Chapter 5: Direct Memory Access

Objectives

To gain insight into the operation of a Direct Memory Access controller.

Definitions

In the discussion on computer architecture and the role of the Central Processing Unit a brief description was given on how the CPU may transfer data to or from a number of external (other than memory) devices. The operation treated the I/O system for reading and writing in the same manner as memory, using address, data lines and WR RD control lines. This requires CPU intervention and is costly in "time".

Direct Memory Access--the ability of an I/O subsystem to transfer data to and from a memory subsystem without processor intervention.

DMA Controller--a device that can control data transfers between an I/O subsystem and a memory subsystem in th same manner that a processor can control such transfers.

DMA Controller

The DMA controller can issue commands to the memory that behave exactly like the commands issued by the CPU. The DMA controller in a sense is a second processor in the system but is dedicated to an I/O function. The DMA controller as shown below connects one or more I/O ports directly to memory, where the I/O data stream passes through the DMA controller faster and more efficiently than through the processor as the DMA channel is specialised to the data transfer task.

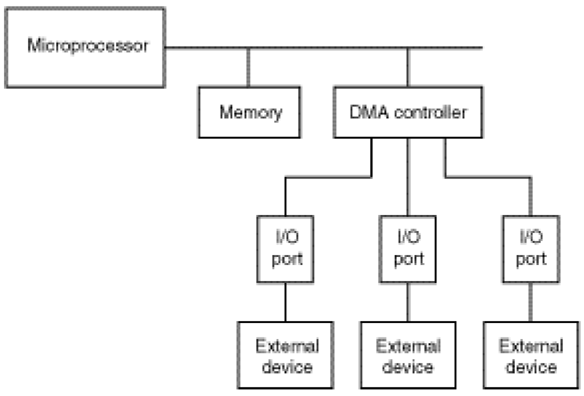


Figure 5-1: A microcomputer with a direct memory-access controller

Stone, H.S. 1982, *Microcomputer interfacing* , Addison-Wesley, Reading, Mass., p. 5.

The DMA interface

The DMA adds one more level of complexity to the I/O interface because a DMA controller has independent access to memory. One set of wires (bus) can carry at most one transaction at a time. If the DMA and a microprocessor share the signal wire to memory there must be a mechanism to arbitrate which shall have access to memory when both attempt to at the same time.

Functional behaviour of a DMA transaction

1. The processor transmits the following information to a DMA controller:

    (a) beginning address in memory
    (b) block length (number of words to transfer)
    (c) direction (memory-to-device or device-to-memory)
    (d) port ID
    (e) end of block action (interrupt request or no interrupt request).

2. The processor returns to other activities while the DMA controller starts the data transfer.

3. Each time the DMA controller accesses memory, it synchronises this memory request with an idle period of the processor--to do this the possibilities are:

    (a) force an immediate disabling of the processor, or
    (b) request a halt of the processor, and await an acknowledgement, or
    (c) time the DMA access to a clock interval or status signal of the processor that signals an idle cycle.

4. When the DMA controller accesses an I/O port or memory, it uses the same functional control signals as used by the processor. I/O port activity can be performed on dedicated lines that do not have to be synchronised with the processor.

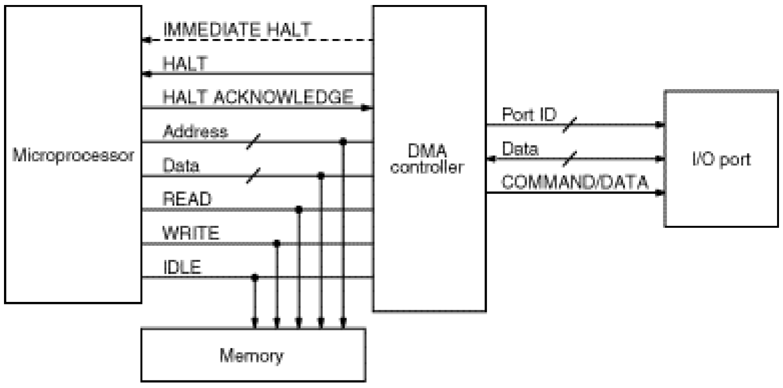
5. At the completion of the block transfer, the DMA controller raises an interrupt request if the interrupts are "armed" and otherwise indicates completion in its status register.

6. The processor recognises I/O completion (either by interrupt or by reading the status register); thereafter the activity between the processor and the DMA controller follows the normal post-completion activity of any I/O port.

This shows that the controller is treated as a standard port before and after block transfer and during transfer the DMA must be able to synchronise with the processor. The controller improves performance especially with a built-in program for moving a stream of data between memory and an I/O port--thereby not requiring to access the instruction from memory and executing them one by one. Some elementary actions can be performed in parallel instead of sequentially when implemented with software in the processor. For example, the controller decrements a counter each time it moves a datum. The controller can overlap the subtraction with memory access and avoid the time penalty for the arithmetic instruction. Because of the ability to achieve higher performance for block transfers, the DMA controller is used most frequently for high speed I/O, especially disk. Fast disks move blocks of data at speeds much greater than any program can control and therefore must be interfaced to computers through DMA controllers.

DMA interface operation

The figure below indicates a typical direct memory-access controller interface. The I/O ports in this example under DMA control are attached only to the DMA controller. Signal lines are the same ones that normally interface the ports to the processor. Memory lines are the conventional lines except that both the processor and the DMA controller exercise the lines. The new lines are the HALT and HALT ACKNOWLEDGE lines. These lines synchronise the DMA controller to the processor. When the DMA controller needs to access the memory it request the processor to halt by asserting the HALT signal. The processor responds with HALT ACKNOWLEDGE at a later time and then the DMA controller takes control of memory. On completion of its task the DMA controller removes its HALT request, the processor continues from its suspension and removes its HALT ACKNOWLEDGE.



The dotted IMMEDIATE HALT is a different type of DMA request. The HALT request may take several clock cycles for the processor to acknowledge due to the time taken for the processor reaching a state where it can suspend processing. Data held in dynamic registers that are refreshed during normal processing must be moved to status registers, or execution has moved to a point where the data is no longer necessary. The IMMEDIATE HALT line avoids this delay but has associated severe restrictions. The IMMEDIATE HALT can be used only briefly one or at most two accesses otherwise the processor may not be able to recover its state correctly and return to the suspended activity.

The IDLE status line can be used by the DMA controllers that can delay data transfers until an IDLE point is reached. The IDLE in some systems can occur frequently, that is, 20 to 30% of the memory cycles. In this situation there is no need to halt the processor and the DMA can achieve high data rates with virtually no impact on processor performance.

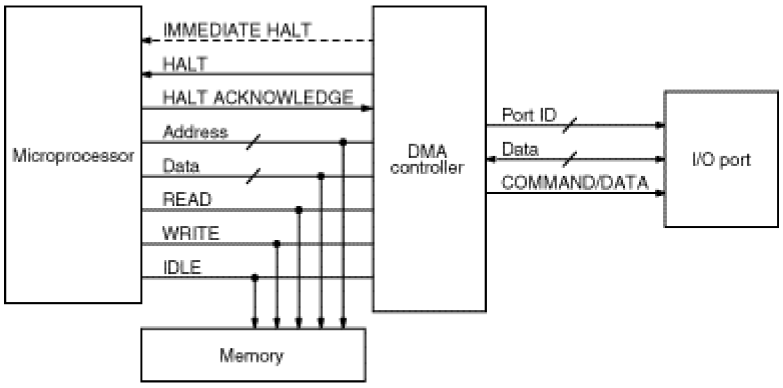


Figure 5-2: A typical direct memory-access controller interface

Stone, H.S. 1982, *Microcomputer interfacing* , Addison-Wesley, Reading, Mass., p. 27.

DMA controllers

DMA controllers were expensive and complex subsystems with complexity comparable to a small processor. Larg Scale Integration (LSI) implementation of the DMA controller has reduced the size to the stage where it can be incorporated on a single chip.

DMA controller operation

The figure below shows the basic structure of a DMA channel and due to it operating in a similar manner to the processor it has a full bus interface. The DMA controller has 3 independent channels, each channel contains an address register, a control register and a byte counter. To transfer a block of data between an external device or I/O port and memory the controller stores initial values in the address control and byte count registers. The DMA channel then transfers the block of information from or to memory according to the direction of the transfer encode in the command register. The starting address of the block in memory is given by the address register, and the length of the block is given by the byte count. To make the transfer the DMA controller has to synchronise the activities of the processor to the external device, before each transfer the controller waits for both the external device to be ready and the processor to be idle.

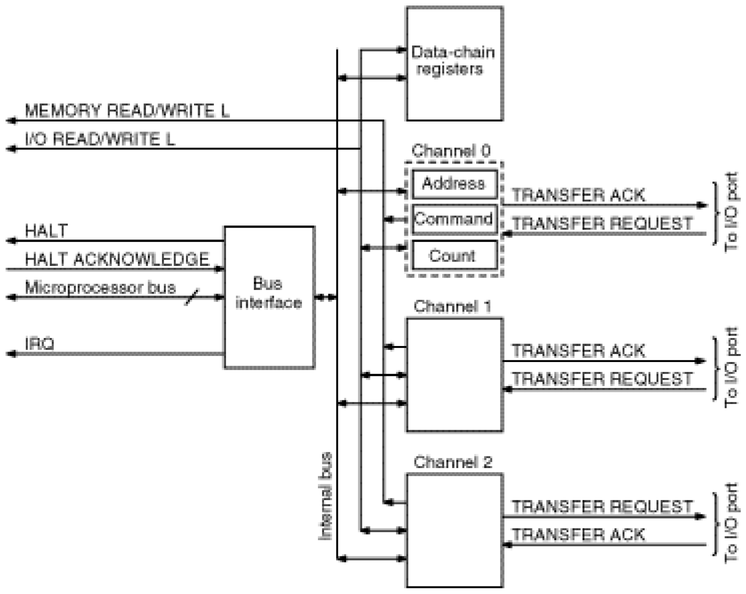


Figure 5-3: The structure of a typical DMA controller

Stone, H.S. 1982, *Microcomputer interfacing* , Addison-Wesley, Reading, Mass., p. 149.

DMA interface with I/O

Interfacing with the I/O port requires two signals per port-- TRANSFER REQUEST and TRANSFER ACK--plus the ability to generate I/O READ/WRITE L to indicate to the port the direction of the transfer. The DMA controlle accepts a TRANSFER REQUEST from the port when the port has data ready to write into memory or has an empty buffer that can accept data from memory. When a transfer is to take place, the DMA outputs the control signal TRANSFER ACK, which indicates the port should receive data from or write data into memory.

Controller accessing memory

As described before the DMA controller has a HALT request O/P signal and a HALT ACKNOWLEDGE I/P. During the byte-by-byte transfer of a block of data, the controller watches for a TRANSFER REQUEST on a channel. Then the controller asserts HALT and waits for HALT ACKNOWLEDGE. This instructs the processor to relinquish the memory bus. When the processor relinquishes the bus it asserts HALT ACKNOWLEDGE and the DMA controller has access to memory. The controller simultaneously:

1. places an address on the bus

2. sends TRANSFER ACK to the requesting I/O port and

3. sends the proper polarity of READ/WRITE L to memory and the complement of this signal to the I/O system.

The I/O port and memory respond in opposite ways so that data moves from one to the other depending on the polarity of READ/WRITE L. At the completion of the memory cycle the DMA controller removes all of the signal placed on the bus and places its bus drivers in high impedance mode. With the HALT deasserted the processor can continue its operation from the point of suspension. In the DMA shown above there are three channels making it possible for the above operations to be multiplexed among the three separate I/O ports.

The microprocessor can be interrupted by the controller at the end of a block transfer. On responding to the interrupt the processor can reload the DMA registers for a new block transfer and thereby maintain the continuity o the I/O flow. It is also possible to achieve the same affect by"chaining" the block transfers within the DMA controller. The " registers" hold the address, byte count control word, plus ID channel which is the data to be used to reload the designated channel's register when that channel completes a transfer.

i8527 DMA controller

The i8527 controller has four independent channels each of which contains an addressregister and a counter. The counter decrements as each byte transfer occurs, and forces termination of the DMA operation after the last transfe The controller increments the address register after each operation, so that successive data transfers are made at contiguous ascending addresses.

The arbiter resolves conflicts among the channels for access to memory. Two methods have been used in this chip to make the chip useful in a variety of different applications. In one mode the channels have a fixed priority and conflicts are resolved according to the priority, for example, Channel 0 has highest priority and Channel 3 lowest. The second mode is a rotating priority scheme in which priority rankings are the four cycle shifts of 0-1-2-3, when a channel is granted access to the bus the priority ranking shifts cyclically to place the channel in the lowest priorit position for the next arbitration cycle.

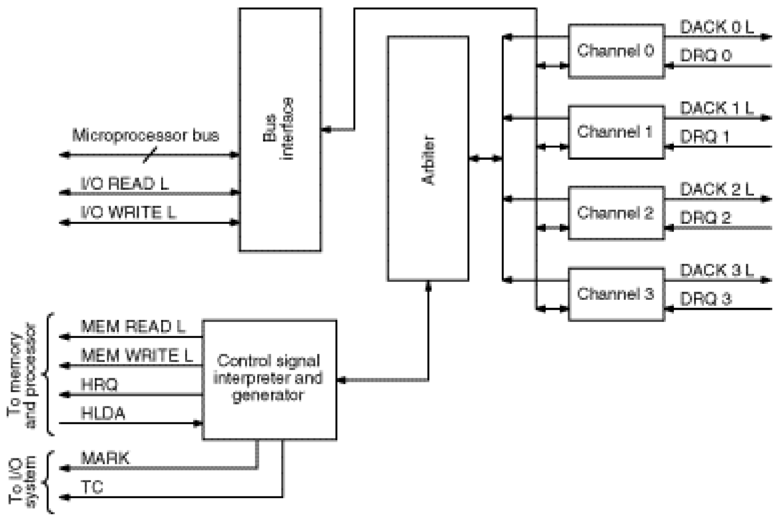


Figure 5-4: Structure of the i8527 DMA controller

Stone, H.S. 1982, *Microcomputer interfacing* , Addison-Wesley, Reading, Mass., p. 152.

The chip has four signals associated with the READ and WRITE operation. MEM READ L and MEM WRITE L are signals produced by DMA controller to exercise memory. The two signals I/O READ L and I/O WRITE L are bidirectional, they are inputs from the microprocessor when the microprocessor sends commands to the 8257 and reads back the 8257 status. During the I/O operation these signals are output from the 8257 and are functionally opposite to the memory signals. The 8257 takes control of the bus by exercising HALT (HRQ) and receives back the "go-ahead" signal on HALT ACKNOWLEDGE (HLDA).

Two signals produced by the DMA controller can be used by the I/O port to assist in controlling the transfer process. One signal TC--terminal count--is asserted during the last cycle of a DMA block. This can be used to describe a DMA mode on an I/O port or to reset the port's internal state to indicate the end of a transfer. The second--MARK--is inserted when the remaining count on a channel became a multiple of 128--providing a convenient timing signal for an external device.