Direct Memory Access

Parts taken from Wikipedia, the free encyclopedia

Direct memory access (**DMA**) is a feature of modern computers that allows certain hardware subsystems within the computer to access system memory for reading and/or writing independently of the central processing unit. Many hardware systems use DMA including disk drive controllers, graphics cards, network cards, and sound cards. Computers that have DMA channels can transfer data to and from devices with much less CPU overhead than computers without a DMA channel.

Without DMA, using programmed input/output (PIO) mode, the CPU typically has to be occupied for the entire time it's performing a transfer. With DMA, the CPU would initiate the transfer, do other operations while the transfer is in progress, and receive an interrupt from the DMA controller once the operation has been done. This is especially useful in real-time computing applications where not stalling behind concurrent operations is critical.

Principle

DMA is an essential feature of all modern computers, as it allows devices to transfer data without subjecting the CPU to a heavy overhead. Otherwise, the CPU would have to copy each piece of data from the source to the destination. This is typically slower than copying normal blocks of memory since access to I/O devices over a peripheral bus is generally slower than normal system RAM. During this time the CPU would be unavailable for any other tasks involving CPU bus access, although it could continue doing any work which did not require bus access.

A DMA transfer essentially copies a block of memory from one device to another. While the CPU initiates the transfer, it does not execute it. For so-called "third party" DMA, the transfer is performed by a DMA controller which is typically part of the motherboard chipset. More advanced bus designs such as PCI typically use bus mastering DMA, where the device takes control of the bus and performs the transfer itself.

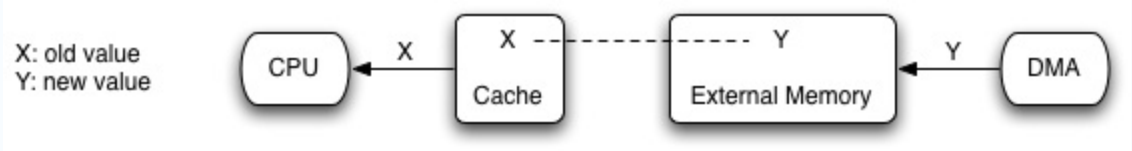
A typical usage of DMA is copying a block of memory from system RAM to or from a buffer on the device. Such an operation does not stall the processor, which as a result can be scheduled to perform other tasks. DMA transfers are essential to high performance embedded systems. It is also essential in providing so-called zero-copy implementations of peripheral device drivers as well as functionalities such as network packet routing, audio playback and streaming video.

Cache coherency problem

DMA can lead to cache coherency problems. Imagine a CPU equipped with a cache and an external memory, which can be accessed directly by devices using DMA. When the CPU

accesses location X in the memory, the current value will be stored in the cache. Subsequent operations on X will update the cached copy of X, but not the external memory version of X. If the cache is not flushed to the memory before the next time a device tries to access X, the device will receive a stale value of X.

Similarly, if the cached copy of X is not invalidated when a device writes a new value to the memory, then the CPU will operate on a stale value of X.



DMA engines

In addition to hardware interaction, DMA can also be used to offload expensive memory operations, such as large copies or scatter-gather operations, from the CPU to a dedicated DMA engine. While normal memory copies are typically too small to be worthwhile to offload on today's desktop computers, they are frequently offloaded on embedded devices due to more limited resources.

DMA Master vs. Slave

In Embedded Systems, often two kinds of components: master and slave. Slave interface is similar to Programmed I/O through which the software (running on embedded CPU) can write/read I/O registers or (less commonly) local memory block inside the device. Master interface can be used by the device to perform DMA transaction to/from system memory without heavily loading the CPU.

Therefore high bandwidth device/hardware such as network controller that need to transfer huge amount of data to/from system memory will have two interface adapters' buses: master and slave interface. Like PCI, no central DMA controller is required since the DMA is bus-mastering, but an arbiter is required in case of multiple masters present on the system.

Internally, a multichannel DMA engine is able to perform multiple concurrent scatter/gather operations as programmed by the software. Basically the hardware supports a linked list of accesses.