example also shows the timing of DQM to prevent writing the two middle words. Since DQM follows the CAS* timing, it is also directly in line with write data. DQM is very useful for writes, especially on multibyte SDRAM devices, because it enables the uniform execution of a burst transfer while selectively preventing the unwanted modification of certain memory locations. When working with an SDRAM array composed of byte-wide devices, it would be possible to deassert chip select to those byte lanes that you don't want written. However, there is no such option for multibyte devices other than DQM.

When the transaction completes, the row is left either activated or precharged, depending on the state of AP during the CAS* assertion. If left activated, the controller may immediately issue a new RD or WR command to the same row. Alternatively, the row may be explicitly precharged. If automatically precharged, a new row in that bank may be activated in preparation for other transactions. A new row can be activated immediately in most cases, but attention must be paid to the SDRAM's specifications for minimum times between active to precharge commands and active to active commands.

After configuring an SDRAM for a particular default burst length, it will expect all transactions to be that default length. Under certain circumstances, it may be desirable to perform a shorter transaction. Reads and writes can be terminated early by either issuing a *precharge* command to the bank that is currently being accessed or by issuing a *burst-terminate* command. There are varying restrictions and requirements on exactly how each type of transaction is terminated early. In general, a read or write must be initiated without automatic precharge for it to be terminated early by the memory controller.

The capability of performing back-to-back transactions has been already mentioned. In these situations, the startup latency of a new transaction can be accounted for during the data transfer phase of the previous transaction. An example of such functionality is shown in Fig. 8.5. This timing diagram uses a common SDRAM presentation style in which the individual control signals are replaced by their command equivalent. The control signals are idle during the data portion of the first transaction, allowing a new request to be asserted prior to the completion of that transaction. In this example, the controller asserts a new read command for the row that was previously activated. By asserting this command one cycle (CAS latency minus one) before the end of the current transaction, the controller guarantees that there will be no idle time on the data bus between transactions. If a the second transaction was a write, the assertion of WR would come the cycle after the read transaction ended to enable simultaneous presentation of write data in phase with the command. However, when following a write with a read, the read command cannot be issued until after the write data completes, causing an idle period on the data bus equivalent to the selected CAS latency.

This concept can be extended to the general case of multiple active banks. Just as the controller is able to assert a new RD in Fig. 8.5, it could also assert an ACTV to activate a different bank. Therefore, any of an SDRAM's banks can be asserted independently during the idle command time of an in-progress transaction. When these transactions end, the previously activated banks can be seamlessly read or written in the same manner as shown. This provides a substantial performance boost and can eliminate most overhead other than refresh in an SDRAM interface.

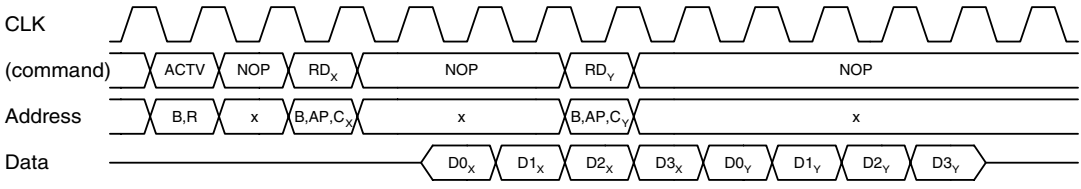

FIGURE 8.5 Back-to-back read transactions (CL = 2, BL = 4).

Periodic refresh is a universal requirement of DRAM technology, and SDRAMs are no exception. An SDRAM device may contain 4,096 rows per bank (or 8,192, depending on its overall size) with the requirement that all rows be refreshed every 64 ms. Therefore, the controller has the responsibility of ensuring that 4,096 (or 8,192) refresh operations are carried out every 64 ms. Refresh commands can be evenly spaced every 15.625 μs (or 7.8125 μs), or the controller might wait until a certain event has passed and then rapidly count out 4,096 (or 8,192) refresh commands. Different SDRAM devices have slightly differing refresh requirements, but the means of executing refresh operations is standardized. The first requirement is that all banks be precharged, because the auto-refresh (REF) command operates on all banks at once. An internal refresh counter keeps track of the next row across each bank to be refreshed when a REF command is executed by asserting RAS* and CAS* together.

It can be easy to forget the asynchronous timing requirements of the DRAM core when designing around an SDRAM's synchronous interface. After a little time spent studying state transition tables and command sets, the idea that an asynchronous element is lurking in the background can become an elusive memory. Always be sure to verify that discrete clock cycle delays conform to the nanosecond timing specifications that are included in the SDRAM data sheet. The tricky part of these timing specifications is that they affect a system differently, depending on the operating frequency. At 25 MHz, a 20-ns time delay is less than one cycle. However, at 100 MHz, that delay stretches to two cycles. Failure to recognize subtle timing differences can cause errors that may manifest themselves as intermittent data corruption problems, which can be very time consuming to track down.

SDRAM remains a mainstream memory technology for PCs and therefore is manufactured in substantial volumes by multiple manufacturers. The SDRAM market is a highly competitive one, with faster and denser products appearing regularly. SDRAMs are commonly available in densities ranging from 64 to 512 Mb in 4, 8, and 16-bit wide data buses. Older 16-Mb parts are becoming harder to find. For special applications, 32-bit wide devices are available, though sometimes at a slight premium as a result of lower overall volumes.

8.2 DOUBLE DATA RATE SDRAM

Conventional SDRAM devices transfer one word on the rising edge of each clock cycle. At any given time, there is an upper limit on the clock speed that is practical to implement for a board-level interface. When this level of performance proves insufficient, *double data rate* (DDR) SDRAM devices can nearly double the available bandwidth by transferring one word on both the rising and falling edges of each clock cycle. In doing so, the interface's clock speed remains constant, but the data bus effectively doubles in frequency. Functionally, DDR and single data rate (SDR) devices are very similar. They share many common control signals, a common command set, and a rising-edge-only control/address interface. They differ not only in the speed of the data bus but also with new DDR data control signals and internal clocking circuitry to enable reliable circuit design with very tight timing margins. Figure 8.6 shows the DDR SDRAM structure.

A DDR SDRAM contains an internal data path that is twice the width of the external data bus. This width difference allows the majority of the internal logic to run at a slower SDR frequency while delivering the desired external bandwidth with half as many data pins as would be required with a conventional SDRAM. Rather than supplying a 2× clock to the SDRAM for its DDR interface, a pair of complementary clocks, CLK and CLK*, are provided that are 180° out of phase with each other. Input and output signals are referenced to the crossings of these two clocks, during which a rising edge is always present in either clock. Commands and addresses are presented to a DDR SDRAM as they would be for an SDR device: on the rising edge of CLK. It is not necessary to dou-