BY VICTOR ECHEVARRIA • RAMBUS

# Choosing and using microprocessor memory interfaces

DESIGNERS EVALUATING MEMORY-INTERFACE OPTIONS NEED TO OPTIMIZE THE TRADE-OFFS AMONG CAPACITY, BANDWIDTH, EFFICIENCY, AND SYSTEM CONSTRAINTS.

Microprocessor-based systems are ideal for executing an essentially infinite number of tasks. The host microprocessors support a limited set of instructions that can combine to produce incredibly complex software programs. In other words, a microprocessor is a piece of hardware designed to be as general-purpose as is feasible, to target as many applications as possible. Moore's Law states that microprocessors will double in complexity roughly every two years. This remarkably accurate prediction has resulted in the latest multicore processors that enable the convergence of cutting-edge technologies.

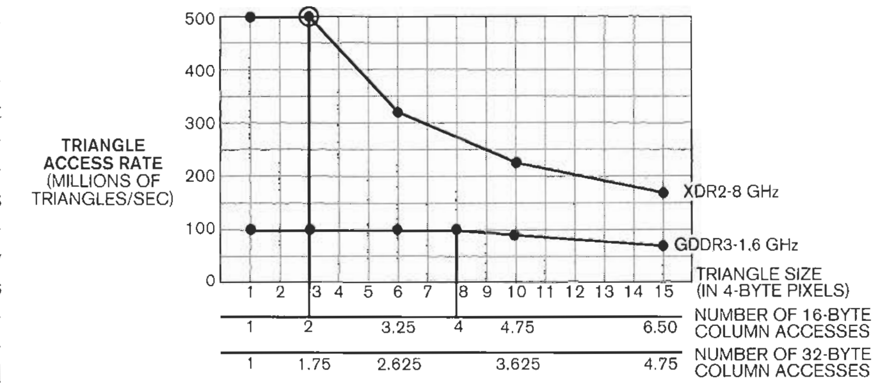

Figure 1 The effective triangle transfer rate compares GDDR3 at 1.6 GHz and XDR2 at 8 GHz.

However, a microprocessor does not make up a complete system. In reality, the supporting components in a system are just as important as the microprocessor itself when determining overall capabilities and performance. Just as microprocessors evolve with time into faster and more efficient devices, supporting components are also evolving to include more complex functions and higher performance interfaces.

PCIe (Peripheral Component Interconnect Express) is quickly becoming the peripheral interconnect of choice, because the slower PCI and AGP (Accelerated Graphics Port) buses place bottlenecks on system performance. For similar reasons, DDR2 (Double Data Rate 2) is slowly taking hold as a general-purpose memory to overcome its slower predecessor, DDR. A system's memory interface can affect performance more than any other system-level interface, and no interface offers more choices and configurations.

At the system level, PCIe interfaces offer configurable options in the form of data rates and lane widths (one, two, four, and, in some cases, eight lanes). In contrast, DDR2 interfaces frequently have widths of 4 to 256 bits and offer a multitude of capacity, data-rate, and core-timing-performance permutations. Add the variety of available memory technologies, and system designers end up with the daunting task of finding an optimal configuration for their systems.

DDR, DDR2, RDRAM (Rambus dynamic-random-access memory), GDDR1/2/3 (graphics DDR1/2/3), and XDR (extreme-data-rate) DRAM are all examples of memory technologies that designers frequently use, and each one has its own set of advantages and drawbacks. To add to this already populous technology space, some memory companies have decided to produce specialty memories that offer superior performance for specially targeted markets. RLDRAM (reduced-latency DRAM) and FCRAM (fast-cycle RAM), for example, are two technologies specifically optimized for network-processor manufacturers that require fast internal DRAM cycle times. XDR2 is a new memory technology from Rambus that incorporates microthreading and offers high efficiency for graphics, networking, and consumer-electronics applications.

Depending on system needs, designers must choose a memory technology and configuration that minimizes the overall system cost and maximizes performance. Typically, designers optimize a memory system for any combination of cost versus capacity, peak bandwidth, efficiency, and system-level restrictions. Although capacity seems straightforward enough, when adding more devices to get more memory, designers must carefully consider how to add the memory devices into the system. For example, adding capacity involves a system-level trade-off, because the added memory devices draw more power.

Enterprise servers and supercomputers are often optimized, at least in part, to contain high-capacity-memory systems. The large and complex programs and data sets that servers and supercomputers often access benefit from more capacity. Users access instructions and data stored in rapid-access areas, such as DRAM subsystems, more quickly than those stored in low-speed, high-latency storage areas, such as hard-disk drives. Therefore, in addition to capacity, these systems are also clearly sensitive to peak bandwidth.

Two ways exist to increase the peak bandwidth of a memory system: increased bus width and increased data rate. The latter involves increasing the rate at which data transfers on each data link, and the former involves increasing the number of data links in the memory system to obtain a higher total aggregate bandwidth. For example, to obtain a total aggregate bandwidth of 12.8 Gbytes/sec, a designer could opt for a 128-bit-wide DDR2 system running at an 800-MHz data rate or a 16-bit-wide XDR system running at a 6.4-GHz data rate.

Most of today's memory technologies provide the capability of achieving high total aggregate data rates, but different applications stress the memory system in different ways. The architectural features of any given memory technology dictate its efficiency for a particular application and its corresponding memory-access requirements. The efficiency of a memory subsystem is defined as the percentage of a system's total aggregate bandwidth that provides useful data to and from the host microprocessor and is the reason memory vendors have added specialty memories to their product portfolios.

ROUTING AREA, CROSSTALK SENSITIVITY, EMI, AND POWER DISTRIBUTION ARE ALL EXAMPLES OF SYSTEM-LEVEL RESTRICTIONS THAT DESIGNERS MUST CONSIDER WHEN SELECTING MEMORY TECHNOLOGY.

RLDRAM, for example, does not have higher capacity or peak-bandwidth specifications than many of the mainstream memory technologies but instead has architectural features that increase its efficiency in certain applications, such as networking. System-level restrictions encompass all of the physical limitations involved with implementing a particular configuration of a given memory technology. Routing area, crosstalk sensitivity, EMI, and power distribution are all examples of system-level restrictions that designers must consider when selecting memory technology.

## OPTIMIZING FOR CAPACITY

Although servers and supercomputers clearly benefit from high-capacity-memory systems, designers of these and other products must determine an optimal means to obtain added capacity. Adding devices to memory subsystems such as those in servers or graphics cards is conceptually straightforward regardless of the memory technology. DDR, DDR2, and GDDR devices are capable of multidrop topologies with certain limitations.

Multidrop topologies in a memory system are those in which each link of the data bus connects to more than one DRAM device. For DDR2 systems, you can connect as many as four devices on each data link. Because GDDR-family devices usually have higher peak data rates, signal-integrity issues typically prevent more than two connections per link, and even then only if the devices reside in close proximity, such as back to back on opposite sides of a pc board.

XDR offers a slightly different approach to scaling capacities. Although the address and command bus is a multidrop configuration, it can connect to 36 devices in sequence on a channel. One reason the XDR address channel supports more devices than DDR links is that DDR multidrop connections are usually stub topologies instead of sequential connections. Stubs generate reflections, which degrade signal quality; with sequential connections, you can electrically compensate for the added capacitive loading of each device along the channel, thereby minimizing impedance discontinuities and their resulting reflections.

Each data link in an XDR system, however, is routed point to point; that is, each data link connects to only one port on the DRAM and one port on the host controller. However, XDR DRAM devices are programmable in width; for example, you can program a ×16 DRAM to act like a ×8, ×4, or ×2 device. Low-capacity systems program each DRAM wider, with more links connecting to each device. Adding capacity merely involves programming the devices to be narrower and connecting fewer data links to each device.

A 32-bit XDR interface can support as little as 64 Mbytes and as much as 1 Gbyte of memory. For systems that require high capacities, an emerging technology, FBDIMM (fully buffered DIMM), introduces a buffer IC onto a module packed with many memory devices. The buffer-IC module connects to the host controller through a serial link, thereby saving controller pins and board-routing area. The buffer chip also connects on the module to each DRAM device and acts as an intermediary between host requests and DRAM responses.

Theoretically, a 32-bit interface could connect to 32 multigigabyte modules.

Each of the above technologies provides its own advantages and drawbacks. Engineers widely use the multidrop topology of a DDR system, but it has limited bandwidth scalability due to complex timing and signal-integrity constraints. XDR is scalable and provides an appropriate cost/capability trade-off but is just now entering production volumes. FBDIMM is an architecture not specifically tied to a particular memory technology, and it provides a high-capacity capability. However, the capacity comes with added latency and a significant price tag, especially at this early stage of the technology's life cycle.

## OPTIMIZING FOR BANDWIDTH

The need for peak bandwidth drives the memory configuration for many applications. Graphics processors, for example, require constant buffering of rendering and frame data to memory. Games, CAD, and digital-content-creation applications all drive the need for higher peak bandwidth due to complex workloads. 3-D-graphics applications require additional bandwidth to render more realistic scene environments with more polygons, richer textures, and additional postprocessing enhancements, such as antialiasing. The latest high-performance graphics cards use 256-bit GDDR3 memory interfaces with 1-GHz data rates for a total aggregate bandwidth of 32 Gbytes/sec.

However, peak bandwidth is not the only parameter to consider when optimizing for bandwidth. Remember that efficiency refers to the percentage of a memory system's total aggregate bandwidth that a controller can actually use. Fewer banks and a higher $t_{RC}$ (row-cycle time) in a DRAM device yield more frequent bank conflicts. Bank conflicts drastically reduce the efficiency of a memory system by forcing potentially long periods of inactivity on the data bus. Write-to-read and read-to-write turnarounds also require long periods of inactivity on the data bus.

MORE AT EDN.COM

Go to www.edn.com/ms4118 and click on Feedback Loop to post a comment on this article.

Memory systems can experience reduced efficiency even with the data bus active 100% of the time. To keep internal pipelines full, DRAM devices implement a feature called prefetch, which allows the DRAM core to run slower than the DRAM interface. The prefetch of a DRAM technology essentially determines how much data transfers for any given transaction, commonly referred to as access granularity.

In the above example, GDDR3 implements a prefetch of four, and a single transaction would therefore yield 32 bytes of data in a configuration that allows for fine access granularity. Graphics processors work largely with units called triangles, and, as graphics-processor generations mature, each triangle decreases in bytes, because smaller triangles yield more realistic rendered images. A transfer of 32 bytes may be more than necessary to access the triangles needed for a process. For example, if only one 4-byte triangle were necessary from memory, 28 bytes of the corresponding access would go to waste. Even though the bus is active during the entire transfer, the efficiency of the transfer significantly reduces. New memory technologies, such as XDR2, are emerging to further enhance bandwidth efficiency for various applications. **Figure 1** shows the effective triangle transfer rate versus triangle size for GDDR3 at 1.6 GHz and XDR2 at 8 GHz. Designers optimizing their memory system for bandwidth must consider both peak bandwidth and efficiency to get the best performance out of their host processor.

MICROPROCESSORS TARGETING SPECIFIC APPLICATIONS MAY GAIN PERFORMANCE FROM THEIR MEMORY SYSTEMS BY INTEGRATING THE MEMORY INTERFACE DIRECTLY ONTO THE PROCESSOR ITSELF.

## OPTIMIZING FOR SYSTEM CONSTRAINTS

Architectural factors, such as capacity and bandwidth, don't limit many designers; however, these designers must still consider total system cost and physical limitations when configuring the memory system. DRAM devices often make up the most expensive portion of a system's BOM (bill of materials), and designers should make every effort to use the fewest devices possible to meet the architectural goals. Expanding on the example above, one XDR device or eight DDR2 devices could satisfy a microprocessor that demands 64 Mbytes of memory with 12.8 Gbytes/sec of total aggregate bandwidth. If the same microprocessor were to demand the same bandwidth but with 1024 Mbytes of memory, eight 1-Gbit DDR2 devices could still satisfy the requirement; however, achieving the same capacity with XDR would require 16 512-Mbit devices until 1-Gbit devices enter mass production.

If the product were particularly sensitive to total system cost, then an option enabling the lowest memory-system BOM cost for the particular system requirement would be ideal. In addition, as data-link speeds increase, signal-integrity concerns, such as EMI, crosstalk, reflections, and power-supply noise, become much more important. Aside from the usual system optimizations, such as stub-length reduction, trace shielding, and supply bypassing, the memory configuration itself can play a role in easing signal-integrity issues. Single-ended traces typically require less room to route on a pc board, but differential data links provide more immunity to crosstalk and EMI radiation.

Low-voltage swings reduce the stress on I/O power supplies as do differential drivers, which drastically reduce di/dt noise. Slower memory technologies are generally easier to route on a pc board, given the eased timing-skew restrictions. Designers must take care with high-speed-memory designs to ensure minimal flight-time skew between data links. Strobe-based systems, for example, introduce complex timing constraints at both the silicon and pc-board levels. New technologies, such as FlexPhase, dynamically calibrate out skew in the datapath but are not integrated into all available memory-interface cells on the market. DDR2, GDDR, and XDR all use on-die termination that minimizes stub reflections, but multidrop DDR2 and GDDR topologies still have stubs between devices. In addition to architectural parameters, designers should consider system constraints when choosing a memory configuration.

Microprocessors are generic devices that you may use for specific applications. Decoupling the memory interface from the microprocessor keeps the processor more generic in scope and enables multiple designs to use different memory configurations with the same processor. On the other hand, microprocessors targeting specific applications may gain performance from their memory systems by integrating the memory interface directly onto the processor itself.

Regardless of the application, designers have many choices when specifying the memory subsystem and should take care to optimize the trade-offs among capacity, bandwidth, efficiency, and system constraints. Most of today's memory technologies ensure that virtually any application has a sufficient if not optimal memory-system technology. Whether it's XDR for high-performance and low-cost optimization or DDR2 for high-capacity commodity systems, system designers and chip architects have a full menu of options that can save money and boost performance. EDN

## AUTHOR'S BIOGRAPHY

*Victor Echevarria is a product-marketing manager in the Platform Solution Group at Rambus Inc. He currently manages product-marketing activity for the RDRAM and XDR product lines for new and existing customers. Echevarria joined Rambus in 2002 as a systems engineer. Before joining Rambus, he worked with Agilent Technologies, where he developed software for its high-speed digital-sampling oscilloscopes. Echevarria graduated from the University of California—Berkeley, in 2002 with a BS in electrical engineering and computer science.*