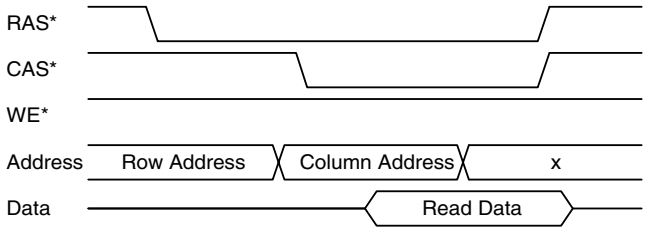


**FIGURE 4.11** Basic DRAM read (CE* = 0, OE* = 0).

of the transaction. Some time later, the read data is made available on the data bus. After waiting for a sufficient time for the DRAM to return the read data, the memory controller removes RAS* and CAS* to terminate the transaction.

Basic writes are similar to single reads as shown in Fig. 4.12. Again, CE* is assumed to be held active, and, being a write, OE* is assumed to be held inactive throughout the transaction.

Like a read, the write transaction begins by loading the row address. From this it is apparent that there is no particular link between loading a row address and performing a read or a write. The identity of the transaction is linked to the falling edge of CAS*, when WE* is asserted at about the same time that the column address and write data are asserted. DRAM chips require a certain setup and hold time for these signals around the falling edge of CAS*. Once the timing requirements are met, address can be deasserted prior to the rising edge of CAS*.

A read/write hybrid transaction, called a *read-modify-write*, is also supported to improve the efficiency of the memory subsystem. In a read-modify-write, the microprocessor fetches a word from memory, performs a quick modification to it, and then writes it back as part of the same original transaction. This is an *atomic* operation, because it functions as an indivisible unit and cannot be interrupted. Figure 4.13 shows the timing for the read-modify-write. Note that CAS* is held for a longer period of time, during which the microprocessor may process the read-data before asserting WE* along with the new data to be written.

Original DRAMs were fairly slow. This was partly because older silicon processes limited the decode time of millions of internal addresses. It was also a result of the fact that accessing a single location required a time-consuming sequence of RAS* followed by CAS*. In comparison, an SRAM is quick and easy: assert the address in one step and grab the data. DRAM went through an architectural evolution that replaced the original devices with *fast-page mode* (FPM) devices that allow more efficient accesses to sequential memory locations. FPM DRAMs provide a substantial increase in usable memory bandwidth for the most common DRAM application: CPU memory. These devices take advantage of the tendency of a microprocessor's memory transactions to be sequential in nature.

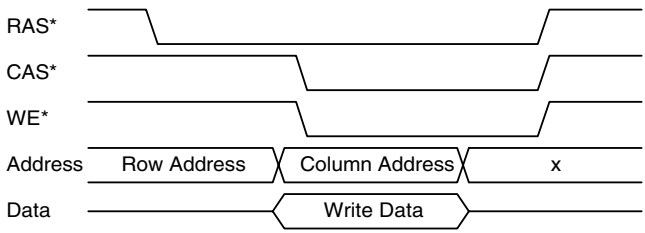


**FIGURE 4.12** Basic DRAM write (CE* = 0, OE* = 1).

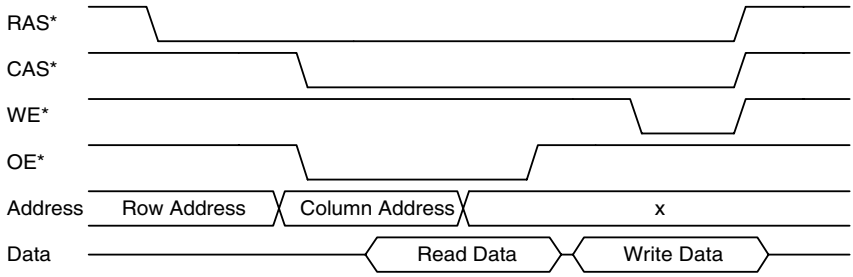


**FIGURE 4.13** Read-modify-write transaction.

Software does occasionally branch back and forth in its memory space. Yet, on the whole, software moves through portions of memory in a linear fashion. FPM devices enable a DRAM controller to load a row-address in the normal manner using RAS* and then perform multiple CAS* transactions using the same row-address. Therefore, DRAMs end their transaction cycles with the rising edge of RAS*, because they cannot be sure if more reads or writes are coming until RAS* rises, indicating that the current row-address can be released.

FPM technology, in turn, gave way to *extended-data out* (EDO) devices that extend the time read data is held valid. Unlike its predecessors, an EDO DRAM does not disable the read data when CAS* rises. Instead, it waits until either the transaction is complete (RAS* rises), OE* is deasserted, or until CAS* begins a new page-mode access. While FPM and EDO DRAMs are distinct types of devices, EDO combines the page-mode features of FPM and consequently became more attractive to use. The following timing discussion uses EDO functionality as the example.

Page-mode transactions hold RAS* active and cycle CAS* multiple times to perform reads and writes as shown in Figs. 4.14 and 4.15. Each successive CAS* falling edge loads a new column address and causes either a read or write to be performed. In the read case, EDO's benefit can be properly observed. Rather than read data being removed when CAS* rises, it remains asserted until just after the next falling edge of CAS* or the rising edge of RAS* that terminates the page-mode transaction.

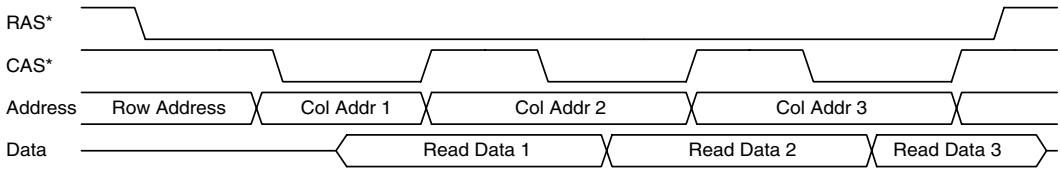


**FIGURE 4.14** Page-mode reads.

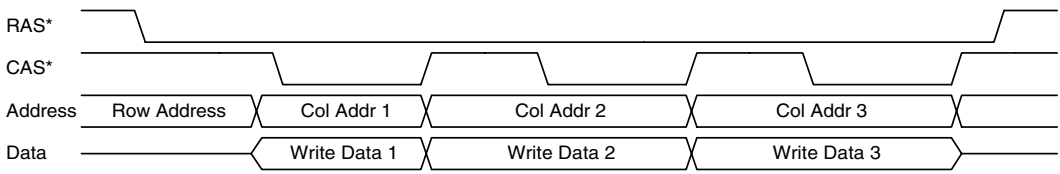


**FIGURE 4.15** Page-mode writes.