Input/Output Systems

Processor needs to communicate with other devices:

• Receive signals from sensors

• Send commands to actuators

• Or both (e.g., disks, audio, video devices)

Input and output are accomplished through one of three different methods: programmed I/O, memory-mapped I/O, or direct memory address (DMA). Each method has advantages and disadvantages with respect to real-time performance, cost, and ease of implementation

1. Programmed Input/Output

In programmed I/O, special data-movement instructions are used to transfer data to and from the CPU. An IN instruction will transfer data from a specified I/O device into a specified CPU register. An OUT instruction will output from a register to some I/O device. Normally, the identity of the operative CPU register is embedded in the instruction code. Both the IN and OUT instructions require the efforts of the CPU, and thus cost time that could impact real-time performance.

For example, a computer system is used to control the speed of a motor. An output port is connected to the motor, and a signed integer is written to the port to set the motor speed. The computer is configured so that when an OUT instruction is executed, the contents of register 1 are placed on the data bus and sent to the I/O port at the address contained in register 2. The following code fragment allows the program to set the motor speed.

2. Direct Memory Access

In DMA, access to the computer's memory is given to other devices in the system without CPU intervention. That is, information is deposited directly into main memory by the external device. Here a DMA controller is required (Figure 2.11)

unless the DMA circuitry is integrated into the CPU. Because CPU participation is not required, data transfer is fast. The DMA controller prevents collisions by requiring each device to issue a DMA request signal (DMARQ) that will be acknowledged with a DMA acknowledge signal (DMACK). Until the DMACK signal is given to the requesting device, its connection to the main bus remains in a tristate condition.

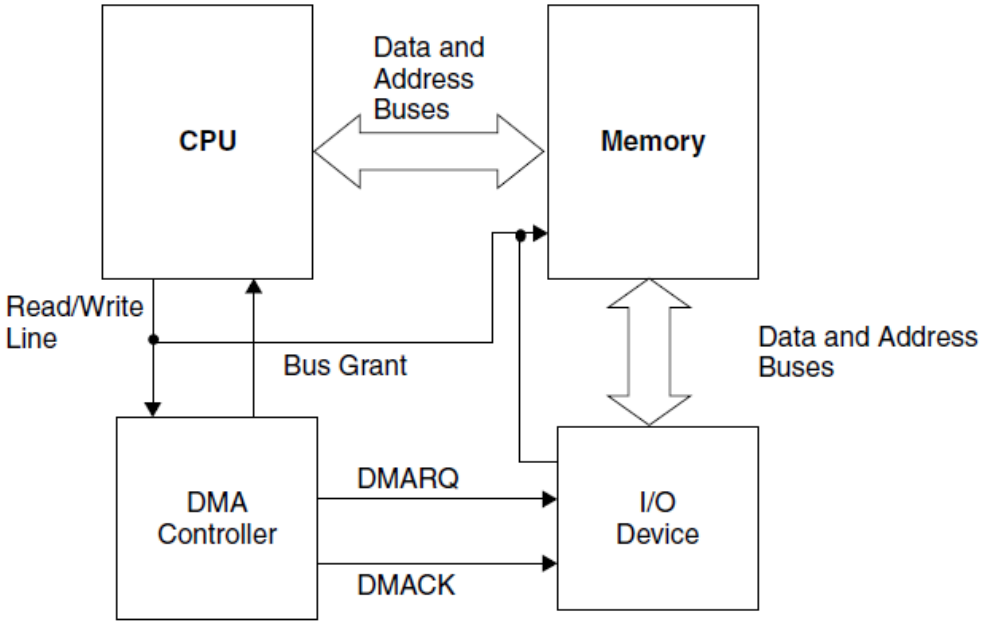

Figure 2.11

DMA circuitry where an external controller is used. This functionality can also be integrated on-chip with the CPU.

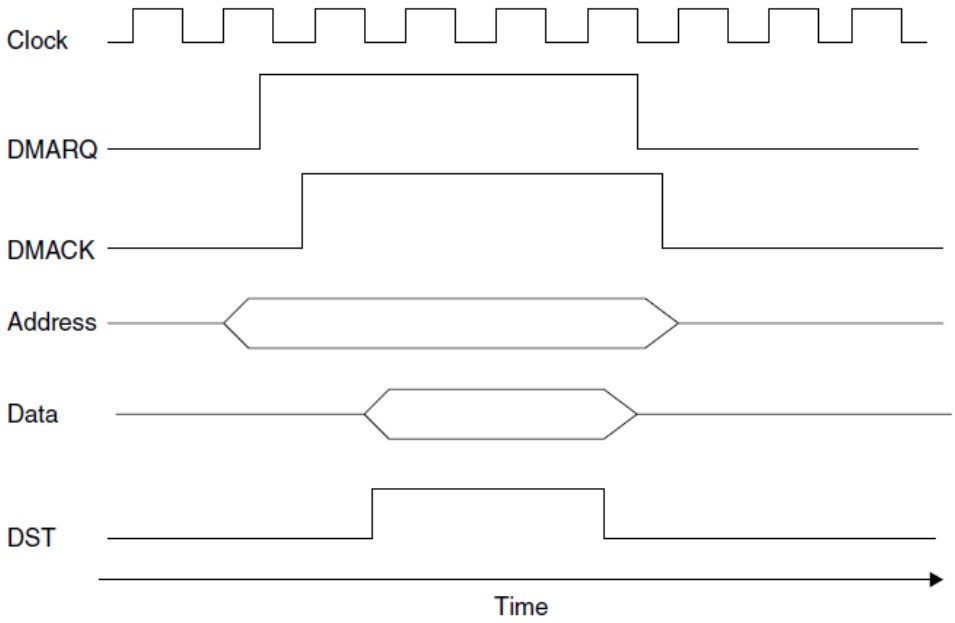

Figure 2.12 The DMA timing process. The sequence is: request transfer (DMARQ high), receive acknowledgment (DMACK high), place data on bus, and indicate data are present on bus (DST high). The signal height indicates voltage high/low.is given to the requesting device, its memory bus lines become active, and data transfer occurs, as with the CPU (Figure 2.12).

The CPU is prevented from performing a data transfer during DMA through the use of a signal called a bus grant. Until the bus grant signal is given by the controller, no other device can obtain the bus. The DMA controller is responsible for assuring that only one device can place data on the bus at any one time through bus arbitration. If two or more devices attempt to gain control of the bus simultaneously, bus contention occurs. When a device already has control of the bus and another obtains access, an undesirable occurrence (a collision) occurs.

The device requests control of the bus by signaling the controller via the DMARQ signal. Once the DMACK signal is asserted by the controller, the device can place (or access) data to/from the bus (which is indicated by another signal, typically denoted DST).

Without the bus grant (DMACK) from the DMA controller, the normal CPU data-transfer processes cannot proceed. At this point, the CPU can proceed with non-bus-related activities (e.g., the execution phase of an arithmetic instruction) until it receives the bus grant, or until it gives up (after some predetermined time) and issues a bus time-out signal.

Because of its speed, DMA is often the best method for input and output for real-time systems.

3. Memory-Mapped Input/Output

Memory-mapped I/O provides a data-transfer mechanism that is convenient because it does not require the use of special CPU I/O instructions. In memory-mapped I/O certain designated locations of memory appear as virtual I/O ports

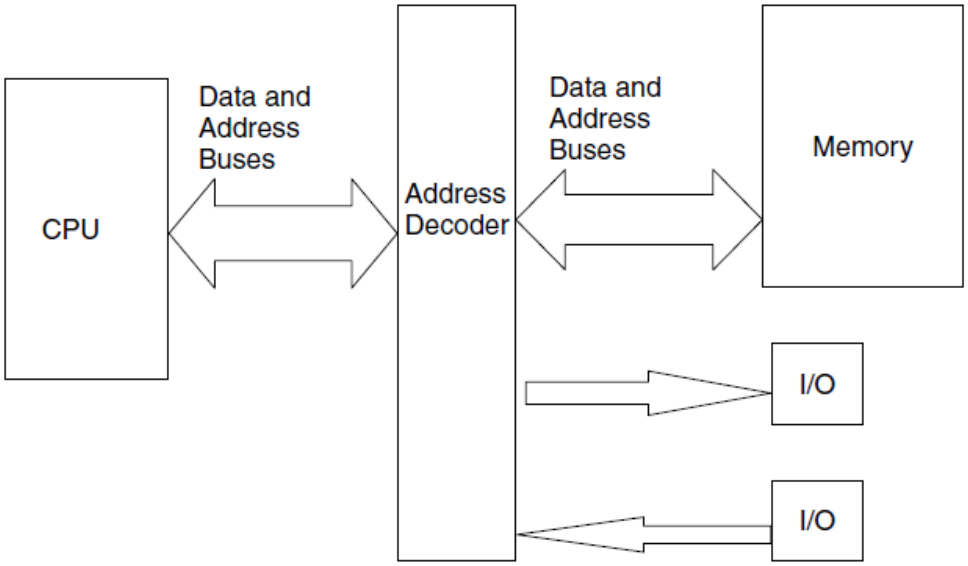

Figure 2.13 Memory-mapped I/O circuitry.

For example, consider the control of the speed of a stepping motor. If it were to be implemented via memory-mapped I/O, the required assembly language code might look like the following:

```
LOAD R1 &speed              ;motor speed into register 1
STORE R1 &motor address     ;store to address of motor control
```

where speed is a bit-mapped variable and motor address is a memory-mapped location.

In many computer systems, the video display is updated via memory-mapped I/O. For example, suppose that a display consists of a 24 row by 80 column array (a total of 1920 cells). Each screen cell is associated with a specific location in memory. To update the screen, characters are stored on the address assigned to cell on the screen.