Advanced Computer Architecture

Lecture No. 31

Reading Material

Vincent P. Heuring & Harry F. Jordan	Chapter 8
Computer Systems Design and Architecture	8.4

Summary

- Direct Memory Access (DMA):

Direct Memory Access (DMA):

Introduction

Direct Memory Access is a technique which allows a peripheral to read from and/or write to memory without intervention by the CPU. It is a simple form of bus mastering where the I/O device is set up by the CPU to transfer one or more contiguous blocks of memory. After the transfer is complete, the I/O device gives control back to the CPU.

The following DMA transfer combinations are possible:

- Memory to memory
- Memory to peripheral
- Peripheral to memory
- Peripheral to peripheral

The DMA approach is to "turn off" (i.e., tri-state and electrically disconnect from the system buses) the CPU and let a peripheral device (or memory - another module or another block of the same module) communicate directly with the memory (or another peripheral).

ADVANTAGE: Higher transfer rates (approaching that of the memory) can be achieved.

DISADVANTAGE: A DMA Controller, or a DMAC, is needed, making the system complex and expensive.

Generally, DMA requests have priority over all other bus activities, including interrupts. No interrupts may be recognized during a DMA cycle.

Reason for DMA:

The instruction **load [2], [9]** is illegal. The symbols [2] and [9] represent memory locations. This transfer has to be done in two steps:

- **load r1,[9]**
- **store r1,bx**

Thus, it is not possible to transfer from one memory location to another without involving the CPU. The same applies to transfer between memory and peripherals connected to I/O ports. e.g., we cannot have **out [6], datap.** It has to be done in two steps:

- **load r1,[6]**
- **out r1, datap**

Similar comments apply to the `in` instruction.
Thus, the real cause of the limited transfer rate is the CPU itself. It acts as an unnecessary "middleman". The above discussion also implies that, in general, every data word travels over the system bus twice.

Some Definitions:

- **MASTER COMPONENT:** A component connected to the system bus and having control of it during a particular bus cycle.
- **SLAVE COMPONENT**: A component connected to the system bus and with which the master component can communicate during a particular bus cycle. Normally the CPU with its bus control logic is the master component.
- **QUALIFICATIONS TO BECOME A MASTER:** A Master must have the capability to place addresses on the address bus and direct the bus activity during a bus cycle.
- **QUALIFIED COMPONENTS:**
    - Processors with their associated bus control logic.
    - DMA controllers.
- **CYCLE STEALING**: Taking control of the system bus for a few bus cycles.

Data Transfer using DMA:
Data transfer using DMA takes place in three steps.
1st Step:
in this step when the processor has to transfer data it issues a command to the DMA controller with the following information:

- Operation to be performed i.e., read or write operation.
- Address of I/O device.
- Address of memory block.
- Size of data to be transferred.

After this, the processor becomes free and it may be able to perform other tasks.
2nd Step:
In this step the entire block of data is transferred directly to or from memory by the DMA controller.
3rd Step:
In this, at the end of the transfer, the DMA controller informs the processor by sending an interrupt signal.

See figure 8.18 on the page number 400 of text book.
The DMA Transfer Protocol:
Most processors have a separate line over which an external device can send a request for DMA. There are various names in use for such a line. HOLD, RQ, or Bus Request (BR), etc. are examples of these names.

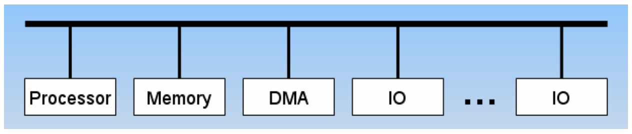
The DMA cycle usually begins with the alternate bus master requesting the system bus by activating the associated Bus Request line and, of course, satisfying the setup and hold times. The CPU completes the current bus cycle, in the same way as it does in case of interrupts, and responds by floating the address, data and control lines. A Bus Grant pulse is then output by the CPU to the same device from where the request occurred. After receiving the Bus Grant pulse, and waiting for the "float delay" of the CPU, the requesting device may drive the system bus. This precaution prevents bus contention. To return control of the bus to the CPU, the alternate bus master relinquishes bus control and issues a release pulse on the same Bus Request line. The CPU may drive the system bus after detecting the release pulse. The alternate bus master should be tri-stated off the local bus and have other CPU interface circuits re-enabled within this time.

DMA has priority over Interrupt driven I/O:
In interrupt driven I/O the I/O transfer depends upon the speed at which the processor tests and service a device. Also, many instructions are required for each I/O transfer. These factors become bottleneck when large blocks of data are to be transferred. While in the DMA technique the I/O transfers take place without the intervention by the CPU, rather CPU pauses for one bus cycle. So DMA technique is the more efficient technique for I/O transfers.

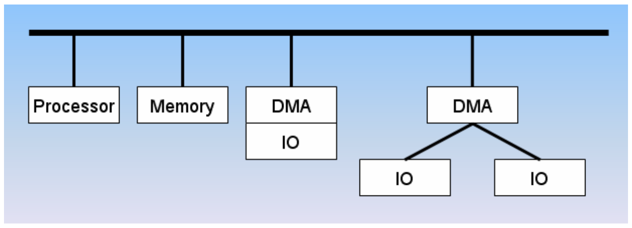
DMA Configurations:

- Single Bus Detached DMA
- Single Bus Integrated DMA
- I/O Bus

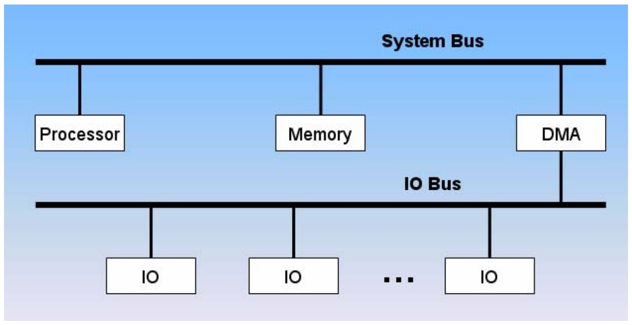
Single Bus Detached DMA
In the example provided by the above diagram, there is a single bidirectional bus connecting the processor, the memory, the DMA module and all the I/O modules. When a particular I/O module needs to read or write large amounts contiguous data it requests the processor for direct memory access. If permission is granted by the processor, the I/O module sends the read or write address and the size of data needed to be read or written to the DMA module. Once the DMA module acknowledges the request, the I/O module is free to read or write its contiguous block of data from or onto main memory. Even though in this situation the processor will not be able to execute while the transfer is going on (as there is a just a single bus to facilitate transfer of data), DMA transfer is much faster then having each word of memory being read by the processor and then being written to its location.



Single Bus Integrated DMA
In this configuration the DMA and one or more I/O modules are integrated without the inclusion of system bus functioning as the part of I/O module or may be as a separate module controlling the I/O module.



IO Bus
In this configuration we integrate the DMA and I/O modules through an I/O bus. So it will cut the number of I/O interfaces required between DMA and I/O module.



Example
An I/O device transfers data at a rate of 10MB/s over a 100MB/s bus. The data is transferred in 4KB blocks. If the processor operates at 500MHz, and it takes a total of 5000 cycles to handle each DMA request, find the fraction of CPU time handling the data transfer with and without DMA.

Solution.
Without DMA
The processor here copies the data into memory as it is sent over the bus. Since the I/O device sends data at a rate of 10MB/s over the 100MB/s bus, 10 % of each second is spent transferring data. Thus 10% of the CPU time is spent copying data to memory.
With DMA
Time required in handling each DMA request is 5000 cycles. Since 2500 DMA requests are issued (10MB/4KB) the total time taken is 12,500,000 cycles. As the CPU clock is 500MHZ, the fraction of CPU time spent is 12,500,000/(500x10^{6}) or 2.5%.

Example
A hard drive with a maximum transfer rate of 1Mbyte/sec is connected to a 32-bit, 10MIPS CPU operating at a clock frequency of 100 MHz. Assume that the I/O interface is DMA based and it takes 500 clock cycles for the CPU to set-up the DMA controller. Also assume that the interrupt handling process at the end of the DMA transfer takes an additional 300 CPU clock cycles. If the data transfer is done using 2 KB blocks, calculate the percentage of the CPU time consumed in handling the hard drive.

Solution
Since the hard drive transfers at 1MB/sec, and each block size is 2KB, there are

1000/2= 500 blocks transferred/sec

Every DMA transfer uses 500+300=800 CPU cycles. This gives us

$$800 \times 500 = 400{,}000 = 400 \times 10^3 \text{ cycles/sec}$$

For the 100 MHz CPU, this corresponds to

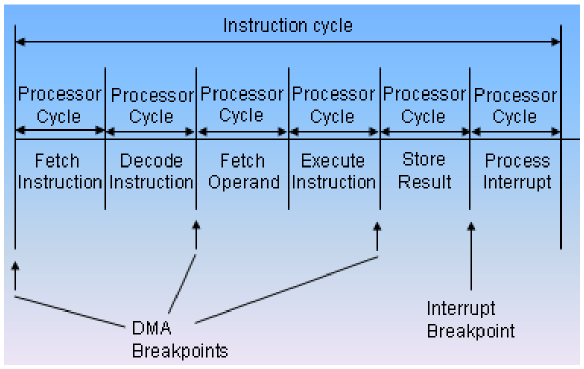
$$(400 \times 10^3) / (100 \times 10^6) = 4 \times 10^{-3} = 0.4\%$$

This would be the case when the hard drive is transferring data all the time. In actual situation, the drive will not be active all the time, and this number will be much smaller than 0.4%.

Another assumption that is implied in the previous example is that the DMA controller is the only device accessing the memory. If the CPU also tries to access memory, then either the DMAC or the CPU will have to wait while the other one is actively accessing the memory. If cache memory is also used, this can free up main memory for use by the DMAC.

Cycle Stealing

The DMA module takes control of the bus to transfer data to and from memory by forcing the CPU to temporarily suspend its operation. This approach is called Cycle Stealing because in this approach DMA steals a bus cycle.



DMA and Interrupt breakpoints during an instruction cycle

The figure shows that the CPU suspends or pauses for one bus cycle when it needs a bus cycle, transfers the data and then returns the control back to the CPU.

I/O processors

When I/O module has its own local memory to control a large number of I/O devices without the involvement of CPU is called I/O processor.

I/O Channels

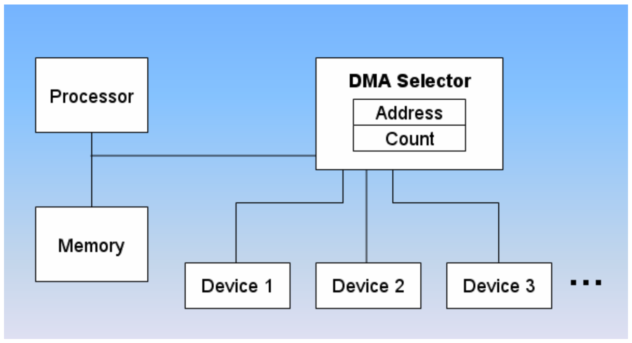
When an I/O module has a capability of executing a specific set of instructions for specific I/O devices in the memory without the involvement of CPU is called I/O channel.

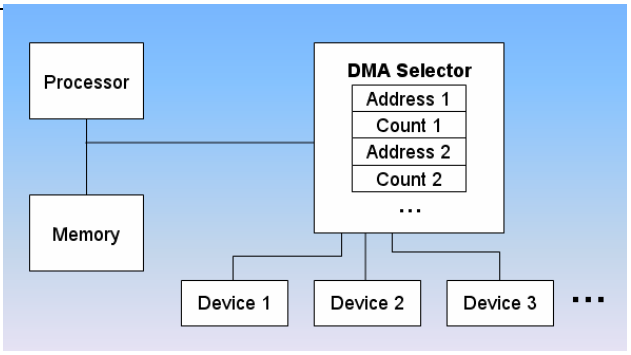
I/O channel architecture:

Types of I/O channels:

Selector Channel

It is the DMA controller that can do block transfers for several devices but only one at a time.





Multiplexer Channel
It is the DMA controller that can do block transfers for several devices at once.

Types of Multiplexer Channel
- Byte Multiplexer
- Block Multiplexer

Byte Multiplexer
- Byte multiplexer accepts or transmits characters.
- Interleaves bytes from several devices.
- Used for low speed devices.

Block Multiplexer
- Block multiplexer accepts or transmits block of characters.
- Interleaves blocks of bytes from several devices.
- Used for high speed devices.

Virtual Address:
Virtual address is generated be the logical by the memory management unit for translation.

Physical Address:
Physical address is the address in the memory.

DMA and memory system
DMA disturbs the relationship between the memory system and CPU.

Direct memory access and the memory system
Without DMA, all memory accesses are handled by the CPU, using address translation and cache mechanism. When DMA is implemented into an I/O system memory accesses can be made without intervening the CPU for address translation and cache access. The problems created by the DMA in virtual memory and cache systems can be solved using hardware and software techniques.

Hardware Software Interface
One solution to the problem is that all the I/O transfers are made through the cache to ensure that modified data are read and updated in the cache on the I/O write. This method can decrease the processor performance because of infrequent usage of the I/O data.
Another approach is that the cache is invalidated for an I/O read and for an I/O write, write-back (flushing) is forced by the operating system. This method is more efficient because flushing of large parts of cache data is only done on DMA block accesses.
Third technique is to flush the cache entries using a hardware mechanism, used in multiprogramming system to keep cache coherent.

SOME clarifications:

- The terms "serial" and "parallel" are with respect to the computer I/O ports --- not with respect to the CPU. The CPU always transfers data in parallel.
- The terms "programmed I/O", "interrupt driven I/O" and "DMA" are with respect to the CPU. Each of these terms refers to a way in which the CPU handles I/O, or the way data flow through the ports is controlled.
- The terms "simplex" and "duplex" are with respect to the transmission medium or the communication link.
- The terms "memory mapped I/O" and "independent I/O" are with respect to the mapping of the interface, i.e., they refer to the CPU control lines used in the interface.