Direct Memory Access And 8237 DMA Controller

Varun Saluja, Vanita
Dronacharya College of Engineering

ABSTRACT: **Many hardware systems use DMA, including disk drive controllers, graphics card, network cards and sound cards. DMA is used for intra-chip data transfer in multi-core processors. DMA can be used for "memory to memory" copying or moving of data within memory. An implementation example is the I/O Acceleration Technology. In recent days we can find out that the DMAC has 8 channels which support hardware and software triggers, linking operation and channel chaining transfer and provides three dimensions transmission by parameter sets so as to perform data block moving, data sorting and subframe extraction of various data structures. Moreover the DMAC supports incrementing and wrapping addressing modes and completes data transfer. And the DMAC could adopt buffer and non-buffer data transfer mode according to the speed of equipments. This review paper covers a detailed study about DMA and 8237 DMA CONTROLLER.**

I. INTRODUCTION

DMA stands for "Direct Memory Access". DMA is a method of transferring data from the computer's RAM to another part of the computer without processing it using theCPU. While most data that is input or output from your computer is processed by the CPU, some data does not require processing, or can be processed by another device. In these situations, DMA can save processing time and is a more efficient way to move data from the computer's memory to other devices.

For example, a sound card may need to access data stored in the computer's RAM, but since it can process the data itself, it may use DMA to bypass the CPU. Video cards that support DMA can also access the system memory and process graphics without needing the CPU. Ultra DMA hard drives use DMA to transfer data faster than previous hard drives that required the data to first be run through the CPU.

In order for devices to use direct memory access, they must be assigned to a DMA channel. Each type of port on a computer has a set of DMA channels that can be assigned to each connected device. For example, a PCI controller and a hard drive controller each have their own set of DMA channels.

II. PRINCIPLE

A DMA controller can generate memory addresses and initiate memory read or write cycles. It contains several processor registers that can be written and read by the CPU. These include a memory address register, a byte count register, and one or more control registers. The control registers specify the I/O port to use, the direction of the transfer (reading from the I/O device or writing to the I/O device), the transfer unit (byte at a time or word at a time), and the number of bytes to transfer in one burst.[1]

To carry out an input, output or memory-to-memory operation, the host processor initializes the DMA controller with a count of the number of words to transfer, and the memory address to use. The CPU then sends commands to a peripheral device to initiate transfer of data. The DMA controller then provides addresses and read/write control lines to the system memory. Each time a byte of data is ready to be transferred between the peripheral device and memory, the DMA controller increments its internal address register until the full block of data is transferred.

DMA transfers can either occur one byte at a time or all at once in burst mode. If they occur a byte at a time, this can allow the CPU to access memory on alternate bus cycles – this is called cycle stealing since the DMA controller and CPU contend for memory access. In burst mode DMA, the CPU can be put on hold while the DMA transfer occurs and a full block of possibly hundreds or thousands of bytes can be moved.[2] When memory cycles are much faster than processor cycles, an interleaved DMA cycle is possible, where the DMA controller uses memory while the CPU cannot.

In a bus mastering system, both the CPU and peripherals can be granted control of the memory bus. Where a peripheral can become bus master, it can directly write to system memory without involvement of the CPU, providing memory address and control signals as required. Some measure must be provided to put the processor into a hold condition so that bus contention does not occur.

III. MODES OF OPERATION

Burst mode

An entire block of data is transferred in one contiguous sequence. Once the DMA controller is granted access to the system bus by the CPU, it transfers all bytes of data in the data block before releasing control of the system buses back to the CPU, but renders the CPU inactive for relatively long periods of time. The mode is also called "Block Transfer Mode". It is also used to stop unnecessary data.

Cycle stealing mode

The cycle stealing mode is used in systems in which the CPU should not be disabled for the length of time needed for burst transfer modes. In the cycle stealing mode, the DMA controller obtains access to the system bus the same way as in burst mode, using **BR (Bus Request) and BG (Bus Grant) signals**, which are the two signals controlling the interface between the CPU and the DMA controller. However, in cycle stealing mode, after one byte of data transfer, the control of the system bus is deasserted to the CPU via BG. It is then continually requested again via BR, transferring one byte of data per request, until the entire block of data has been transferred. By continually obtaining and releasing the control of the system bus, the DMA controller essentially interleaves instruction and data transfers. The CPU processes an instruction, then the DMA controller transfers one data value, and so on. On the one hand, the data block is not transferred as quickly in cycle stealing mode as in burst mode, but on the other hand the CPU is not idled for as long as in burst mode. Cycle stealing mode is useful for controllers that monitor data in real time.

Transparent mode

The transparent mode takes the most time to transfer a block of data, yet it is also the most efficient mode in terms of overall system performance. The DMA controller only transfers data when the CPU is performing operations that do not use the system buses. It is the primary advantage of the transparent mode that the CPU never stops executing its programs and the DMA transfer is free in terms of time. The disadvantage of the transparent mode is that the hardware needs to determine when the CPU is not using the system buses, which can be complex.

IV. EXAMPLES

ISA

Short for **Industry Standard Architecture**, **ISA** was introduced by IBM and headed byMark Dean. ISA was originally an 8-bit computer bus that was later expanded to a 16-bit bus in 1984. When this bus was originally released it was a proprietary bus, which allowed only IBM to create peripherals and the actual interface. However, in the early 1980's other manufacturers were creating the bus.

In 1993, Intel and Microsoft introduced a PnP ISA bus that allowed the computer to automatically detect and setup computer ISA peripherals, such as a modem or sound card. Using the PnP technology, an end-user would have the capability of connecting a device and not having to configure the device using jumpers or dip switches.

All recent computers today no longer included the ISA slots and instead are using more PCI, AGP, and other slots. Below is a graphic of what an ISA expansion card may look like as well as the slot it connects into on the motherboard.

V. SUMMARY

DMA (Direct Memory Access) is a way for I/O devices to bypass the CPU when accessing memory. This is done by sending the same signals to memory that the CPU would send.

Direct memory access (DMA) channels are system pathways used by many devices to transfer information directly to and from memory. DMA channels are not nearly as "famous" as IRQs as system resources go. This is mostly for a good reason: there are fewer of them and they are used by many fewer devices, and hence they usually cause fewer problems with system setup. However, conflicts on DMA channels can cause very strange system problems and can be very difficult to diagnose. DMAs are used most commonly today by floppy disk drives, tape drives and sound cards.

The DMA controller, built into the system chipset on modern PCs, manages standard DMA transfers. The original PC and XT had one of these controllers and supported 4 DMA channels, 0 to 3.

Starting with the IBM AT, a second DMA controller was added. Much in the way that the second interrupt controller was cascaded with the first, the first DMA

controller is cascaded to the second. The difference is that with IRQs, the second controller is cascaded to the first, but with DMAs the first is cascaded to the second. As a result, there are 8 DMAs, from 0 to 7, but DMA 4 is not usable. There is no rerouting as with IRQ2 and IRQ9 here, because all of the original DMAs (0 to 3) are still usable directly.