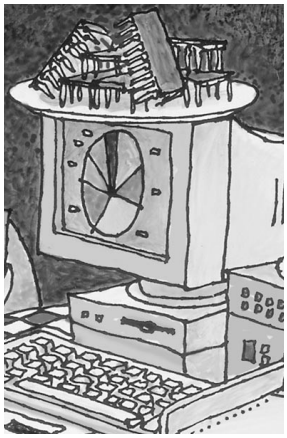
SLDRAM: HIGH-PERFORMANCE, OPEN-STANDARD MEMORY

Peter Gillingham

MOSAID Technologies

Bill Vogley

Texas Instruments



Synchronous-link DRAM—the highest rung yet on DRAM's evolutionary ladder—is a high-bandwidth interface standard meeting memory requirements into the 21st century.

The primary objective of DRAM—dynamic random access memory—is to offer the largest memory capacity at the lowest possible cost. Designers achieve this by two means. First, they optimize the process and the design to minimize die area. Second, they ensure that the device serves high-volume markets and can be mass-produced to achieve the greatest economies of scale.

SLDRAM—synchronous-link DRAM—is a new memory interface specification developed through the cooperative efforts of leading semiconductor memory manufacturers and high-end computer architects and system designers. SLDRAM meets the high data bandwidth requirements of emerging processor architectures and retains the low cost of earlier DRAM interface standards. These and other benefits suggest that SLDRAM will become the mainstream commodity memory of the early 21st century.

Benefits

In developing SLDRAM (see Origins box for details), we had several goals beyond high bandwidth and low cost. It was also important that system designers could adopt the new interface smoothly, that it would work for a wide variety of applications, and that its manufacturing yields would be high. Finally, it was important for SLDRAM to be an open standard.

Evolutionary solution. Over more than 20 years encompassing nine generations of DRAMs (from 1 Kbit to 64 Mbits), the market has always embraced evolutionary solutions. With evolutionary changes, system designers can adapt existing technology and migrate to higher performance solutions. As shown in Figure 1 next page, SLDRAM represents the next step in DRAM's evolution, after EDO, SDRAM, and DDR (extended data out, synchronous, and double-data-rate DRAM).

SDRAM includes several important architectural features over standard EDO, including multiple internal banks, a clocked synchronous interface, terminated small-swing signaling, and programmable data bursts. These changes decouple internal DRAM address and control paths from the data interface to achieve higher bandwidth. The emerging standard for DDR adds data clocking on both clock edges and a return clock, allowing higher speed operation with an improved system timing margin.

SLDRAM builds on the features of SDRAM and DDR by adding an address/control packet protocol, in-system timing and signaling optimization, and full compatibility from generation to generation. Command packets in the SLDRAM protocol include spare bits to accommodate addressing for the 4-Gbit generation and beyond.

The first SLDRAM samples will be 64-Mbit devices operating at 400 Mbps/pin. Interfaces with bandwidths of 600 and 800 Mbps/pin, and eventually greater than 1 Gbps/pin, will come on the market when they become cost-effective. The protocol allows us to mix interfaces of different speeds. So, for example, we can plug an 800-Mbps/pin device into a 400-Mbps/pin system, and it will operate correctly at 400 Mbps/pin.

Application variety. The standards working group specified SLDRAM as a general-purpose, high-performance DRAM for a wide variety of applications. As computer main memory—in desktop, mobile, and high-end servers and workstations—SLDRAM offers high sustainable bandwidth, low latency, low power, user upgradability, and support for large hierarchical memory configurations. For video, graphics, and telecommunications applications, SLDRAM provides multiple independent banks, fast read/write bus turnaround, and the capabil-

Origins of SLDRAM

Only recently, the computer industry made the transition from fast-page-mode DRAM to EDO (extended data out) DRAM. Now, it has accepted SDRAM (synchronous DRAM) as the standard for mainstream computer applications.

In the early days of SDRAM, as the standards were being developed, both manufacturers and users were concerned about cost. They did not know how much silicon the new functions would consume and how much additional complexity it would take to test and verify them. With SDRAM now fully standardized, these issues have been resolved, and the technology has achieved volume production, commodity-level pricing.

During the development of SDRAM in 1989 and 1990, the SCI working group (IEEE Std. P1596—Scalable Coherent Interface) was developing a memory interface as a subgroup. RamLink, which became IEEE Std. 1596.4 for memory interfaces, is a point-to-point interface using the SCI protocol but with a reduced command set that makes it memory specific. During RamLink's development, the working group realized that the interface would have to support large memory configurations. The protocol allowed this, and so did the architecture. The drawback was the serial delay from one device to the next in a point-to-point system. This increased latency from 4 to 6 ns per device on the interface. If a configuration incorporated 64 devices, the additional latency would be intolerable—around 300 to 400 ns.

It made sense to stick with the RamLink protocol, but we needed a parallel interface. Because the new devices would fundamentally be synchronous DRAMs linked in a different manner, the group chose the name SyncLink. A little over a year ago, we discovered that Synclink had already been registered and used as a trademark. Thus, we instead adopted the name SLDRAM, for synchronous link DRAM. We retained the RamLink protocol but made the interface parallel.

To determine the most efficient method of exchanging address, control, and read and write data between the controller and memory devices, we simulated various bus configurations with actual traces of processor memory requests. In real systems, main memory address requests are virtually random, and the read/write ratio averages about 4:1.

We rejected several configurations, including one with a single, bidirectional bus for address, control, and data. Because of the need to interrupt data flow to issue a command in this configuration, effective data bandwidth was a small fraction of theoretical peak bandwidth. The single bus sacrificed the benefits of hidden bank activation and deactivation.

We also rejected a configuration having one unidirectional bus from controller to memory for command, address, and write data, and another from memory to controller for read data. Although this was an improvement over the single bus, effective bandwidth still suffered from the collision of commands with write data.

The configuration we chose consists of one unidirectional bus for command and address and a bidirectional bus for read and write data. In this optimal configuration, the data bus bandwidth could be fully utilized. Placement of the command and address in a packet reduces pin count, allows future address expansion without pin increments, and enables higher pipelining capability.

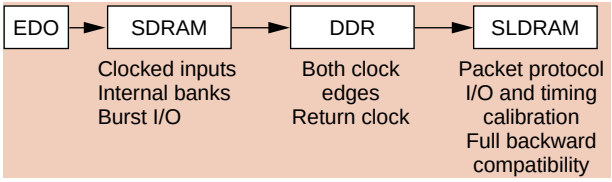


Figure 1. Evolution from EDO to SLDRAM, by way of SDRAM and DDR.

ity for small, fully pipelined bursts. SLDRAM addresses the requirements of all major high-volume DRAM applications.

Cost-effectiveness. As was the case with SDRAM, industry will adopt a new, high-performance DRAM interface once its cost premium over the conventional alternative falls below 5% (see Figure 2, next page). Designers achieve high performance by improving the interface while leaving the DRAM core relatively unchanged. We could easily have improved core performance with reduced bit-line, word-line, and data-bus loading. However, this would have generated a significant die area penalty because of increased array fragmentation, grossly exceeding the 5% cost target.

SDRAM, DDR, and SLDRAM all use a DRAM core that has a page-mode cycle time of roughly 10 ns. To maintain an efficient die layout and obtain an interface rate higher than the core cycle time, the device must fetch several words in parallel. For example, a 16-bit I/O, 200-Mbps/pin, DDR device must read or write two I/O words over a 32-bit internal data path in one 10-ns core cycle.

The need to widen the internal data path also leads to a die cost penalty. For the 16-Mbit generation of SDRAM (100-Mbps/pin, 16-bit I/O), this penalty has recently fallen below 5%. DDR running at 200 Mbps/pin will become cost-effective for 64-Mbit devices, which have a sufficient number of active memory subarrays to support a 32-bit data path without substantial area penalty. SLDRAM devices will first become available with a 400-Mbps/pin, 16-bit I/O interface employing a 64-bit internal data path. These devices will be cost-effective in the 256-Mbit density. Later SLDRAM devices will run at 800 Mbps/pin and employ 128-bit internal data paths. SLDRAM or any other memory technology using a 128-bit internal data path will not be cost-effective until the 1-Gbit generation.

Manufacturability. DRAM's low cost is due to high manufacturing yields, but the tight timing specifications required for high-frequency operation are incompatible with high yield. Therefore, we defined SLDRAM so that virtually any functional part will meet the specification in one speed grade

or another. On power-up, a system can ascertain the speed performance capabilities of all SLDRAMs present and then make the appropriate adjustments. The SLDRAM packet protocol permits the system to adjust and match setup and hold time, data delay, and output drive levels of individual memory devices for consistent system operation. Periodically during operation, the system recalibrates the devices to account for system drift. (We will discuss this in more detail later in the article.) The flexibility afforded by the SLDRAM packet protocol ensures high yield and low cost.

Low system cost. SLDRAM achieves low system cost through conventional packaging and printed-circuit-board technology. The SLDRAM devices themselves are packaged in standard, 0.5-mm pitch TSOPs (thin, small-outline packages) or in 0.8-mm, staggered-pitch vertically mounted packages (VSMPs). With buffered modules, an SLDRAM controller must support only 33 high-speed signals to accommodate gigabyte memory configurations. Wide modules can add 16-bit data channels without additional control overhead for increased memory bandwidth. For the SLDRAM interface, we recommend conventional low-cost PCB material with 5-mm tracks, using two of four layers for interconnect.

Open standard. SLDRAM is an open standard that will be formalized by IEEE (Std P1596.7) and JEDEC specifications. Open standards allow manufacturers to develop differentiated products that address emerging applications and niche´ opportunities. Open competition will ensure rapid development of DRAM technology at the lowest possible cost.

Architectural and functional overview

In this section we describe how SLDRAM works in a system, covering the basics of SLDRAM protocol, timing, and signals. For a full description of SLDRAM functionality, we encourage readers to refer to the SLDRAM data sheet, which is available on the SLDRAM Consortium Web page (www.sldram.com).

Bus topology. The SLDRAM multidrop bus has one memory controller and up to eight loads. A load can be either a single SLDRAM device or a buffered module with many SLDRAM devices. Command, address, and control information from the memory controller flows to the SLDRAMs on the unidirectional CommandLink. Read and write data flow between controller and SLDRAM on the bidirectional DataLink. Both CommandLink and DataLink operate at the same rate (400-Mbps/pin, 600-Mbps/pin, 800-Mbps/pin, and so on). Figure 3 illustrates SLDRAM signals and data flow. In actual SLDRAM modules, all connections are on one side.

Signal names and definitions. The CommandLink comprises signals CCLK, CCLK*, FLAG, CA[9:0], LISTEN, LINKON, and RESET. (See Table 1 for descriptions.) Commands consist of four consecutive 10-bit words on CA[9:0]. A 1 on the FLAG bit indicates the first word of a command. The SLDRAMs use both edges of the differential free-running clock (CCLK/CCLK*) to latch command words. For a 400-Mbps/pin SLDRAM, the clock frequency is 200 MHz, and bit period N is equal to 2.5 ns—half the clock period. While the LISTEN pin is high, the SLDRAMs monitor the CommandLink for commands. When LISTEN is low, there can be no commands on the CommandLink, so the SLDRAMs enter a power-saving standby mode. When LINKON is low, the SLDRAMs enter a shutdown mode, in which CCLK can be turned off to achieve zero power on the link. A RESET signal puts the SLDRAMs in a known state on power-up.

The DataLink is a bidirectional bus for the transmission of write data from controller to the SLDRAMs and read data from the SLDRAMs back to the controller. It consists of DQ[17:0], DCLK0, DCLK0*, DCLK1, and DCLK1*. Read and write data packets of a minimum burst length of four are

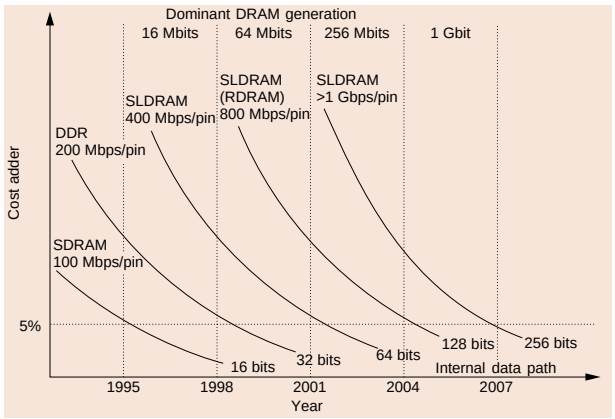


Figure 2. DRAM bandwidth cost penalty for 16-bit I/O.

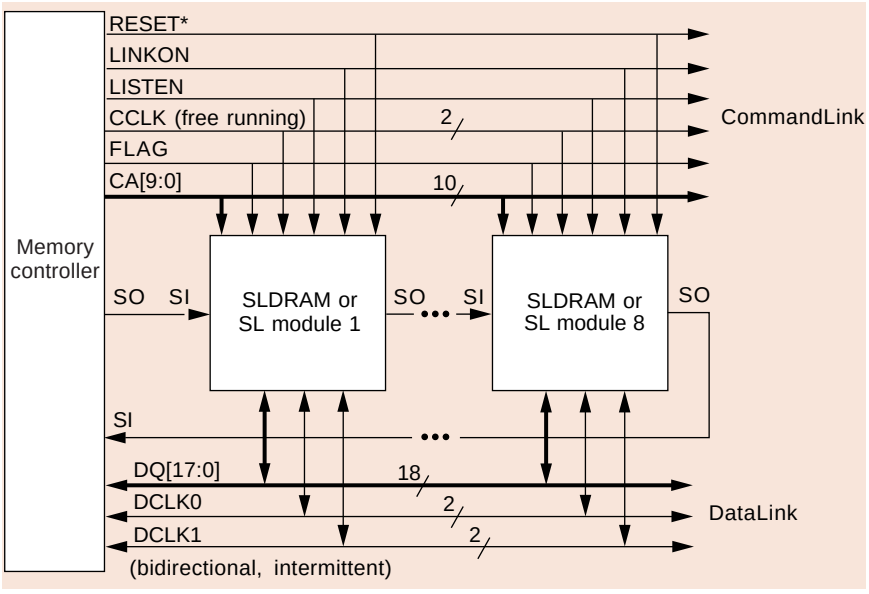


Figure 3. SLDRAM bus topology.

Table 1. SLDRAM signal names and descriptions. MC: memory controller; SSTL_2: series stub terminated logic for 2.x-V generation.

Bus	Signal	Description	Direction	Level
CommandLink	CCLK/CCLK*	Command clock	MC → SLDRAM	SLIO
	CA[9:0]	Command address bus	MC → SLDRAM	SLIO
	LISTEN	Standby mode	MC → SLDRAM	SLIO
	LINKON	Shutdown mode	MC → SLDRAM	LVTTL
	RESET	Hard reset	MC → SLDRAM	LVTTL
DataLink	DCLK0/DCLK0*	Data clock 0	MC fl↔ SLDRAM	SLIO
	DCLK1/DCLK1*	Data clock 1	MC fl↔ SLDRAM	SLIO
	DQ[17:0]	Data bus	MC fl↔ SLDRAM	SLIO
Serial	SI	Serial input	MC → SLDRAM, ´ SLDRAM → SLDRAM	LVTTL
	SO	Serial output	SLDRAM → SLDRAM, SLDRAM → MC	LVTTL

accompanied by either differential clock—DCLK0/DCLK0* or DCLK1/DCLK1*. The two sets of clocks allow control of the DataLink to pass from one device to another with the minimum gap. On power-up, a daisy-chained serial bus with input SI and output SO on each device synchronizes the SLDRAMs and assigns unique IDs to each one.

Pipelined transactions. The timing diagram in Figure 4 shows a series of page read and page write commands issued by the memory controller to the SLDRAMs. For purposes of illustration, all burst lengths are 4*N*, although the controller can dynamically mix 4*N* and 8*N* bursts. The read access time to an open bank, also known as page read latency, is shown here as 12*N* (30 ns). The first two commands are page reads to SLDRAM 0 to either the same or differnt banks. SLDRAM 0 drives the read data on the data bus along with DCLK0 to provide the memory controller the necessary edges to strobe in read data. Since the first two page read commands are for the same SLDRAM, it is not necessary to insert a gap between the two 4*N* data bursts. The SLDRAM itself ensures that DCLK0 is driven continuously without glitches.

The data burst for the next page read—to SLDRAM 1—must be separated by a 2*N* gap. This allows for settling of the DataLink bus and for timing uncertainty between SLDRAM 0 and SLDRAM 1. A 2*N* gap is necessary any time control of the DataLink passes from one device to another. This occurs in reads to different SLDRAMs and in read-to-write and write-to-read transitions between SLDRAMs and the memory controller. The memory controller creates the gap between data by inserting a 2*N* gap between commands. SLDRAM 1 begins driving the DCLK lines well in advance of the actual data burst.

The next command is a write command in which the controller drives DCLK0 to strobe write data into SLDRAM 2. The page write latency of the SLDRAM is programmed to equal page read latency minus 2*N*. To create a 2*N* gap between Read1 and Write2 data on the DataLink, the Write2 command must be delayed 4*N* after the Read1 command. Programming write latency in this manner creates an open 4*N* command slot on the CommandLink, which could be used for nondata commands such as row open or close, register write, or refresh, without affecting DataLink utilization. The subsequent read command to SLDRAM 3 does not require any additional delay to achieve the 2*N* gap on the DataLink. The final burst of three consecutive write commands shows that the 2*N* gap between data bursts is not necessary when the system is writing to different SLDRAM devices. This is because all write data originates from the memory controller.

Data clocks. When control of the DataLink passes from one device to another, the bus remains at a midpoint level for nominally 2*N*. This results in indeterminate data and possibly multiple transitions at the input buffers. This is acceptable for the data lines themselves, but not the data clocks, which strobe data. To solve this problem, the data clocks

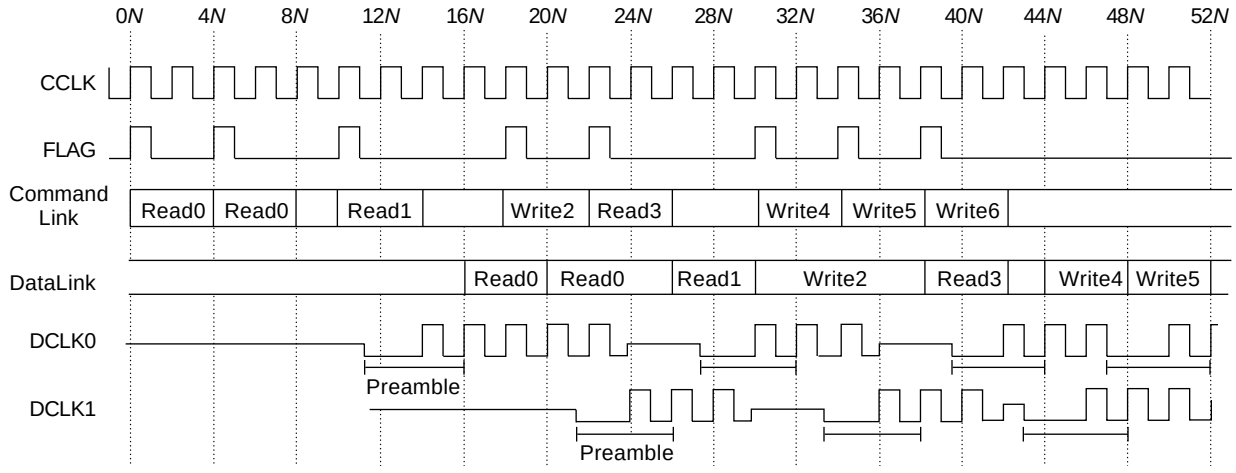


Figure 4. Timing diagram, SLDRAM bus transactions.

have a 00010 preamble before the transition associated with the first bit of data occurs. The device receiving data can enable the DCLK input buffer at any time during the first 000 period. The preamble also includes dummy transition 10 to remove pulse-width-dependent skew (also known as intersymbol interference, or ISI) from the DCLK signal. The receiving device ignores the first rising and falling edge of DCLK and begins clocking data on the second rising edge. There are two data clocks, so the device can accommodate gapless 4*N* write bursts to different SLDRAMs and 4*N* read bursts from different SLDRAMs. The controller indicates in each command packet which DCLK is to be used.

The controller transmits CCLK edges coincidentally with edges of CA[9:0] and FLAG data. DCLK edges originating from the controller are also coincident with DQ[17:0] data. The SLDRAMs add fractional delay to incoming CCLK and DCLKs to sample commands and write data at the optimum time. (We discuss this in greater detail in the following section and in the section on synchronization.) The controller can program the SLDRAMs to add fractional delay to outgoing DCLKs during read operations. This allows the controller read data input registers to directly strobe in read data using the received DCLK without internal delay adjustments.

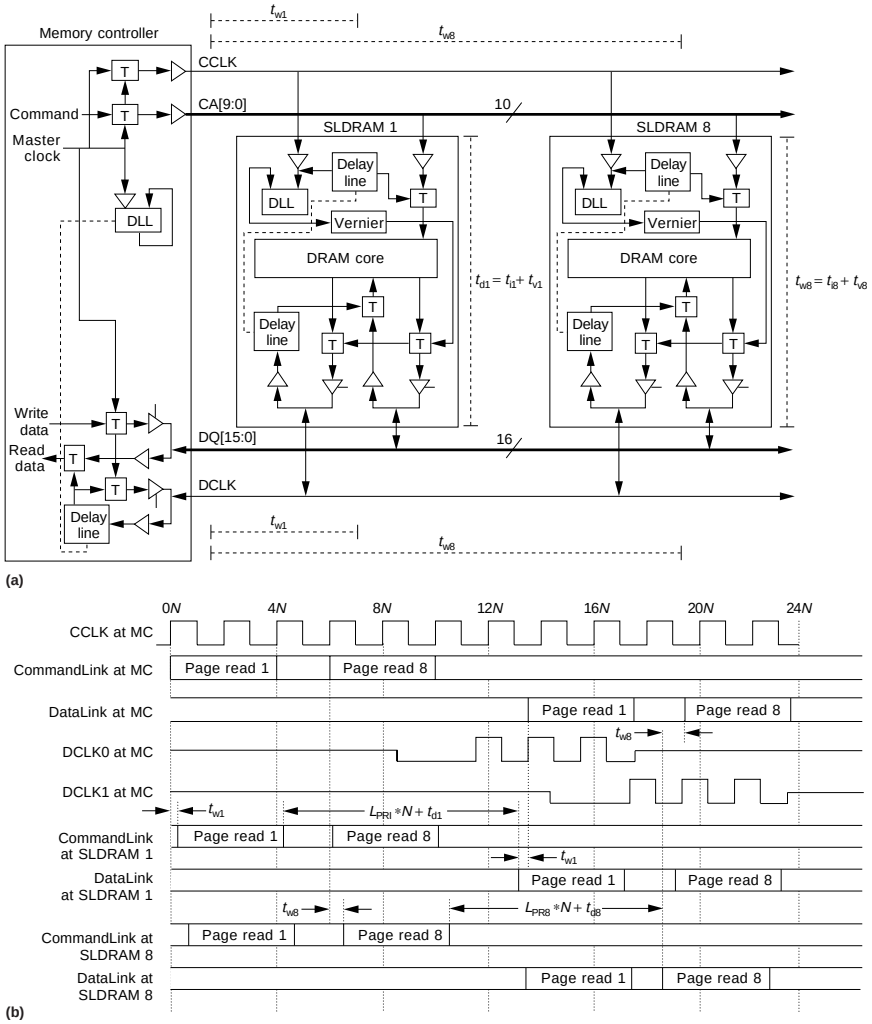


Figure 5. SLDRAM bus timing adjustment system block diagram (a); waveforms (b).

Timing adjustment. The controller programs each SLDRAM with four timing latency parameters: page read, page write, bank read, and bank write. We define latency as the time between the command burst and the start of the associated data burst. For consistent operation of the memory subsystem, each SLDRAM should be programmed with the same values. On power-up, the memory controller polls the status registers in each SLDRAM to determine minimum latencies, which may vary by manufacturer and speed grade. The controller then programs each SLDRAM with the worst-case values.

Read latency is adjustable in coarse increments of unit bit intervals and fine increments of fractional bit intervals. The controller programs the coarse and fine read latency of each SLDRAM so that read data bursts from different devices, at different electrical distances from the controller, all arrive back at the controller with equal delay from the command packet. Write latency is only adjustable in coarse increments. The write latency values determine when the SLDRAM begins looking for transitions on DCLK to strobe in write data. Since this can occur any time during the 000 portion of the DCLK preamble, it does not require fine adjustment.

Figure 5a shows a conceptual block diagram of the SLDRAM system with major timing components. A DLL (delay-locked loop) in both the controller and SLDRAM locks to the free-running clock to provide a stable reference for the input sampling delay lines. Input and output latches acting on both edges of the clock are labeled T.

Figure 5b shows the timing of two consecutive page read commands, the first addressed to the SLDRAM nearest the controller and the second to the SLDRAM at the far end of the bus. Total latency observed by the controller on the first read includes the programmed page read latency (L_{PR1}), the clock-to-data propagation delay through the SLDRAM itself (t_{d1}), and wire delay to the SLDRAM and back ($2t_{w1}$). The clock-to-data propagation delay is composed of intrinsic delay (t_{i1}) and the programmed fine vernier delay (t_{v1}). When an increment vernier command issued by the controller causes the fine read vernier to overflow past 1*N* delay, digital latency L_{PR} is incremented by one and fine vernier setting t_v is reduced to 0. If a decrement vernier command causes the

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.