One control line, usually labelled R/$\overline{W}$, specifies whether a Read or a Write operation is to be performed. As the label suggests, it specifies Read when set to 1 and Write when set to 0. When several data sizes are possible, such as byte, halfword, or word, the required size is indicated by other control lines. The bus control lines also carry timing information. They specify the times at which the processor and the I/O devices may place data on or receive data from the data lines. A variety of schemes have been devised for the timing of data transfers over a bus. These can be broadly classified as either synchronous or asynchronous schemes.

In any data transfer operation, one device plays the role of a *master*. This is the device that initiates data transfers by issuing Read or Write commands on the bus. Normally, the processor acts as the master, but other devices may also become masters as we will see in Section 7.3. The device addressed by the master is referred to as a *slave*.

7.2.1 SYNCHRONOUS BUS

On a *synchronous* bus, all devices derive timing information from a control line called the *bus clock*, shown at the top of Figure 7.3. The signal on this line has two *phases*: a high level followed by a low level. The two phases constitute a *clock cycle*. The first half of the cycle between the low-to-high and high-to-low transitions is often referred to as a clock pulse.

The address and data lines in Figure 7.3 are shown as if they are carrying both high and low signal levels at the same time. This is a common convention for indicating that

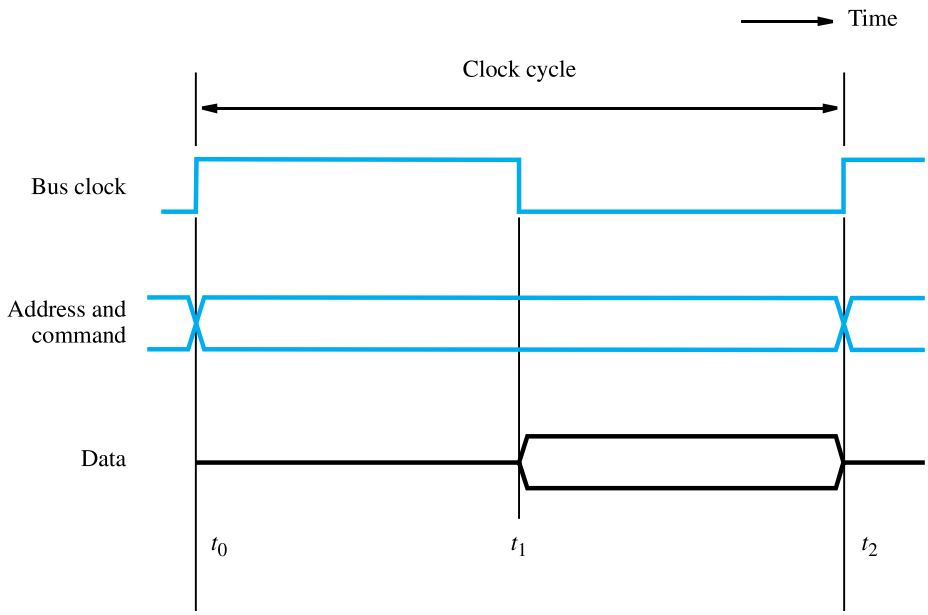

Figure 7.3 Timing of an input transfer on a synchronous bus.

some lines are high and some low, depending on the particular address or data values being transmitted. The crossing points indicate the times at which these patterns change. A signal line at a level half-way between the low and high signal levels indicates periods during which the signal is unreliable, and must be ignored by all devices.

Let us consider the sequence of signal events during an input (Read) operation. At time t_0, the master places the device address on the address lines and sends a command on the control lines indicating a Read operation. The command may also specify the length of the operand to be read. Information travels over the bus at a speed determined by its physical and electrical characteristics. The clock pulse width, $t_1 - t_0$, must be longer than the maximum propagation delay over the bus. Also, it must be long enough to allow all devices to decode the address and control signals, so that the addressed device (the slave) can respond at time t_1 by placing the requested input data on the data lines. At the end of the clock cycle, at time t_2, the master loads the data on the data lines into one of its registers. To be loaded correctly into a register, data must be available for a period greater than the setup time of the register (see Appendix A). Hence, the period $t_2 - t_1$ must be greater than the maximum propagation time on the bus plus the setup time of the master's register.

A similar procedure is followed for a Write operation. The master places the output data on the data lines when it transmits the address and command information. At time t_2, the addressed device loads the data into its data register.

The timing diagram in Figure 7.3 is an idealized representation of the actions that take place on the bus lines. The exact times at which signals change state are somewhat different from those shown, because of propagation delays on bus wires and in the circuits of the devices. Figure 7.4 gives a more realistic picture of what actually happens. It shows two views of each signal, except the clock. Because signals take time to travel from one device to another, a given signal transition is seen by different devices at different times. The top view shows the signals as seen by the master and the bottom view as seen by the slave. We assume that the clock changes are seen at the same time by all devices connected to the bus. System designers spend considerable effort to ensure that the clock signal satisfies this requirement.

The master sends the address and command signals on the rising edge of the clock at the beginning of the clock cycle (at t_0). However, these signals do not actually appear on the bus until t_{AM}, largely due to the delay in the electronic circuit output from the master to the bus lines. A short while later, at t_{AS}, the signals reach the slave. The slave decodes the address, and at t_1 sends the requested data. Here again, the data signals do not appear on the bus until t_{DS}. They travel toward the master and arrive at t_{DM}. At t_2, the master loads the data into its register. Hence the period $t_2 - t_{DM}$ must be greater than the setup time of that register. The data must continue to be valid after t_2 for a period equal to the hold time requirement of the register (see Appendix A for hold time).

Timing diagrams often show only the simplified picture in Figure 7.3, particularly when the intent is to give the basic idea of how data are transferred. But, actual signals will always involve delays as shown in Figure 7.4.

Multiple-Cycle Data Transfer

The scheme described above results in a simple design for the device interface. However, it has some limitations. Because a transfer has to be completed within one clock cycle,

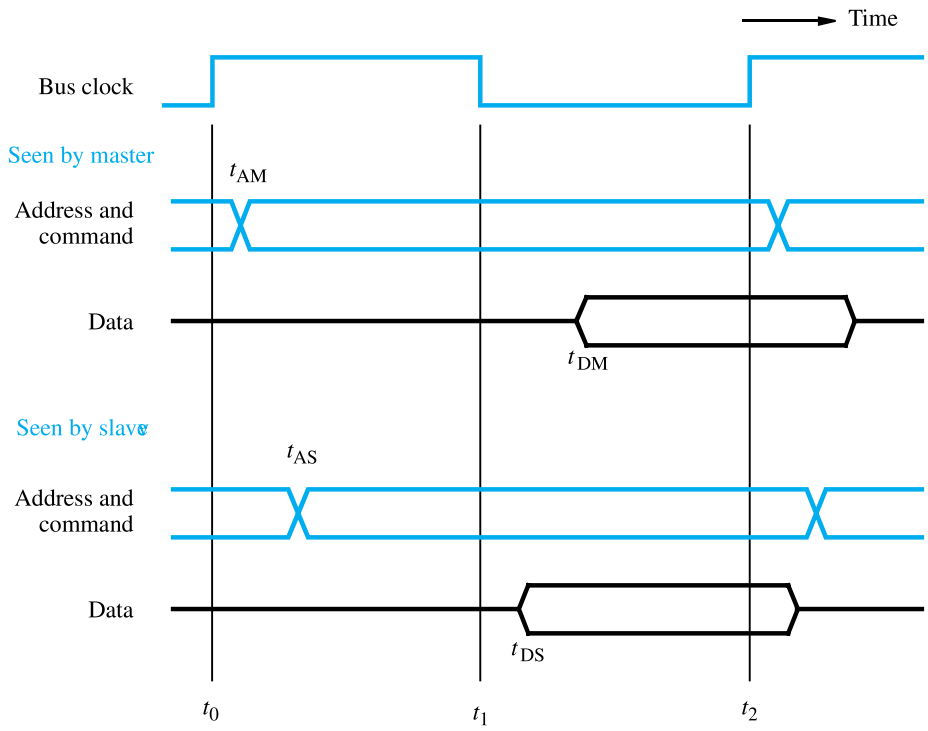

Figure 7.4 A detailed timing diagram for the input transfer of Figure 7.3.

the clock period, $t_2 - t_0$, must be chosen to accommodate the longest delays on the bus and the slowest device interface. This forces all devices to operate at the speed of the slowest device.

Also, the processor has no way of determining whether the addressed device has actually responded. At t_2, it simply assumes that the input data are available on the data lines in a Read operation, or that the output data have been received by the I/O device in a Write operation. If, because of a malfunction, a device does not operate correctly, the error will not be detected.

To overcome these limitations, most buses incorporate control signals that represent a response from the device. These signals inform the master that the slave has recognized its address and that it is ready to participate in a data transfer operation. They also make it possible to adjust the duration of the data transfer period to match the response speeds of different devices. This is often accomplished by allowing a complete data transfer operation to span several clock cycles. Then, the number of clock cycles involved can vary from one device to another.

An example of this approach is shown in Figure 7.5. During clock cycle 1, the master sends address and command information on the bus, requesting a Read operation. The slave receives this information and decodes it. It begins to access the requested data on the active

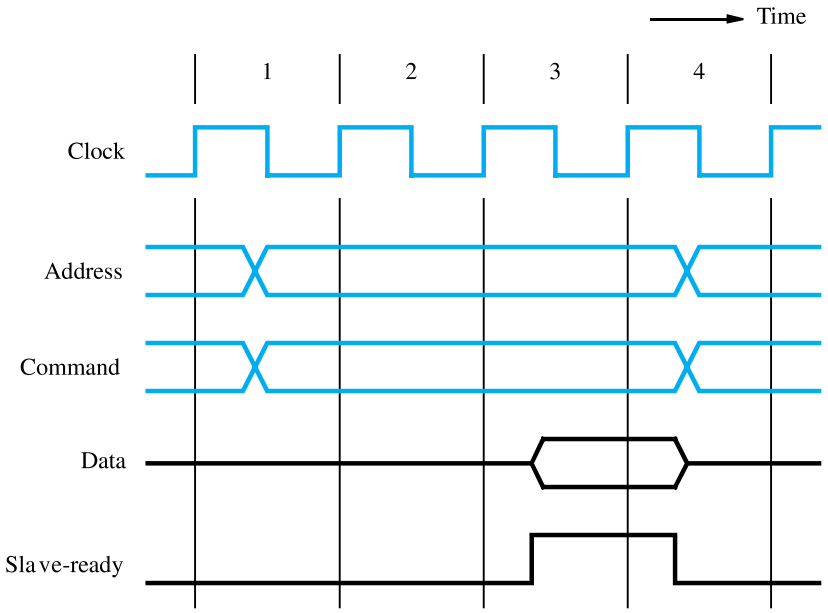

Figure 7.5 An input transfer using multiple clock cycles.

edge of the clock at the beginning of clock cycle 2. We have assumed that due to the delay involved in getting the data, the slave cannot respond immediately. The data become ready and are placed on the bus during clock cycle 3. The slave asserts a control signal called Slave-ready at the same time. The master, which has been waiting for this signal, loads the data into its register at the end of the clock cycle. The slave removes its data signals from the bus and returns its Slave-ready signal to the low level at the end of cycle 3. The bus transfer operation is now complete, and the master may send new address and command signals to start a new transfer in clock cycle 4.

The Slave-ready signal is an acknowledgment from the slave to the master, confirming that the requested data have been placed on the bus. It also allows the duration of a bus transfer to change from one device to another. In the example in Figure 7.5, the slave responds in cycle 3. A different device may respond in an earlier or a later cycle. If the addressed device does not respond at all, the master waits for some predefined maximum number of clock cycles, then aborts the operation. This could be the result of an incorrect address or a device malfunction.

We will now present a different approach that does not use a clock signal at all.

7.2.2 ASYNCHRONOUS BUS

An alternative scheme for controlling data transfers on a bus is based on the use of a *handshake* protocol between the master and the slave. A handshake is an exchange of

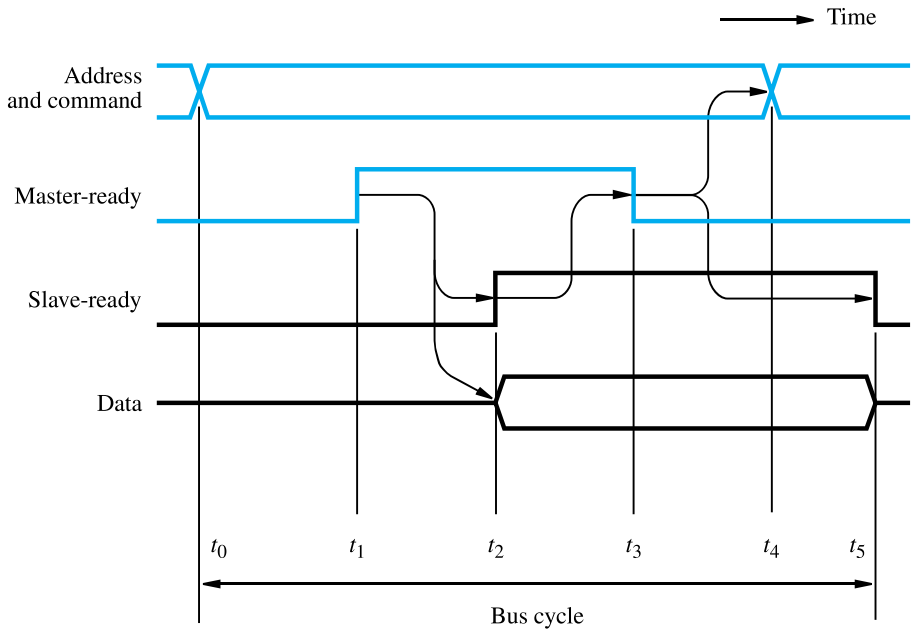

Figure 7.6 Handshake control of data transfer during an input operation.

command and response signals between the master and the slave. It is a generalization of the way the Slave-ready signal is used in Figure 7.5. A control line called Master-ready is asserted by the master to indicate that it is ready to start a data transfer. The Slave responds by asserting Slave-ready.

A data transfer controlled by a handshake protocol proceeds as follows. The master places the address and command information on the bus. Then it indicates to all devices that it has done so by activating the Master-ready line. This causes all devices to decode the address. The selected slave performs the required operation and informs the processor that it has done so by activating the Slave-ready line. The master waits for Slave-ready to become asserted before it removes its signals from the bus. In the case of a Read operation, it also loads the data into one of its registers.

An example of the timing of an input data transfer using the handshake protocol is given in Figure 7.6, which depicts the following sequence of events:

t_0—The master places the address and command information on the bus, and all devices on the bus decode this information.

t_1—The master sets the Master-ready line to 1 to inform the devices that the address and command information is ready. The delay $t_1 - t_0$ is intended to allow for any *skew* that may occur on the bus. Skew occurs when two signals transmitted simultaneously from one source arrive at the destination at different times. This happens because different lines of the bus may have different propagation speeds. Thus, to guarantee

that the Master-ready signal does not arrive at any device ahead of the address and command information, the delay $t_1 - t_0$ should be longer than the maximum possible bus skew. (Note that bus skew is a part of the maximum propagation delay in the synchronous case.) Sufficient time should be allowed for the device interface circuitry to decode the address. The delay needed can be included in the period $t_1 - t_0$.

t_2—The selected slave, having decoded the address and command information, performs the required input operation by placing its data on the data lines. At the same time, it sets the Slave-ready signal to 1. If extra delays are introduced by the interface circuitry before it places the data on the bus, the slave must delay the Slave-ready signal accordingly. The period $t_2 - t_1$ depends on the distance between the master and the slave and on the delays introduced by the slave's circuitry.

t_3—The Slave-ready signal arrives at the master, indicating that the input data are available on the bus. The master must allow for bus skew. It must also allow for the setup time needed by its register. After a delay equivalent to the maximum bus skew and the minimum setup time, the master loads the data into its register. Then, it drops the Master-ready signal, indicating that it has received the data.

t_4—The master removes the address and command information from the bus. The delay between t_3 and t_4 is again intended to allow for bus skew. Erroneous addressing may take place if the address, as seen by some device on the bus, starts to change while the Master-ready signal is still equal to 1.

t_5—When the device interface receives the 1-to-0 transition of the Master-ready signal, it removes the data and the Slave-ready signal from the bus. This completes the input transfer.

The timing for an output operation, illustrated in Figure 7.7, is essentially the same as for an input operation. In this case, the master places the output data on the data lines at the same time that it transmits the address and command information. The selected slave loads the data into its data register when it receives the Master-ready signal and indicates that it has done so by setting the Slave-ready signal to 1. The remainder of the cycle is similar to the input operation.

The handshake signals in Figures 7.6 and 7.7 are said to be *fully interlocked*, because a change in one signal is always in response to a change in the other. Hence, this scheme is known as a *full handshake*. It provides the highest degree of flexibility and reliability.

Discussion

Many variations of the bus protocols just described are found in commercial computers. The choice of a particular design involves trade-offs among factors such as:

- Simplicity of the device interface
- Ability to accommodate device interfaces that introduce different amounts of delay
- Total time required for a bus transfer
- Ability to detect errors resulting from addressing a nonexistent device or from an interface malfunction

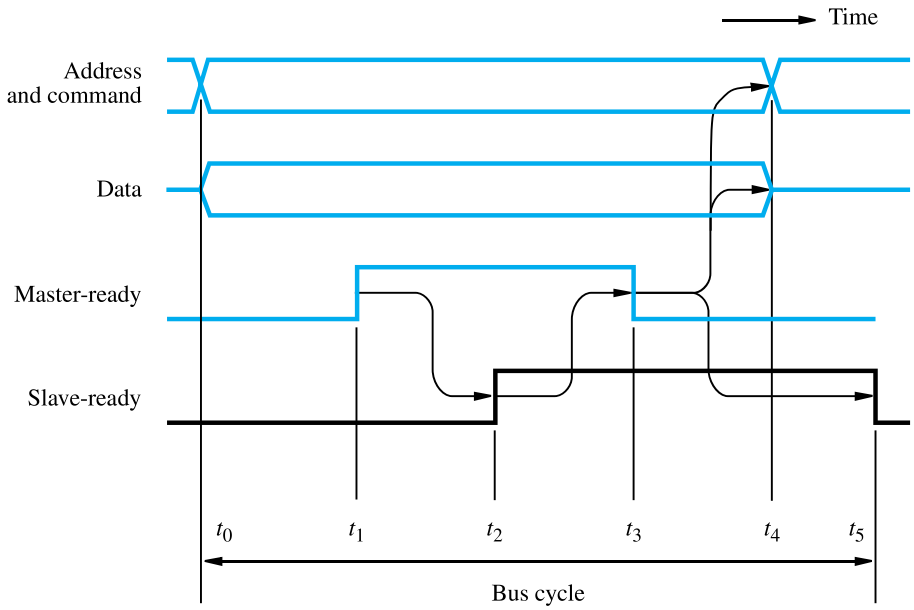

Figure 7.7 Handshake control of data transfer during an output operation.

The main advantage of the asynchronous bus is that the handshake protocol eliminates the need for distribution of a single clock signal whose edges should be seen by all devices at about the same time. This simplifies timing design. Delays, whether introduced by the interface circuits or by propagation over the bus wires, are readily accommodated. These delays are likely to differ from one device to another, but the timing of data transfers adjusts automatically. For a synchronous bus, clock circuitry must be designed carefully to ensure proper timing, and delays must be kept within strict bounds.

The rate of data transfer on an asynchronous bus controlled by the handshake protocol is limited by the fact that each transfer involves two round-trip delays (four end-to-end delays). This can be seen in Figures 7.6 and 7.7 as each transition on Slave-ready must wait for the arrival of a transition on Master-ready, and vice versa. On synchronous buses, the clock period need only accommodate one round trip delay. Hence, faster transfer rates can be achieved. To accommodate a slow device, additional clock cycles are used, as described above. Most of today's high-speed buses use the synchronous approach.

7.2.3 ELECTRICAL CONSIDERATIONS

A bus is an interconnection medium to which several devices may be connected. It is essential to ensure that only one device can place data on the bus at any given time. A logic gate that places data on the bus is called a *bus driver*. All devices connected to the bus, except the one that is currently sending data, must have their bus drivers turned off. A special type of logic gate, known as a *tri-state gate*, is used for this purpose. A tri-state gate