19MCE304- DESIGN OF EMBEDDED SYSTEMS

CPU BUSES:

A computer system encompasses much more than the CPU;it also includes memory and I/O devices. The ***bus*** is the mechanism by which the CPU communicates with memory and devices. A bus is, at a minimum, a collection of wires, but the bus also defines a protocol by which the CPU, memory, and devices communicate. One of the major roles of the bus is to provide an interface to memory. (Of course, I/O devices also connect to the bus.)

1. Bus Protocols:

The basic building block of most bus protocols is the ***four-cycle handshake***, illustrated in Figure 2.1. The handshake ensures that when two devices want to communicate, one is ready to transmit and the other is ready to receive.

The handshake uses a pair of wires dedicated to the handshake: ***enq*** (meaning enquiry) and ***ack*** (meaning acknowledge). Extra wires are used for the data transmitted during the handshake. The four cycles are described below.

Device 1 raises its output to signal an enquiry, which tells *device 2* that it should get ready to listen for data.

When *device 2* is ready to receive, it raises its output to signal an acknowledgment. At this point, *devices 1* and *2* can transmit or receive.

Once the data transfer is complete, *device 2* lowers its output, signaling that it has received the data.

After seeing that *ack* has been released, *device 1* lowers its output.

At the end of the handshake, both handshaking signals are low, just as they were at the start of the handshake. The system has thus returned to its original state in readiness for another handshake-enabled data transfer.

Microprocessor buses build on the handshake for communication between the CPU and other system components. The term *bus* is used in two ways.

The most basic use is as a set of related wires, such as address wires. However, the term may also mean a protocol for communicating between components.

To avoid confusion, we will use the term ***bundle*** to refer to a set of related signals. The fundamental bus operations are reading and writing. Figure 2.2 shows the structure of a typical bus that supports reads and writes.

The major components follow:

Clock provides synchronization to the bus components,

R/W is true when the bus is reading and false when the bus is writing,

Address is an *a*-bit bundle of signals that transmits the address for an access,

Data is an *n*-bit bundle of signals that can carry data to or from the CPU, and

Data ready signals when the values on the data bundle are valid.

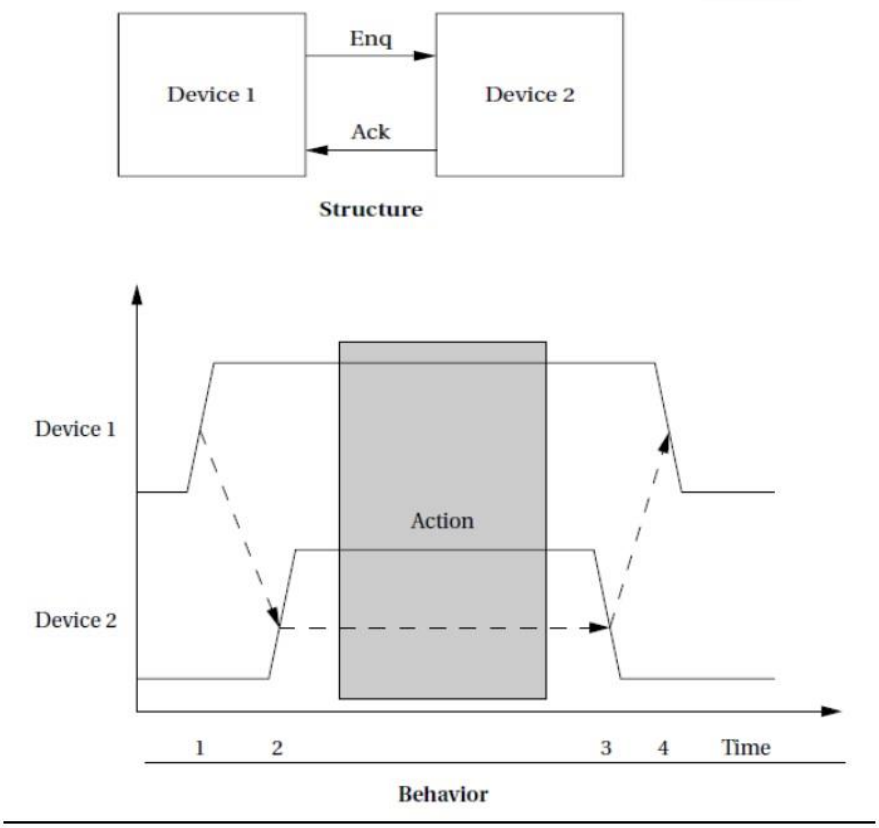

Fig 2.1 The four-cycle handshake.

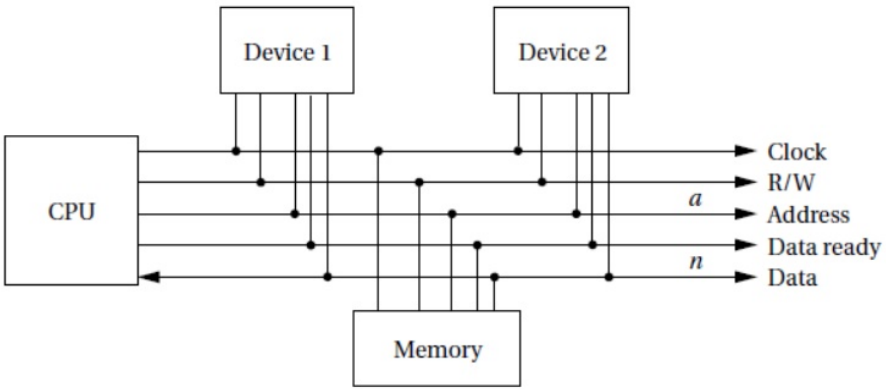

Fig 2.2 A typical microprocessor bus.

2. DMA:

Standard bus transactions require the CPU to be in the middle of every read and write transaction. However, there are certain types of data transfers in which the CPU does not need to be involved.

For example, a high-speed I/O device may want to transfer a block of data into memory. While it is possible to write a program that alternately reads the device and writes to memory, it would be faster to eliminate the CPU's involvement and let the device and memory communicate directly. This capability requires that some unit other than the CPU be able to control operations on the bus.

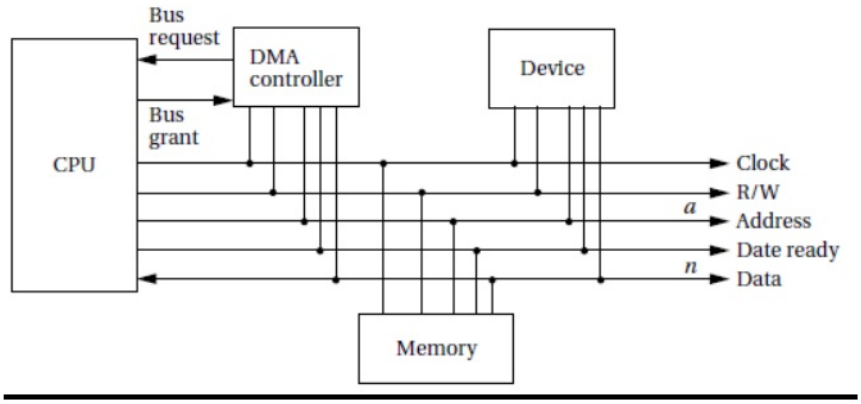

Fig 2.3 A bus with a DMA controller.

Direct memory access (DMA) is a bus operation that allows reads and writes not controlled by the CPU. A DMA transfer is controlled by a ***DMA controller***, which requests control of the bus from the CPU.

After gaining control, the DMA controller performs read and write operations directly between devices and memory. Figure 2.3 shows the configuration of a bus with a DMA controller. The DMA requires the CPU to provide two additional bus signals:

The ***bus request*** is an input to the CPU through which DMA controllers ask for ownership of the bus.

The ***bus grant*** signals that the bus has been granted to the DMA controller.

A device that can initiate its own bus transfer is known as a ***bus master***. Devices that do not have the capability to be ***bus masters*** do not need to connect to a bus request and bus grant.

The DMA controller uses these two signals to gain control of the bus using a classic four-cycle handshake. The bus request is asserted by the DMA controller when it wants to control the bus, and the bus grant is asserted by the CPU when the bus is ready.

The CPU will finish all pending bus transactions before granting control of the bus to the DMA controller. When it does grant control, it stops driving the other bus signals: R/W, address, and so on. Upon

becoming bus master, the DMA controller has control of all bus signals (except, of course, for bus request and bus grant).