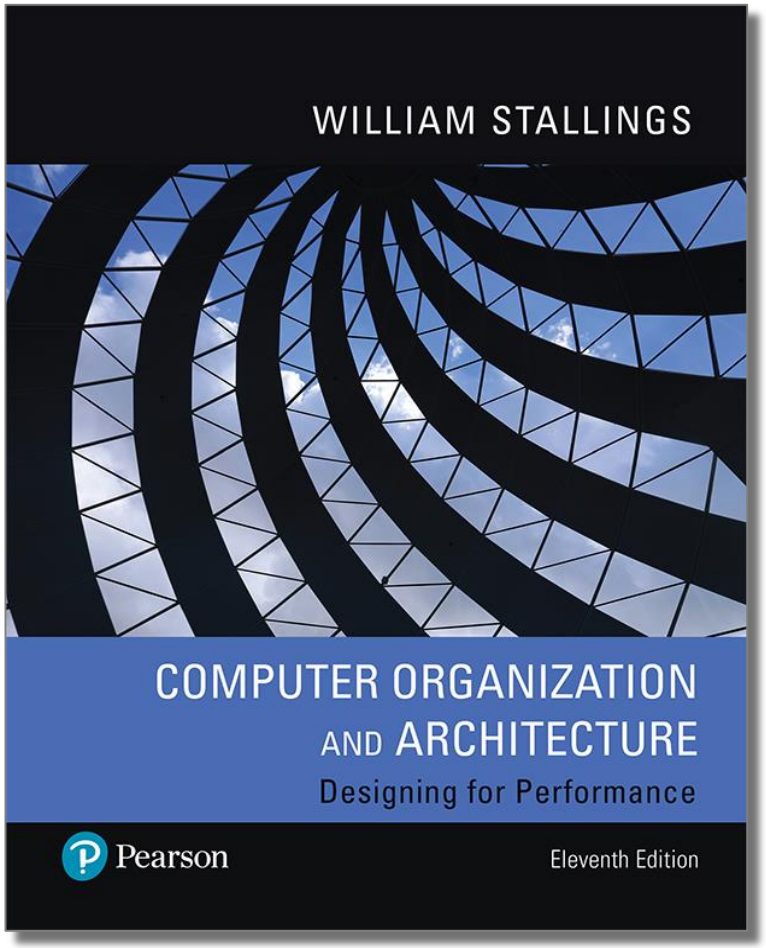
Computer Organization and Architecture

Designing for Performance

11th Edition



Chapter 8

Input/Output

Table 8.1
I/O Techniques

	No Interrupts	Use of Interrupts
I/O-to-memory transfer through processor	Programmed I/O	Interrupt-driven I/O
Direct I/O-to-memory transfer		Direct memory access (DMA)

Drawbacks of Programmed and Interrupt-Driven I/O

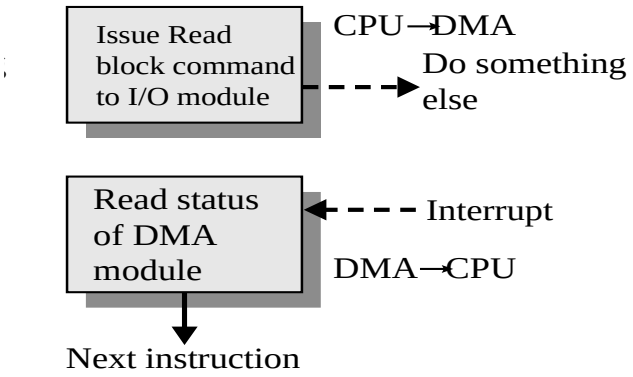
- Both forms of I/O suffer from two inherent drawbacks:
  1) The I/O transfer rate is limited by the speed with which the processor can test and service a device
  2) The processor is tied up in managing an I/O transfer; a number of instructions must be executed for each I/O transfer
- There is a trade-off between these two drawbacks.
- Consider the transfer of a block of data. Using simple programmed I/O, the processor is dedicated to the task of I/O and can move data at a rather high rate, at the cost of doing nothing else.

Drawbacks of Programmed and Interrupt-Driven I/O

- Interrupt I/O frees up the processor to some extent at the expense of the I/O transfer rate. Nevertheless, both methods have an adverse impact on both processor activity and I/O transfer rate.

- When large volumes of data are to be moved a more efficient technique is direct memory access (DMA)

Figure 8.4 – Direct Memory Access (DMA)



(c) Direct memory access

- Direct memory access (DMA)
  - The I/O module and main memory exchange data directly without processor involvement

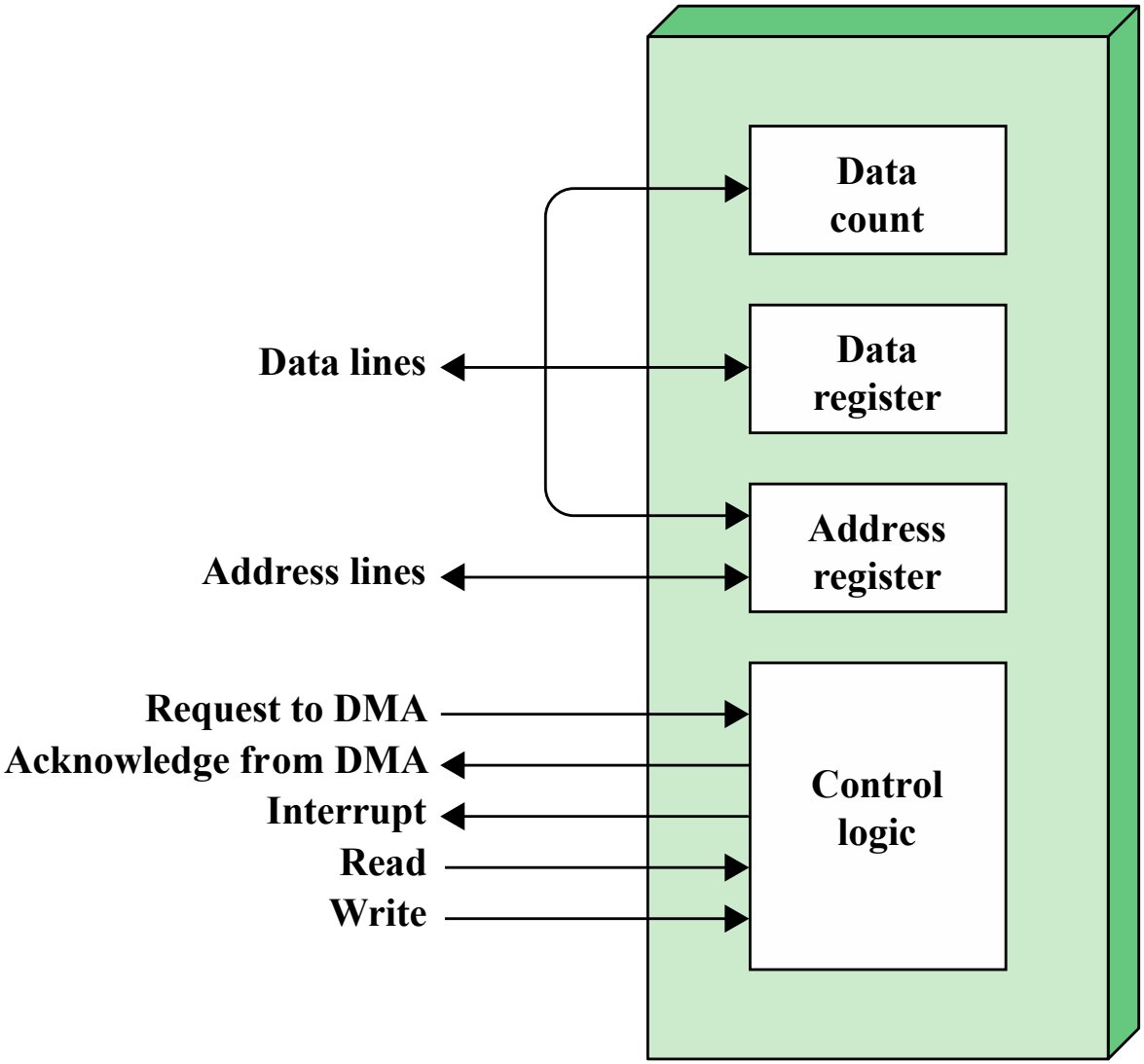
DMA

- DMA involves an additional module on the system bus.
- The DMA module is capable of mimicking the processor and, indeed, of taking over control of the system from the processor.
- It needs to do this to transfer data to and from memory over the system bus.

DMA

- For this purpose, the DMA module must use the bus only when the processor does not need it, or it must force the processor to suspend operation temporarily.
- The latter technique is more common and is referred to as cycle stealing, because the DMA module in effect steals a bus cycle.

Figure 8.12
Typical DMA Block Diagram



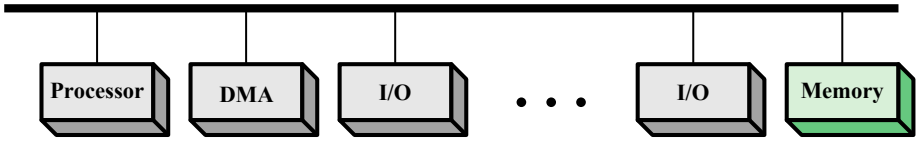
DMA

- When the processor wishes to read or write a block of data, it issues a command to the DMA module, by sending to the DMA module the following information:
  - Whether a read or write is requested, using the read or write control line between the processor and the DMA module
  - The address of the I/O device involved, communicated on the data lines
  - The starting location in memory to read from or write to, communicated on the data lines and stored by the DMA module in its address register
  - The number of words to be read or written, again communicated via the data lines and stored in the data count register

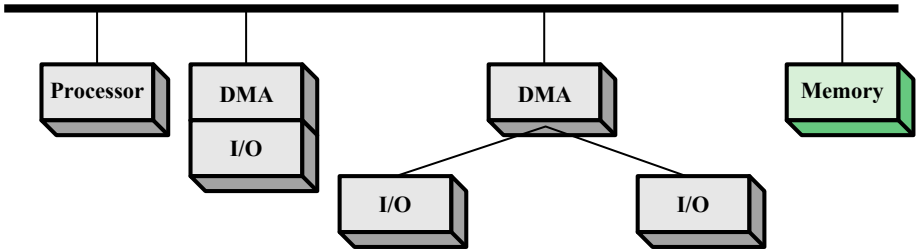
DMA

- The processor then continues with other work since it has **delegated this I/O operation to the DMA module.**
- The DMA module transfers the entire block of data, one word at a time, directly to or from memory, without going through the processor.
- When the transfer is complete, the DMA module sends an interrupt signal to the processor.
- Thus, the processor is involved only at the **beginning and end of the transfer**.

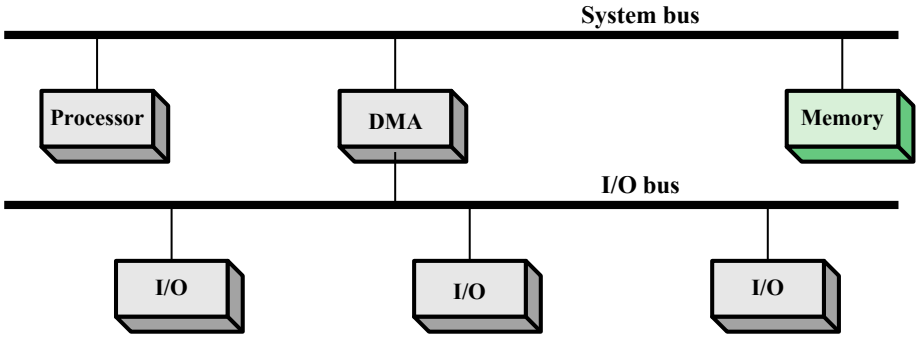
Figure 8.14
Alternative DMA Configurations



(a) Single-bus, detached DMA



(b) Single-bus, Integrated DMA-I/O

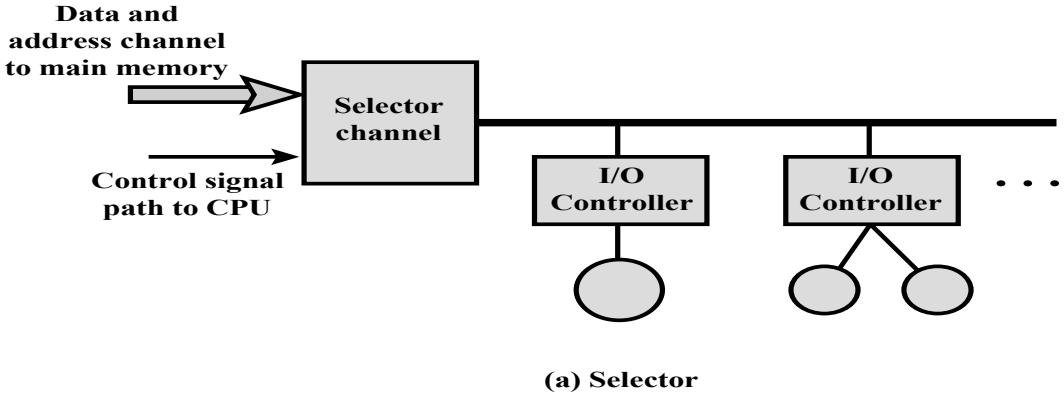


(c) I/O bus

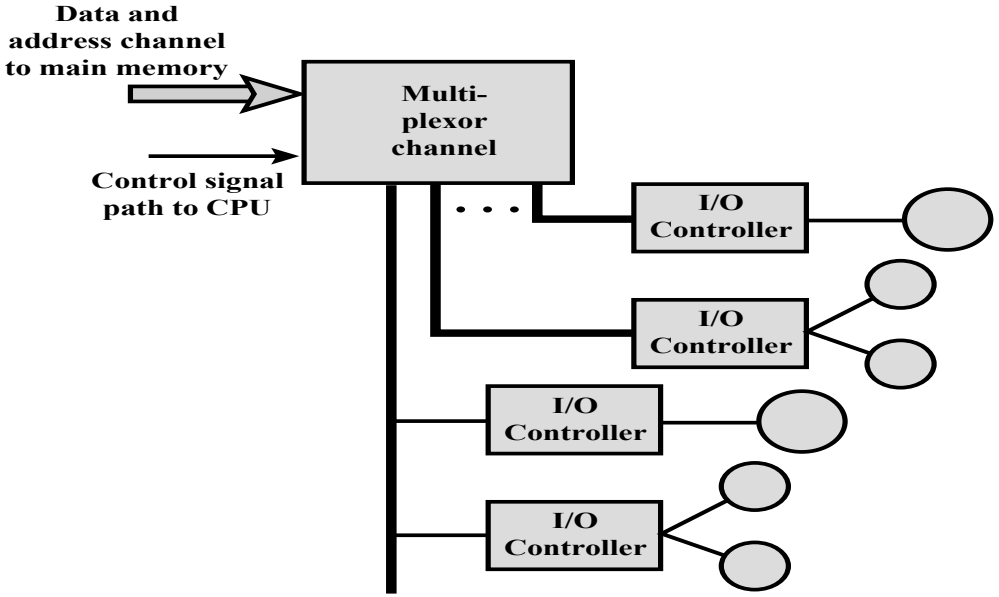
Evolution of the I/O Function

1. The CPU directly controls a peripheral device.
2. A controller or I/O module is added. The CPU uses programmed I/O without interrupts.
3. Same configuration as in step 2 is used, but now interrupts are employed. The CPU need not spend time waiting for an I/O operation to be performed, thus increasing efficiency.
4. The I/O module is given direct access to memory via DMA. It can now move a block of data to or from memory without involving the CPU, except at the beginning and end of the transfer.
5. The I/O module is enhanced to become a processor in its own right, with a specialized instruction set tailored for I/O
6. The I/O module has a local memory of its own and is, in fact, a computer in its own right. With this architecture a large set of I/O devices can be controlled with minimal CPU involvement.

Figure 8.18
I/O Channel Architecture



(a) Selector



(b) Multiplexor

I/O Channel

- A **selector channel** controls multiple high-speed devices and, at any one time, is dedicated to the transfer of data with one of those devices. Thus, the I/O channel selects one device and effects the data transfer.
- Each device, or a small set of devices, is handled by a controller, or I/O module, that is much like the I/O modules we have been discussing.
- Thus, the I/O channel serves in place of the CPU in controlling these I/O controllers.
- A **multiplexor channel** can handle I/O with multiple devices at the same time.

Copyright