Access memory directly

In this article we will show you some of the typical architectures used to implement direct memory access.

Network Management - In this article, I will show you some of the typical architectures used to implement direct memory access.

Direct memory access (Direct Memory Access or DMA for short) is an important component in any modern computer architecture. DMA allows CPUs to load offload memory stacks increasingly to other components. This method frees the CPU from errands and provides more cycles to solve more complex tasks.

For example, perhaps some of you would like to save this article to your hard drive so you can read the reference later. When you choose to do this, you will need to select a location to store the article. Now the data is retrieved by the network card and then routed to the desired location on your hard drive. A very simple task and the ability of the CPU to inherit such types of operations.

Although the main idea of ??direct memory access techniques is quite simple, direct memory access operations are complex. As you can imagine, when there are multiple devices (or peripherals) that want access to memory locations, the problem will occur. This is the reason or request for a DMA controller. DMA controller is essentially a device that controls all DMA operations.

In order to control DMA activities, DMA controllers need to have information related to upcoming activities. This information includes things like source and destination addresses, mode, and data size that are transmitted. DMA controllers also need to be knowledgeable about what it will transmit, how it will be delivered and how it will be delivered, how long it will take to do it. With this knowledge, DMA controllers can require memory bus control from the CPU. When the CPU is ready to yield the memory bus control, it sends an ACK response signal in response to the DMA controller's required signal.

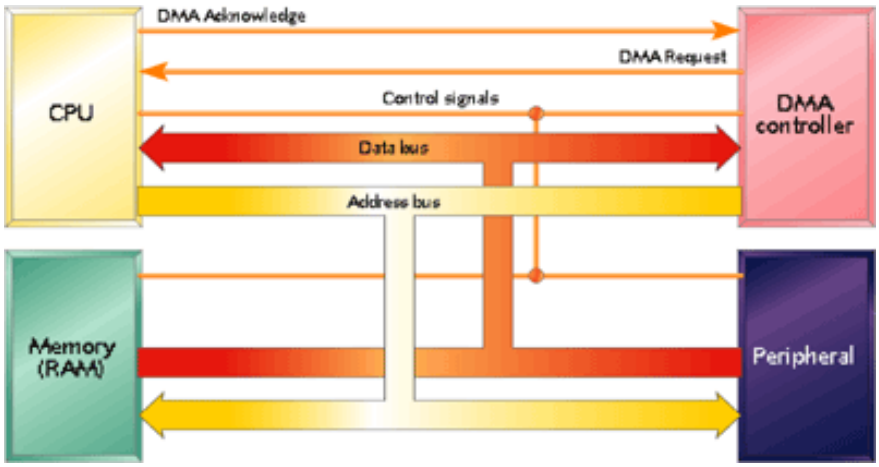

Figure 1: Direct Memory Access interaction

Burst or Single-cycle

What happens after the DMA controller increases the control of the memory bus completely depends on the previous DMA controller mode specified. There are two general modes of DMA controller operation. The first is *burst* . When the DMA controller operates in burst mode, it maintains the control of the memory bus during the memory transfer period. The downside in burst mode is that the CPU cannot access the memory bus until the DMA controller completes the memory transfer. At this time, the CPU can still access its L1 and L2 caches but cannot access other memory; This has limited the CPU to perform the task and may cause it to wait until the DMA controller completes the memory transfer and then returns the bus control back to itself.

To avoid the situation where the CPU is forced to wait for the complete memory transfer, the DMA controller can operate in another mode with a *single-cycle* name. The way it works in the *single-cycle* mode is as follows, the DMA controller will return the memory bus control after each memory block has been transferred. The size of each memory block usually falls into 256 or 512 bytes. This allows the CPU to have more opportunities to use the memory bus for private purposes without having to wait for a sizable amount of time to complete the memory transfer as in *b urst* mode. However, there is a downside to DMA operation in this *single-cycle* mode. When the DMA controller returns bus control, it must send a request to the CPU to increase bus control, then wait for the ACK response from the CPU before it can increase bus control to perform the transmission. Other memory blocks. Thus, repeated repetition of the request / response string will consume time to complete the memory transfer process.

Single-cycle operation mode is most commonly used, although most DMA controllers are available for both modes. The optimal length of each block that DMA will transmit before returning bus control and requiring control is a rather complex issue. A key factor in determining the optimal size of a block is the observed error rate. When there is a high error rate, the smaller the length of the block, the better. However, it is important to know that, when there are fewer errors due to reducing the block length, we will suffer losses in increasing the request / response process between DMA controller and CPU. The decision to implement is dependent on each DMA controller manufacturer (especially the engineers who designed it!). If you want to turn this way on how to define a specific DMA controller, you can search for it somewhere in the document, otherwise you can request it from the company.

When the DMA controller has completed the transmission of a block and returns the memory bus control to the CPU, the CPU will now be able to access the bus to perform its own purposes. In the example we used above and in many other examples have provided the CPU with an opportunity to update progress indicators and update new information relevant to the DMA activity being performed for the system. file system.

Cache Coherency

Another problem that arises during DMA operation is *cache coherency* . When the CPU accesses a memory location, the value of that location is stored in the memory cache of the CPU. If the DMA operation is related to this memory location, the value in the memory cache of the CPU may not match the value at the actual memory location.

To overcome this problem, there are two solutions given. The entire cache consolidation system needs to implement a hardware solution in which the DMA controller sends a signal to the cache controller when it wishes to access the memory location. If DMA wants to write to that location, the cache controller will invalidate the CPU cache memory value. If the DMA wants to read the memory location, the cache controller

will clear the CPU cache to ensure that the memory location contains the latest value (CPU cache value). This method of operation requires some extra overhead but ensures CPU cache tightness.

In incomplete cache consolidation systems, memory cache maintenance is left to the operating system. The operating system will receive a request to decide whether the cache will be deleted first for DMA activity or invalidate later. Which method is better? We are not sure if there is an exact answer because both methods are satisfactory. However, the way we choose and recommend that you implement hardware in the entire cache consolidation systems.