 bytes wide. If error checking is used, it is understood that these terms refer to widths of 36 bits and 72 bits, respectively. If error checking is not being used, it is to be understood that these terms refer to widths of 32 bits and 64 bits, respectively.

Memory Controller Functions

There are a multiplicity of functions performed by the PC platform memory controller. Nevertheless, most of these functions can be conceptually partitioned into a few broad categories: the transaction reception from multiple request ports, state-machine controller, address processing logic, data routing and staging logic, I/O interface to multiple DRAM banks, and programmable configuration logic. Each of these categories will be given an initial overview, followed by more detailed discussions in select areas.

Transaction Requests from Multiple Sources

Each of the other major ports on the North-Bridge (i.e., the CPU/Cache, PCI-Buses and AGP) may be the source for a transaction with the main memory. While the exact protocols and capabilities vary among these interfaces, each somehow conveys at least an initial starting address and an indication of the transaction desired (e.g., a bus "command" control signal). Depending upon the particular bus attached, the main memory-to-bus interfaces may need to optionally support burst transfers, transfers of reduced width, and possibly other operations, such as touching, masking, and other modest manipulations (that are generally performed on the data in transit via selective transmission or routing and do not require logic that would introduce perceptible delays). Requests that involve writes will additionally include the data being written to main memory.

The transaction command information from each of the major ports is coupled to the state-machine controller. The state-machine controller is responsible for the dynamic behavior of the memory controller. It processes the transaction commands received from each of the major ports, it implements all necessary state changes to generate all required control signals and execute various types of requested DRAM accesses. The state-machine controller also performs "housekeeping" tasks, necessary for proper operation, but largely independent of the transaction commands.

processing of bus transaction commands

State-Machine Control of FPM and EDO DRAM

The portion of the state-machine controller responsible for the control of FPM and EDO DRAM is generally implemented as a multiplicity of interacting state machines, each tracking a particular aspect of operation. For example, each bank may have a dedicated state machine from which the bank's RAS! signal is derived. Likewise, there may be a similarly dedicated state machine from which the CAS! signals are derived. These and other state machines control all aspects of the activation and timing of the internal controls for row and column address multiplexing; row and column address strobes; output enables for address and data; read/write controls;

determination of timing diagram behavior and data I/O buffering and routing. The state machines are the underlying physical mechanism that determines the timing diagram behavior of all asynchronous DRAM signals, including the interdependence between signals, and the cycle-by-cycle timing for each signal.

Design Note

Memory Bank Selection for Asynchronous DRAM in the AMD-640

When the AMD-640 is controlling FPM and EDO DRAM, 4-byte-wide DRAM banks 0, 2, and 4 are driven by a first set of 4 CAS! lines. The 4-byte-wide banks 1, 3, and 5 are driven by a second set of 4 CAS! lines. Bank-pairs configured for 8-byte-wide simultaneous accesses effectively are driven by 8 CAS! lines. These lines double as column address strobes and byte enables (selects). Only those bytes specifically requested will have their CAS! lines brought active. The memory controller will process the starting address and access width of a request into one or more DRAM access cycles, each generating a bank select, row and column addresses, and one or more contiguous byte enables (CAS! lines). Accesses straddling 4-byte boundaries for 4-byte-wide banks, and 8-byte boundaries for 8-byte-wide bank-pairs, result in two separate cycles, with the most significant byte enables active in the first cycle and the least significant byte enables active in the second cycle.

The timing of the address, strobe, and data sequencing is a function of the type of DRAM memory being accessed in the selected bank, the address of the access to be performed, the immediate past history of accesses in the selected bank (including the address of the last access and the current state of RAS! and CAS!), the type of memory access being performed (read, write, read-modify-write, burst), and the possible need for refresh.

Report on CD-ROM

REPORT

To learn more about Asynchronous DRAM Timing, see the IBM Application Note, *Understanding DRAM Performance Specifications*, on the companion CD-ROM.

State-Machine Control of SDRAM

emission of SDRAM commands

Coarse-grain timing and sequencing of the SDRAM core is carried out indirectly via digital binary-encoded commands, in direct contrast to the fine-grain analog-esque control used for FPM and EDO DRAM. A memory controller for SDRAM emits the commands for sampling on each rising clock edge of the SDRAMs' clocked command interface. The SDRAM command interface maps each encoded command into the desired low-level operation and accordingly executes the fine-grain timing and sequencing of the core's control signals.

The memory controller for SDRAM generates the encoded command primarily using the CAS!, RAS! and WE! lines. These lines are commonly coupled to the command decoder of all banks, with each bank command decoder also receiving a new bank-specific CS! (chip-select) line. The four signals are captured and evaluated by the command decoder within each SDRAM. Only a select one of the bank-specific CS! lines is active, such that only one bank's SDRAM command decoder is enabled. In those banks where the CS! line is deasserted, previous commands continue execution, and the encoded command inputs are ignored.

Aspects of the interface between the memory controller and SDRAM are reminiscent of asynchronous DRAM operation, but constrained to operate within the confines of a clock period. Addresses are multiplexed as row and column address components over a dedicated address bus, shared by all banks. Data to all banks are sent over a dedicated data bus, also shared by all banks. Information on the addresses and data inputs are latched on the rising clock edge of each cycle. If a command is latched at the same time that defines meaningful processing for the contemporaneous address or data information, that information is processed during the subsequent cycle. On reads, the output data is valid for one cycle, at the end of which the memory controller must latch the data. Each address component is sent in conjunction with a separate command. The row address is transmitted over the address bus simultaneously with a Bank Activate command, which is encoded such that RAS! is asserted, but WE! and CAS! are not. The column address is transmitted over the address bus simultaneously with either a Read or Write command, which is encoded such that CAS! is asserted, RAS! is deasserted, and WE! is either deasserted or asserted, respectively.

The command encodings chosen (and listed earlier in Table 5.1 on page 438, are clearly reminiscent of asynchronous DRAM operation. This provides designers with a level of familiarity, has mnemonic value, and likely results in some logic minimizations in the implementation of the core control signals. Nevertheless, because the clocked command interface isolates the core from the memory controller, it should be understood that the encodings are theoretically arbitrary.

While tracking low-level timing is greatly simplified, and the SDRAM can generally be permitted to idle between commands, the memory controller must still keep track of certain aspects of correct sequencing and certain time-outs. For example, the Bank Activate command must be given prior to either a Read or Write command. Also, a bank is not permitted to remain active without precharge beyond a critical time interval.

The memory controller also generates the DQM7! through DQM0! data mask bits, which drive the SDRAM's I/O buffers coupled to the data bus. The data masks function as per byte enables. On reads these enables select (with a two-clock latency) whether the buffers are in a high impedance mode (deasserted), or are driving the data bus with data (asserted). On writes these enables select (with zero clock latency) whether the input data from the data bus will be ignored (deasserted) or written (asserted). Finally, the memory controller for SDRAM must generate the clock enable CKE, which controls clock generation internal to the SDRAM. CKE may be used to initiate Power Down, Suspend, and Self-Refresh modes, depending on how the SDRAM has been configured. CAS!, RAS!, WE!, CS!, and DQMx! are all sampled on each rising clock edge.

Address Processing

The address processing functions include, in the conceptual order of processing: address relocation, address decoding, and address multiplexing.

address relocation

Address relocation acts to map certain programmable ranges of transaction-supplied addresses to different addresses. The mappings provide legacy compatibility with the PC memory architecture and functionality and performance enhancements related to reserved BIOS areas of the PC memory architecture or System Management Mode. One relocation option permits reclamation of DRAM locations that are otherwise inaccessible, because they coincide with reserved BIOS addresses. This relocation acts to extend the highest memory address available by the extent of memory relocated. A second relocation option instead makes the DRAM coinciding with the reserved BIOS addresses to be used for a private System Management address space. Shadowing is a third relocation option that permits BIOS code to be executed from the main memory DRAM, rather than slower BIOS ROM.

address decoding

Once address relocation has been performed, *address decoding* identifies the destination DRAM bank or bank-pair to the state-machine controller. The bank selects are combinational functions of the relocated address and the programmable configuration information, including the densities of the installed DRAMs, the possible use of multi-bank interleave, and the data widths of the installed banks and bank-pairs.

address multiplexing

Each memory cycle, the state-machine controller dynamically configures *address multiplexing* logic in the address processing logic to appropriately parse the memory address into row, column, and byte-select components and sequence the row and column address components over the common DRAM address bus. The routing of address bits to row and column address bits is a combinational function of the relocated address and the programmable configuration information, in particular the row/column address bit-width and the access byte-width of the bank or bank-pair selected by the address decode logic. Figure 5.13 illustrates a typical mapping of physical address bits to row and column address bits in the AMD-640 System Controller. The state-machine controller coordinates the address sequencing over the common DRAM address bus with the address strobes to the appropriate DRAM bank and byte selects and the data sequencing over the common DRAM data bus.

EDO/FP DRAM

Reg 59h Bits 7–5		MA13	MA12	MA11	MA10	MA9	MA8	MA7	MA6	MA5	MA4	MA3	MA2	MA1	MA0	Row:Col
000	Row		23	22	21	11	20	19	18	17	16	15	14	13	12	12:8, 13:8
	Column							10	9	8	7	6	5	4	3	
001	Row		24	23	22	21	20	19	18	17	16	15	14	13	12	10:9, 12:9, 13:9
	Column						11	10	9	8	7	6	5	4	3	
010	Row		25	24	23	21	20	19	18	17	16	15	14	13	12	11:10, 12:10, 13:10
	Column					22	11	10	9	8	7	6	5	4	3	
011	Row		26	25	23	21	20	19	18	17	16	15	14	13	12	12:11, 13:11
	Column				24	22	11	10	9	8	7	6	5	4	3	
100	Row		27	25	23	21	20	19	18	17	16	15	14	13	12	13:12
	Column			26	24	22	11	10	9	8	7	6	5	4	3	

SDRAM

Reg 59h Bits 7–5		MA13	MA12	MA11	MA10	MA9	MA8	MA7	MA6	MA5	MA4	MA3	MA2	MA1	MA0	Row:Col
0xx 16 Mbit	Row			11	11	22	21	20	19	18	17	16	15	14	13	11:10, 11:9, 11:8
	Column			11	PC	24	23	10	9	8	7	6	5	4	3	
1xx 64 Mbit Rev C	Row	12	13	25	22	21	20	19	18	17	16	15	14	24	23	x4 (14:10) x8 (14:9)
	Column	12	13	11	PC	26	11	10	9	8	7	6	5	4	3	
1xx 64 Mbit Rev D	Row	25	12	13	22	21	20	19	18	17	16	15	14	24	23	x4 (14:10) x8 (14:9)
	Column	25	12	13	PC	26	11	10	9	8	7	6	5	4	3	

Figure 5.13 Mapping of Physical Address Bits to Row and Column Address Bits in the AMD-640 System Controller

Copyright 1997, Advanced Micro Devices, Inc., used with permission.

Design Note

Address Signals and Address Control in the AMD-640

In the AMD-640, up to 14 bits of multiplexed address bus signals (MA<13:0>) are shared by all banks of DRAM. DRAM banks 0, 2, and 4 share a first 4-byte-wide data bus, MD<31:0>. Banks 1, 3, and 5 share a second 4-byte-wide data bus, MD<63:32>. Banks configured as bank-pairs effectively share an 8-byte-wide data bus, MD<63:0>. MPD<7:0> is optionally used for parity and ECC error detection and correction, when DRAMs are used that support such features.

The concatenated row and column address is effectively prescaled by 2 bits for 4-byte-wide banks, due to the use of unencoded byte-selects, CAS!<3:0>. The concatenated row and column address is effectively prescaled by 3 bits for 8-byte- wide bank-pairs, due to the use of 8-bit unencoded byte-selects (CAS!<7:0> for FPM and EDO parts) or data masks (DQM!<7:0> for SDRAM parts). The same physical lines are used for CAS!<7:0> and DQM!<7:0>, the line's functionality being defined by the memory controller configuration.

For FPM and EDO parts, RAS!<5:0> is used to activate a specific bank and strobe in the row-address. For SDRAM, the same physical lines are reconfigured and renamed as CS!<5:0>, which serve solely as chip-selects, used to enable the SDRAM on-chip interface command decoders. MWE is a write-enable signal that is shared by all of the FPM and EDO DRAM.

For SDRAM, the AMD-640 provides a clock CLK and clock-enable CKE. The clock is synchronous to and running at the same rate as the CPU clock. The AMD-640 also provides additional signals SRASx!, SCASx!, and SWEx!, corresponding to the generic RAS!, CAS!, and WE! terminology used in the main text. (The x symbol represents one of three identical logic function drive signals, designated by A, B, and C, which allow parallel drive of up to three load trees.) These signals make up the encoded bit-field command that defines the SDRAM's next operation.

Programmable Configuration

A PC platform memory controller permits the system designer a wide scope of customization with respect to part types, organization, and modes of use. This flexibility in behavior is generally accomplished by using per-bank and global configuration parameters to alter the timing and sequencing of the state machines in the state-machine controller. These configuration parameters are held in the CRs detailed earlier in

Figure 5.10 on page 447. Programmable registers may be included to cover the configuration of multiple DRAM types, densities, low-level timing, error detection (including parity or ECC), drive strengths, bank access-width, and refresh.

DESIGN NOTE

System BIOS Memory Detection using the AMD-640

During platform power up and initialization, the memory detection function of the system BIOS evaluates each bank for type and size, starting with bank 0. During bank evaluation the memory controller's "last bank populated" parameter is set to the number of the bank under test. The test addresses generated are kept greater than the known size of previously evaluated banks, if any, while the last bank populated configuration ensures that no bank number greater than the one under test will be enabled. Once the type and size of a bank is determined, the last-bank-populated parameter is incremented and the type and size of the next bank is evaluated.

The type of memory is tested by a write, read, compare sequence using modified EDO timing. If the compare is correct, the bank is EDO. If the test fails, FPM timing is used, and the test repeated. If the compare is correct, the bank is FPM. If the test fails, SDRAM timing is used, and the test repeated to confirm correct operation. All tests are performed across eight consecutive bytes to confirm that both banks in a bank-pair are identical. At the end of a successful compare the memory detection routine sets the bank type in the configuration registers and then proceeds to test the size.

The size of memory is tested by starting with the maximum valid memory column address size and testing for a successful compare at all possible memory densities, starting with the largest. Upon finding a successful compare, the memory detection routine sets the bank size in the configuration registers.

Power-On-Self-Test (POST)

The platform's system BIOS executes *Power-On-Self-Test (POST)* routines from its ROM early during system initialization. These routines will detect the type and size of DRAM installed in the memory banks attached to the memory controller and program the DRAM type and size configuration registers accordingly. The remaining configuration registers are restored from some form of nonvolatile storage, generally the CMOS Memory. The use the CMOS memory has supplanted the earlier use of switches and jumpers on the platform's motherboard to configure bank parameters. If the values in CMOS memory are not present, the POST routines will load the remaining configuration registers with default values from ROM.

DRAM Type and Size

The particular type of FPM, EDO, or SDRAM for each bank or bank-pair must be identified, so that the state-machine controller and DRAM interface can be configured appropriately. Multiple DRAM densities are supported by indicating a starting address and size for each bank. Or alternatively, the memory controller may require population of the banks in order, contiguous population of the address space, and the specification of an ending address. This information is used by the address decoders in the address processing logic to identify the appropriate bank to be accessed for each transaction address. Because there may be more than one combination of row and column bit-widths for a given density, this information also must be provided to the memory controller. See Figure 5.13 on page 457.

Design Note

DRAM Type and Size Support in the AMD-640

In the AMD-640, 36-bit or 32-bit FPM and EDO DRAM in 72-pin SIMMs are supported in 1-, 2-, 4-, and 16-Mbit densities. 72-bit or 64-bit SDRAM in 168-pin DIMMs is supported in 16- and 64-Mbit densities. (The greater bit widths in each type are required for the use of parity or ECC.) The type of DRAM is configured in CR60h, by bits <5:4>, <3:2>, and <1:0>, for bank-pairs 4/5, 2/3, and 0/1, respectively. (Banks within bank-pairs must be identical.)

Banks must be populated in order and contiguously populate the address space. Each bank's ending address must be specified to the memory controller. CR5A-5Fh are used for banks 0-5, respectively. The column-address size (bit-width) is specified in CR58h<7:5>, CR58h<3:1>, and CR59h<7:5>, for bank pairs 0/1, 2/3, and 4/5, respectively.

Reports on CD-ROM

REPORT

To learn more about Programmable Configuration, see the following reports on the companion CD-ROM:

- *Application Note AMD K86 Family BIOS Design*, Document 21329, June 1997.
- *AMD K86 Family BIOS and Software Tools Developers Guide*.

Chapter Summary

This chapter concluded our examination of component-level technologies that play important roles in PC platform architecture. We described how increased processor and I/O performance is forcing a transition from the asynchronous DRAMs common in the past to synchronous DRAMs. PM, FPM, EDO, and SDRAM operating principles were discussed. The emerging Rambus and SLDRAM memory types were briefly introduced. Finally, memory controller requirements for FPM, EDO, and SDRAM were overviewed. Chapter 6 looks at techniques to optimize the platform and the directions in which the platform is heading.