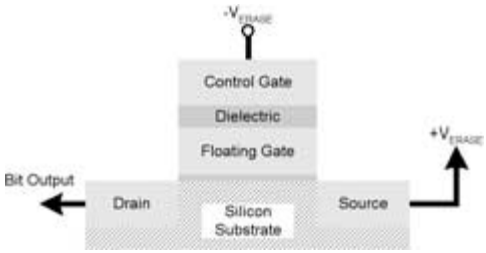

FIGURE 4.5 Flash bit erasure.

smart enough to track how many times each portion of a flash device has been erased and dynamically make decisions about where to place new data accordingly.

Flash chips are offered in two basic categories, NOR and NAND, named according to the circuits that make up each memory bit. NOR flash is a random access architecture that often functions like an EPROM when reading data. NOR memory arrays are directly accessed by a microprocessor and are therefore well suited for storing boot code and other programs. NAND flash is a sequential access architecture that segments the memory into many pages, typically 256 or 512 bytes. Each page is accessed as a discrete unit. As such, NAND flash does not provide the random access interface of a NOR flash. In return for added interface complexity and slower response time, NAND flash provides greater memory density than NOR flash. NAND's greater density makes it ideal for bulk data storage. If programs are stored in NAND flash, they must usually be loaded into RAM before they can be executed, because the NAND page architecture is not well suited to a microprocessor's read/write patterns. NAND flash is widely used in consumer electronic memory cards such as those used in digital cameras. NAND flash devices are also available in discrete form for dense, nonvolatile data storage in a digital system.

NOR flash is discussed here because of its direct microprocessor interface capability. When operating in read-only mode, many NOR flash devices function similarly to EPROMs with a simple asynchronous interface. More advanced flash devices implement high-performance synchronous burst transfer modes that increase their bandwidth for special applications. Most NOR flash chips, however, are used for general processor boot functions where high memory bandwidth is not a main concern. Therefore, an inexpensive asynchronous interface *a la* 27xxx is adequate.

Writing to flash memory is not as simple as presenting new data to the chip and then applying a write enable, as is done with a RAM. Like an EPROM, an already programmed bit must first be erased before it can be reprogrammed. This erasure process takes longer than a simple read access. As Fig. 4.5 shows, the programming and source contacts of each flash bit must be switched to special voltage levels for erasure. Instead of building switches for each individual bit, the complexity of the silicon implementation is reduced by grouping many bits together into blocks. Therefore, a flash device is not erased one bit or byte at a time, but rather a block at a time. Flash chips are segmented into multiple blocks, depending on the particular device and manufacturer. This block architecture is beneficial in that the whole device does not have to be erased, allowing sensitive information to be preserved. A good system design takes the flash block structure into account when deciding where to locate certain pieces of data or sections of software, thereby requiring the erasure of only a limited number of blocks when performing an update of system software or configuration. The block erasure process takes a relatively long time when measured in microprocessor clock cycles. Given that the erase procedure clears an entire range of memory, special algorithms are built into the chips to protect the blocks by requiring a special sequence of flash accesses before the actual erase process is initiated.

Flash chips are not as standard as EPROMs, because different manufacturers have created their own programming algorithms, memory organizations, and pin assignments. Many conventional parallel data-bus devices have part numbers with "28F" or "29F" prefixes. For example, Advanced Micro Devices' flash memory family is the 29Fxxx. Intel's family is the 28Fxxx. Aside from programming differences, the size and organization of blocks within a flash device is a key functional difference that may make one vendor's product better than another for a particular application. Two main attributes of flash chips are uniformity of block size and hardware protection of blocks.

Uniform-block devices divide the memory array into equally sized blocks. Boot-block devices divide the memory array into one or more small *boot blocks* and then divide the remainder of memory into equally sized blocks. Boot-block devices are popular, because the smaller boot blocks can be used to hold the rarely touched software that is used to initialize the system's microprocessor when it first turns on. Boot code is often a small fraction of the system's overall software. Due to its critical nature, boot code is often kept simple to reduce the likelihood of errors. Therefore, boot code is seldom updated. In contrast, other flash ROM contents, such as application code and any application data, may be updated more frequently. Using a boot-block device, a microprocessor's boot code can be stored away into its own block without wasting space and without requiring that it be disturbed during a more general software update. Applications that do not store boot code in flash may not want the complexity of dealing with nonuniform boot blocks and may therefore be better suited to uniform-block devices.

Hardware protection of blocks is important when some blocks hold very sensitive information whose loss could cause permanent damage to the system. A common example of this is boot code stored in a boot block; if the boot code is corrupted, the CPU will fail to initialize properly the next time it is reset. A flash device can implement a low-level protection scheme whereby write/erase operations to certain blocks can be disabled with special voltage levels and data patterns presented to the device.

Examples of real flash devices serve well to explain how this important class of nonvolatile memory functions. Advanced Micro Devices (AMD) manufactures two similar flash devices: the 29LV010B and the 29LV001B. Both devices are 3.3-V, 1-MB, 128k × 8 parts that offer hardware sector protection. The 29LV010B is a uniform-sector device, and the 29LV001B is a boot-sector device. AMD uses the term *sector* instead of block. Both chips have the same basic functional bock diagram shown in Fig. 4.6.

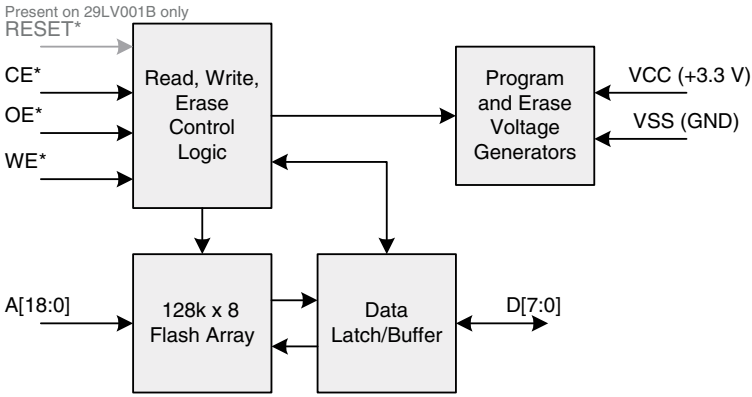

FIGURE 4.6 AMD 29LV010B/29LV001B block diagram.