UNIT – 3

Chapter – 2

Input/Output Organization

Input/output Organization

1 Bus Structure

2 Bus Operation

    2.1 Synchronous Bus

    2.2 Asynchronous Bus

3 Arbitration

4 Interface Circuits

    4.1 Parallel Interface

    4.2 Serial Interface

5 Interconnection Standards

    5.1 PCI Bus

    5.2 SCSI Bus

Bus Structure

The bus is a simple structure that implements a **high-speed internal connection**. Buses are used to send control signals and data between the processor and other components.

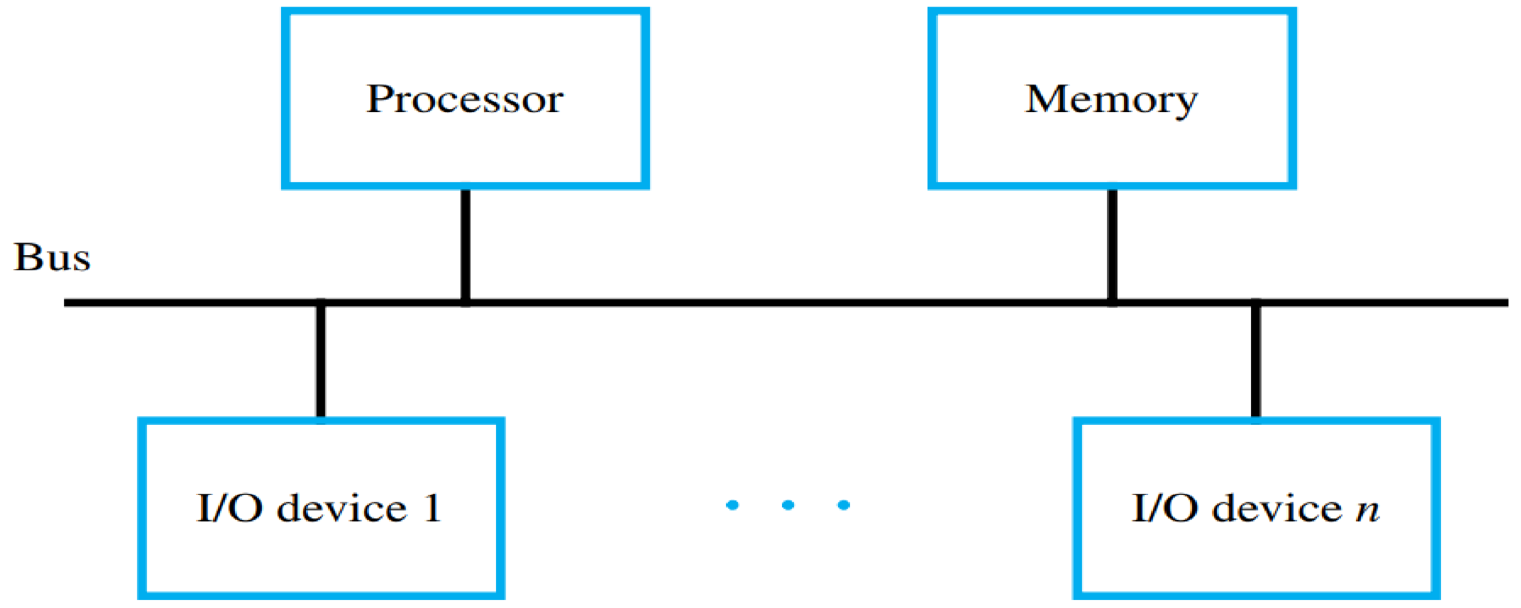

Figure 7.1 A single-bus structure.

The bus consists of three sets of lines used to carry address, data, and control signals.

I/O device interfaces are connected to these lines for an input device.

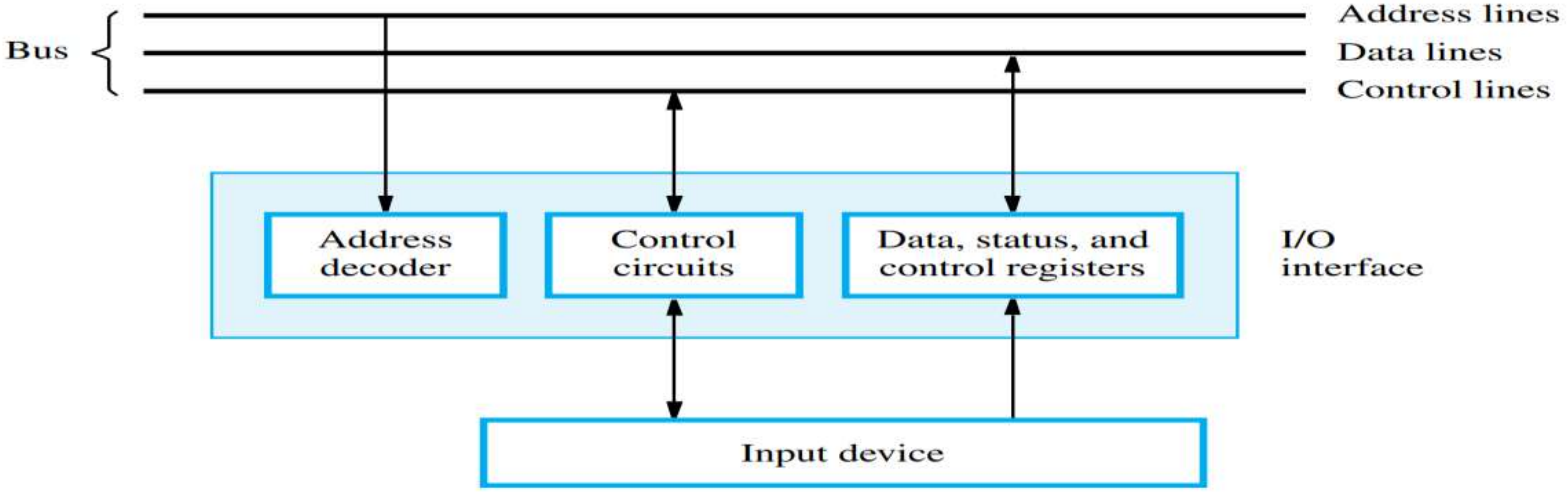

Figure 7.2 I/O interface for an input device.

Each I/O device is **assigned a unique set of addresses for the registers in its interface**. When the processor places a particular address on the address lines, it is examined by the address decoders of all devices on the bus.

The device that recognizes this address responds to the commands issued on the control lines. The processor uses the control lines to request either a Read or a Write operation, and the requested data are transferred over the data lines.

- When I/O devices and the memory share the same address space, the arrangement is called memory-mapped I/O.
- Any machine instruction that can access memory can be used to transfer data to or from an I/O device. For example, if the input device in Figure 7.2 is a keyboard and if DATAIN is its data register, the instruction
- Load R2, DATAIN
- reads the data from DATAIN and stores them into processor register R2. Similarly, the instruction
- Store R2, DATAOUT
- sends the contents of register R2 to location DATAOUT, which may be the data register of a display device interface.
- The status and control registers contain information relevant to the operation of the I/O device.
- The address decoder, the data and status registers, and the control circuitry required to coordinate I/O transfers constitute the device’s interface circuit.

Bus Operation

- A bus requires a **set of rules, often called a bus protocol**, that govern how the bus is used by various devices.
- The **bus protocol determines** when a device may place information on the bus, when it may load the data on the bus into one of its registers, and so on.
- These rules are implemented by control signals that indicate what and when actions are to be taken.
- 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.

The device addressed by the master is referred to as a slave.

SYNCHRONOUS BUS

The synchronous bus is a bus used to interconnect devices that comprise a computer system where the timing of transactions between devices is under the control of a synchronizing clock signal.

On a synchronous bus, All devices use timing information from a **common clock line**. i.e. timing information for all devices is generated by “Control Line” called **“ Bus Clock”.**

The signal on this line has two phases: a high level followed by a low level These 2 phases called **Clock Cycle.**[2 raising Edges 1 falling Edge] .

Clock Pulse: The first half of the cycle between the low-to-high and high-to-low transitions is often referred to as a clock pulse.

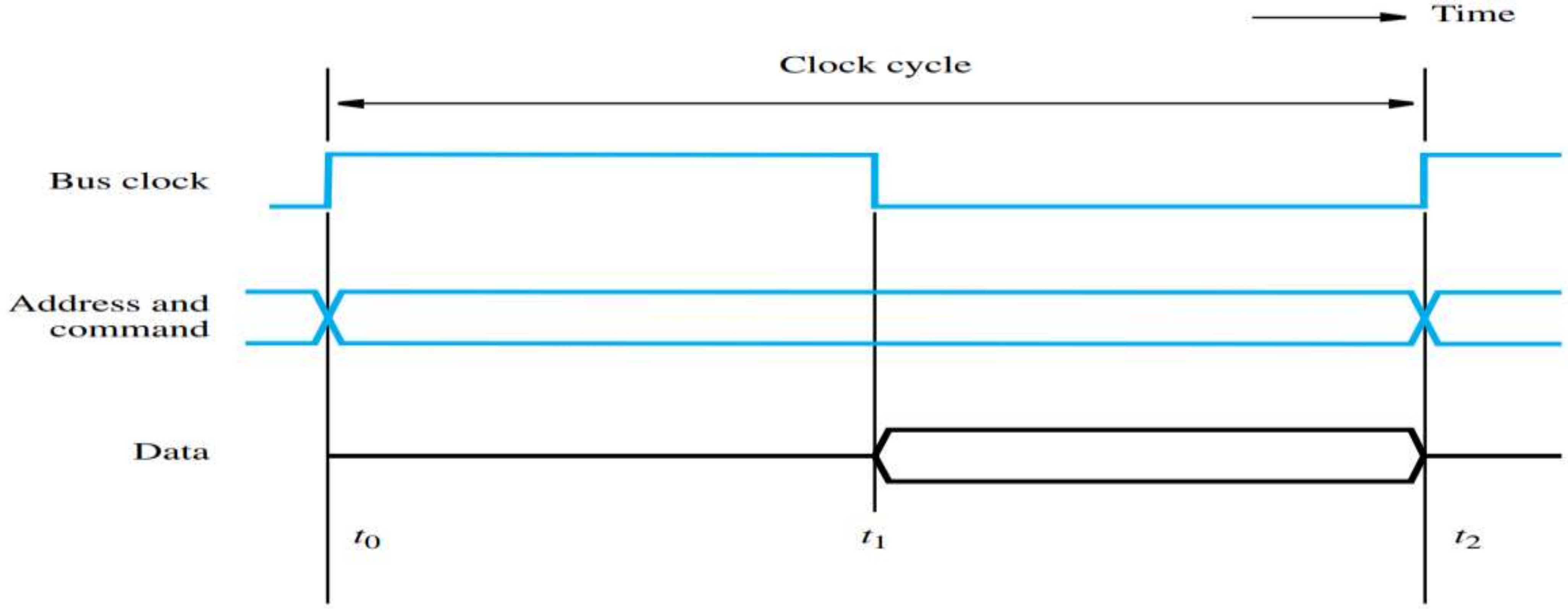

Figure 7.3 Timing of an input transfer on a synchronous bus.

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 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 events during an input (read) operation.
At t0 ,Master places the device address and command on the bus, and indicates that it is a Read operation.
At T1 Addressed slave respond by places data on the data lines.
Master "loads" the data on the data lines into one of its registers.

- Once the master places the device address and command on the bus, it takes time for this information to propagate to the devices:
  - This time depends on the physical and electrical characteristics of the bus.
- Also, all the devices have to be given enough time to decode the address and control signals, so that the addressed slave can place data on the bus at T1.
- Width of the pulse t_1 - t_0 depends on:
  - **Maximum propagation delay** between two devices connected to the bus.
  - Time taken by all the devices to decode the address and control signals, so that the addressed slave can respond at time t_1.
  - At the end of the clock cycle, at time t_2, the master loads the data on the data lines into one of its register.

- 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

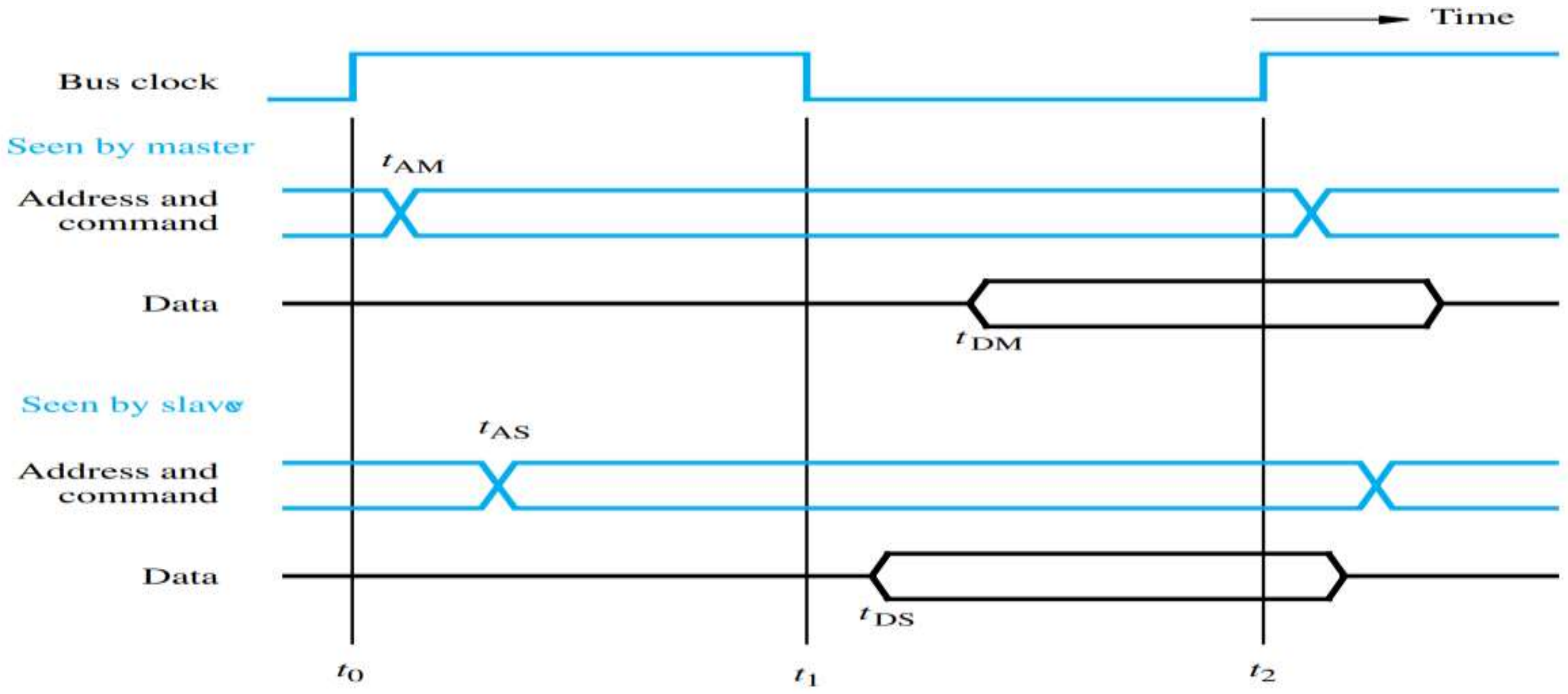

Figure 7.4 A detailed timing diagram for the input transfer of Figure 7.3.

- The master sends the address and command signals on the rising edge of the clock at the beginning of the clock cycle (at t0).

  However, these signals do not actually appear on the bus until tAM, largely due to the delay in the electronic circuit output from the master to the bus lines.

  A short while later, at tAS, the signals reach the slave.

- The slave decodes the address, and at t1 sends the requested data. Here again, the data signals do not appear on the bus until tDS.

  They travel toward the master and arrive at tDM. At t2, the master loads the data into its register.

  Hence the period t2 − tDM must be greater than the setup time of that register. The data must continue to be valid after t2 for a period equal to the hold time requirement of the register.

Multiple-Cycle Data Transfer

- Data transfer has to be completed within one clock cycle.
  - Clock period t2 - t0 must be such that the longest propagation delay on the bus and the slowest device interface must be accommodated.
  - Forces all the devices to operate at the speed of the slowest device.

- Processor just assumes that the data are available at t2 in case of a Read operation, or are read by the device in case of a Write operation.
  - What if the device is actually failed, and never really responded?
  - The CPU has no way to determine whether the addressed device has actually responded.
- Most buses have control signals to represent a response from the slave.
- Control signals serve two purposes:
  - Inform the master that the slave has recognized the address, and is ready to participate in a data transfer operation.
  - Enable to adjust the duration of the data transfer operation based on the speed of the participating slaves.
- High-frequency bus clock is used:
  - Data transfer spans several clock cycles instead of just one clock cycle as in the earlier case.

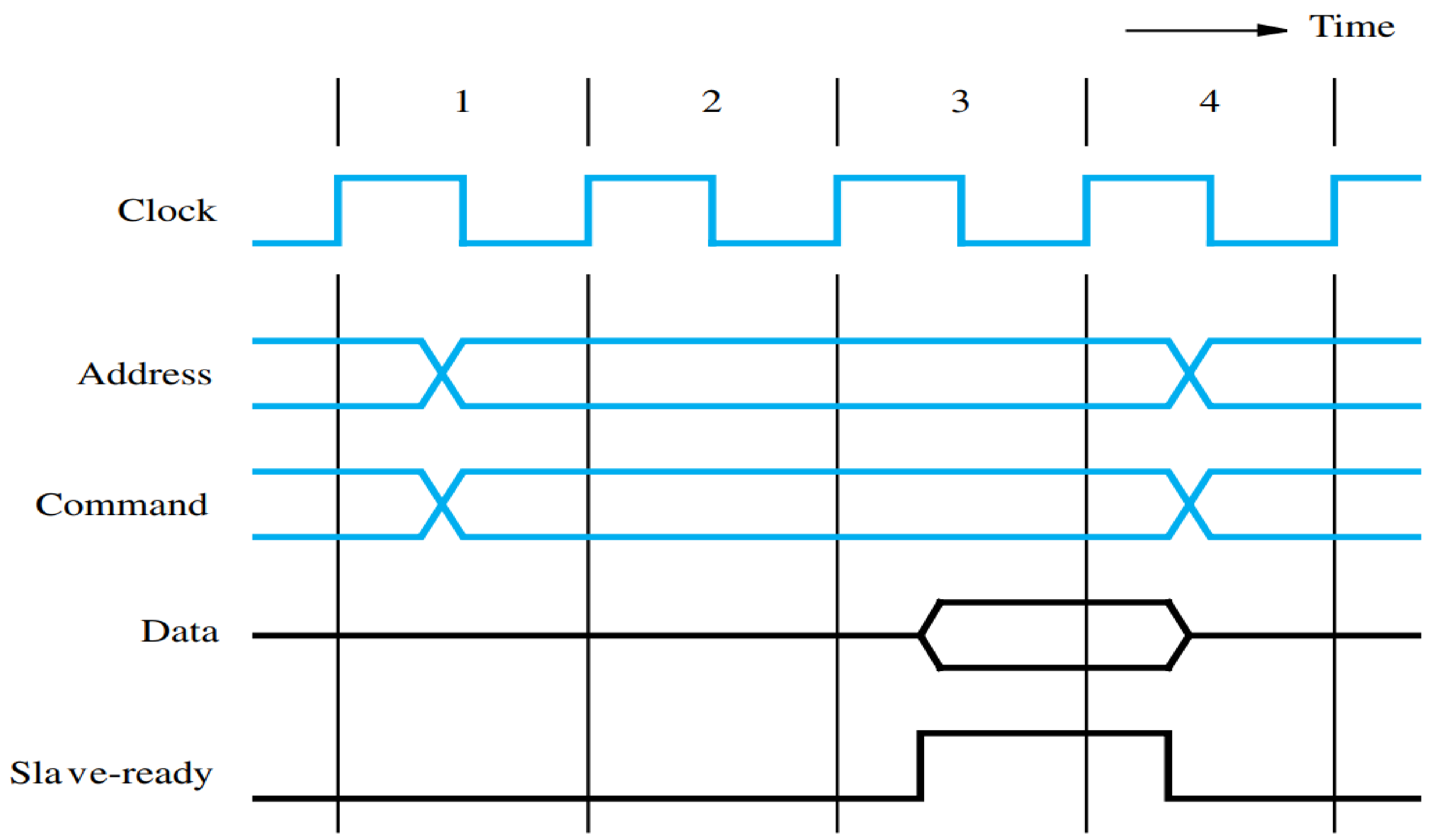

Figure 7.5 An input transfer using multiple clock cycles.

Asynchronous Bus

- Data transfers on the bus is controlled by a handshake between the master and the slave.
- A handshake is an exchange of command and response signals between the master and the slave
- **Common clock in the synchronous bus case is replaced by two timing control lines:**

  Master-ready,

  Slave-ready.
- Master-ready signal is asserted by the master to indicate to the slave that it is ready to participate in a data transfer.
- Slave-ready signal is asserted by the slave in response to the master-ready from the master, and it indicates to the master that the slave is ready to participate in a data transfer.
- 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.

Data transfer using the handshake protocol:

- Master places the address and command information on the bus.
- **Asserts the Master-ready signal to indicate to the slaves that the address and command information has been placed on the bus.**
- All devices on the bus decode the address.
- Address slave performs the required operation, and informs the processor it has done so by asserting the Slave-ready signal.
- Master removes all the signals from the bus, once Slave-ready is asserted.
- If the operation is a Read operation, Master also strobes the data into its input buffer.

Handshake control of data transfer during an operation.

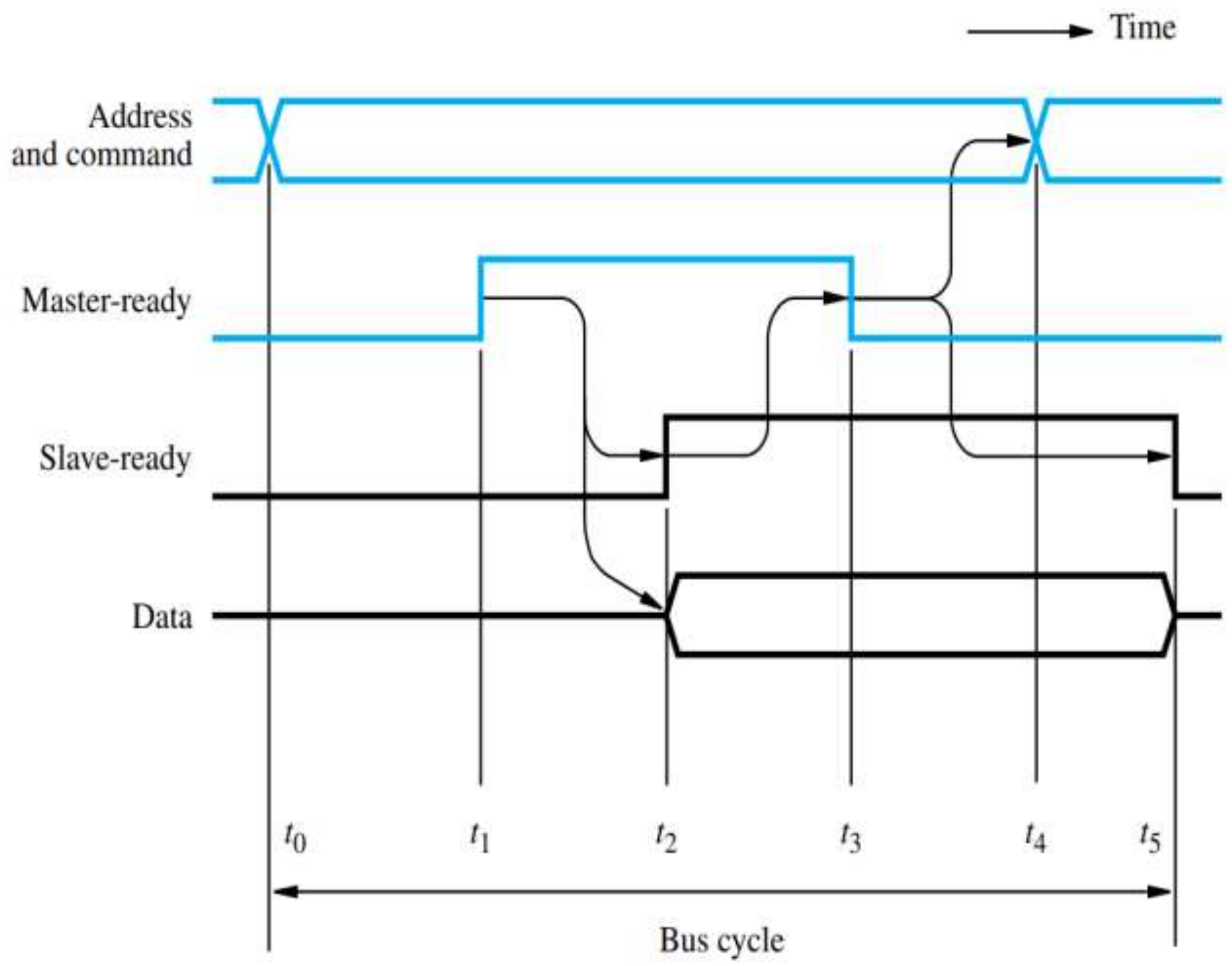

Figure 7.6 Handshake control of data transfer during an input operation.

t_0 - Master places the address and command information on the bus.
t_1 - Master asserts the Master-ready signal. Master-ready signal is asserted at t_1 instead of t_0
t_2 - Addressed slave places the data on the bus and asserts the Slave-ready signal.
t_3 - Slave-ready signal arrives at the master.
t_4 - Master removes the address and command information.
t_5 - Slave receives the transition of the Master-ready signal from 1 to 0. It removes the data and the Slave-ready signal from the bus.

Asynchronous vs. Synchronous bus

- Advantages of asynchronous bus:
  - Eliminates the need for synchronization between the sender and the receiver.
  - Can accommodate varying delays automatically, using the Slave-ready signal.
- Disadvantages of asynchronous bus:
  - Data transfer rate with full handshake is limited by two-round trip delays.
  - Data transfers using a synchronous bus involves only one round trip delay, and hence a synchronous bus can achieve faster rates.

Arbitration

The process of determining which competing bus master will be allowed access to the bus is called Bus Arbitration.

- **Multiple devices may need to use the bus at the same time so must have a way to arbitrate multiple requests. Arbitration allows more than one module to control the bus at one particular time.**
- The decision is usually made in an arbitration process performed by an arbiter circuit. The arbitration process starts by each device sending a request to use the shared resource. The arbiter associates priorities with individual requests. If it receives two requests at the same time, it grants the use of the slave to the device having the higher priority first.

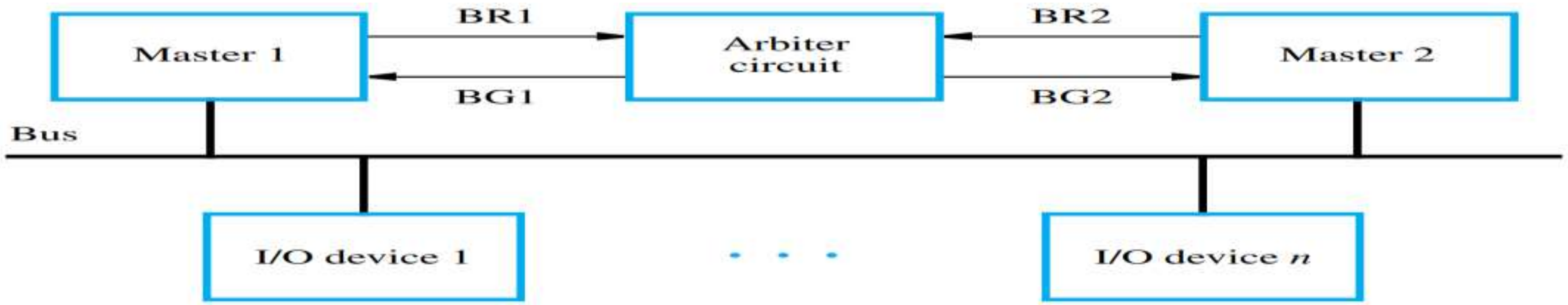

Figure 7.8 Bus arbitration.

Figure 7.8 illustrates an arrangement for bus arbitration involving two masters. There are two Bus-request lines, BR1 and BR2, and two Bus-grant lines, BG1 and BG2, connecting the arbiter to the masters.

A master requests use of the bus by activating its Bus-request line. If a single Bus-request is activated, the arbiter activates the corresponding Bus-grant.

This indicates to the selected master that it may now use the bus for transferring data. When the transfer is completed, that master deactivates its Bus-request, and the arbiter deactivates its Bus-grant.

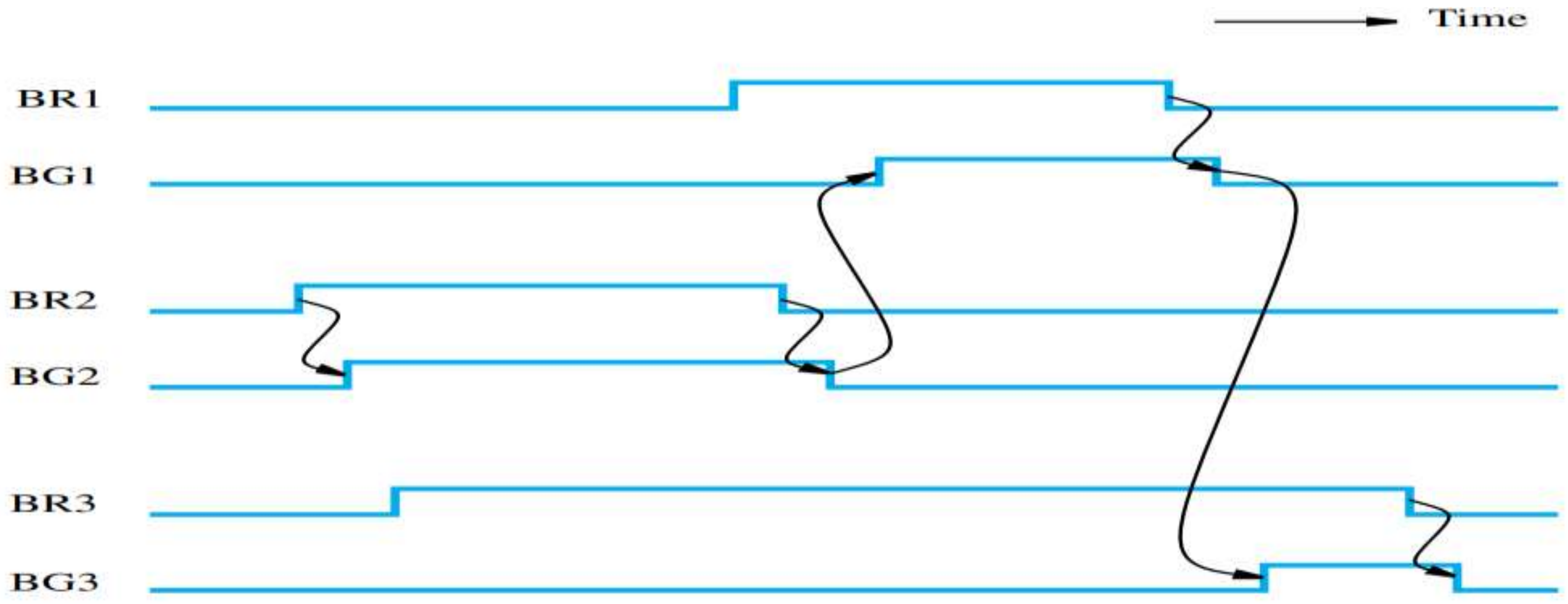

Figure 7.9 Granting use of the bus based on priorities.

Figure illustrates a possible sequence of events for the case of three masters.

Assume that master 1 has the highest priority, followed by the others in increasing numerical order.

Master 2 sends a request to use the bus first. Since there are no other requests, the arbiter grants the bus to this master by asserting BG2.

When master 2 completes its data transfer operation, it releases the bus by deactivating BR2. By that time, both masters 1 and 3 have activated their request lines.

Since device 1 has a higher priority, the arbiter activates BG1 after it deactivates BG2, thus granting the bus to master 1.

Later, when master 1 releases the bus by deactivating BR1, the arbiter deactivates BG1 and activates BG3 to grant the bus to master 3.

Note that the bus is granted to master 1 before master 3 even though master 3 activated its request line before master 1

Interface Circuits

The **interface circuit** is a mediator between the I/O device and the system to which this I/O has to be connected.

On one side of the interface are the bus lines for address, data, and control. On the other side are the connections needed to transfer data between the interface and the I/O device. This side is called a port, and it can be either a parallel or a serial port.

A parallel port transfers multiple bits of data simultaneously to or from the device.

A serial port sends and receives data one bit at a time

Features of the I/O interface circuit.

1. Provides a register for temporary storage of data
2. Includes a status register containing status information that can be accessed by the processor
3. Includes a control register that holds the information governing the behavior of the interface
4. Contains address-decoding circuitry to determine when it is being addressed by the processor
5. Generates the required timing signals
6. The interface circuit is also responsible for the format conversion that is essential for exchanging data between the processor and the I/O interface.

Parallel Interface(Parallel Port)

- To understand the interface circuit with a parallel port we will take the example of two I/O devices. i.e., a keyboard that has an 8-bit input port, and then an output device i.e., a display that has an 8-bit output port. Here multiple bits are transferred at once.

Input Port

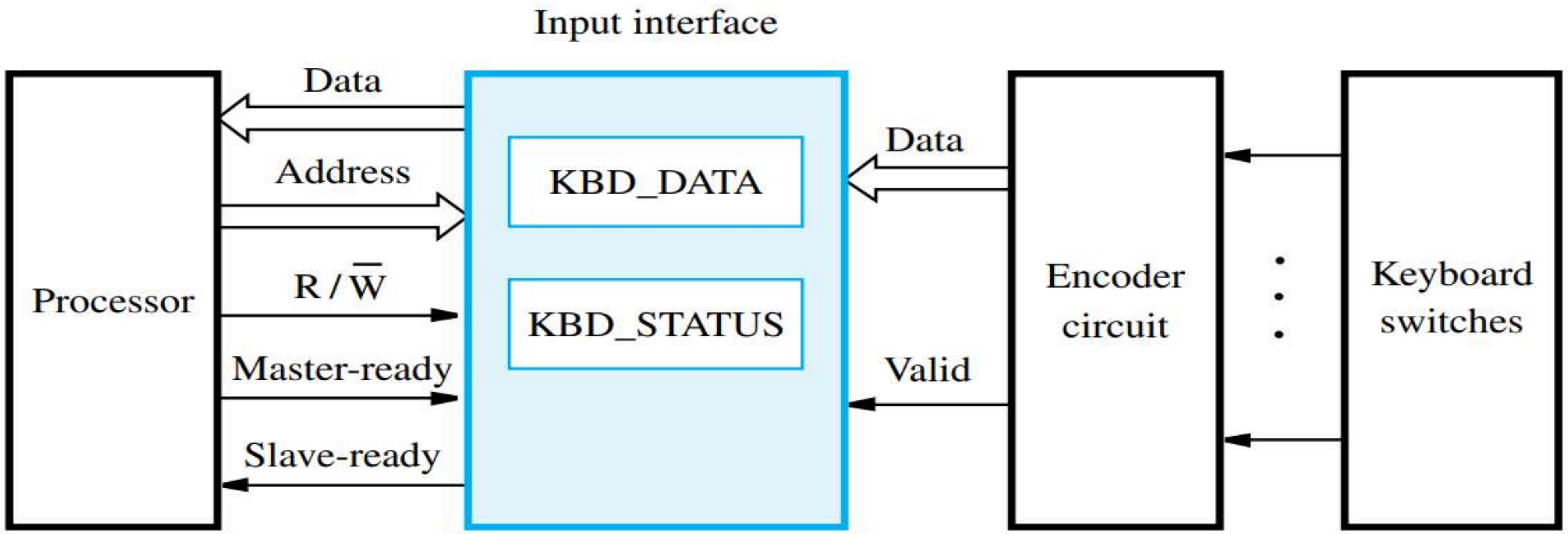

Figure 7.10 Keyboard to processor connection.

- Observe the parallel input port that connects the keyboard to the processor. Now, whenever the key is tapped on the keyboard an electrical connection is established that generates an electrical signal. This signal is encoded by the encoder to convert it into ASCII code for the corresponding character pressed at the keyboard.
- The encoder then outputs one byte of data that presents the character encoded by the encoder along with one valid bit. This valid bit changes its status from 0 to 1 when the key is pressed. So, when the valid bit is 1 the ASCII code of the corresponding character is loaded to the KBD_DATA register of the input interface circuit.
- Now, when the data is loaded into the KBD_DATA register the KIN status flag present in the KBD_STATUS register is set to1. Which causes the processor to read the data from KBD_DATA.
- Once the processor reads the data from KBD_DATA register the KIN flag is again set to 0. Here the input interface is connected to the processor using an asynchronous bus.
- So, the way they alert each other is using the master ready line and the slave ready line. Whenever the processor is ready to accept the data, it activates its master-ready line and whenever the interface is ready with the data to transmit it to the processor it activates its slave-ready line.
- The bus connecting processor and interface has one more control line i.e., R/W which is set to one for reading operation.

- Figure shows a possible circuit for the input interface. There are two addressable locations in this interface, KBD_DATA and KBD_STATUS.

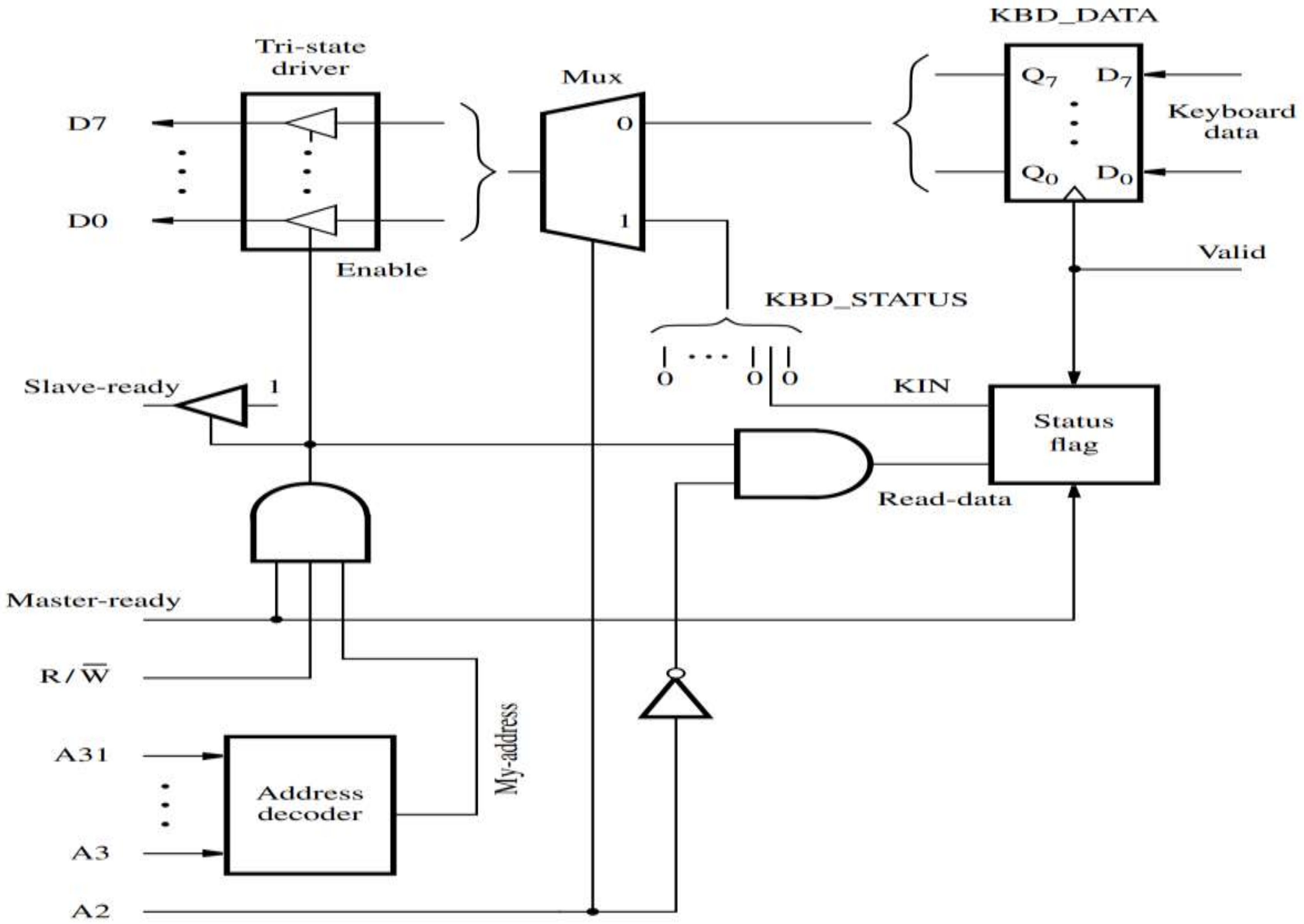

Figure 7.11 An input interface circuit.

Output Port

- The output interface shown in the figure below that connects the display and processor. The display device uses two handshake signals that are ready and new data and the other master and slave-ready.

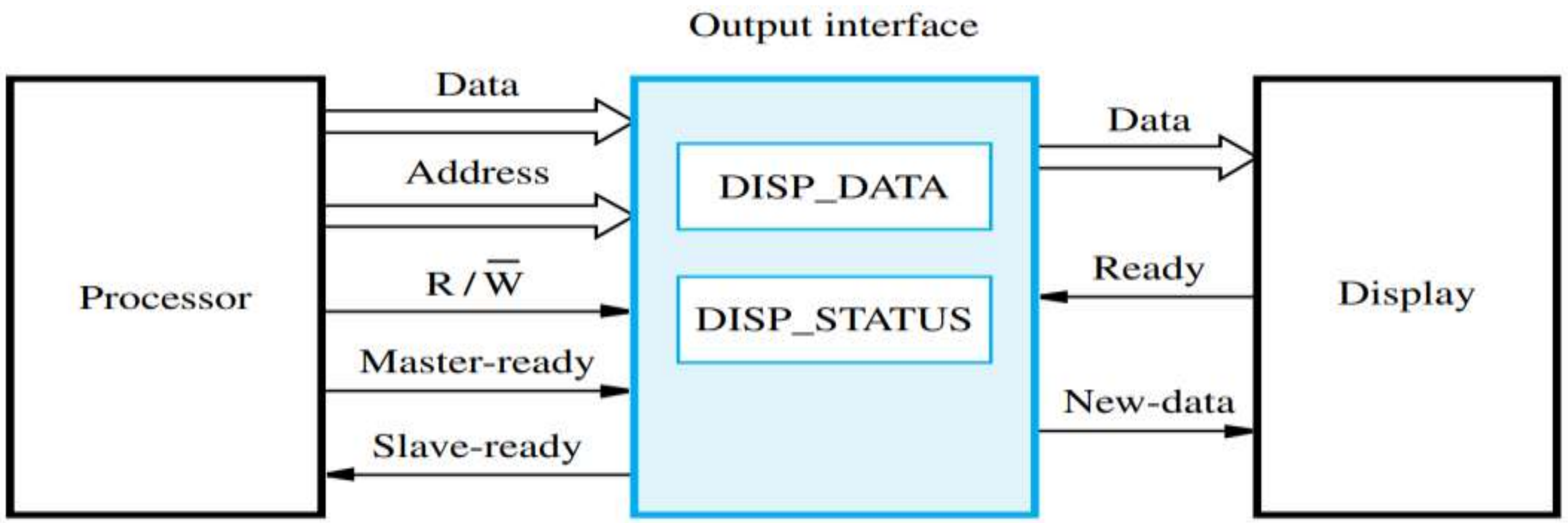

Figure 7.13 Display to processor connection.

- When the display unit is ready to display a character, it activates its ready line to 1 which setups the DOUT flag in the DISP_STATUS register to 1. This indicates the processor and the processor places the character to the DISP_DATA register.
- As soon as the processor loads the character in the DISP_DATA the DOUT flag setbacks to 0 and the New-data line to 1. Now as the display senses that the new-data line is activated it turns the ready line to 0 and accepts the character in the DISP_DATA register to display it.

Serial Interface(Serial Port)

- Opposite to the parallel port, **the serial port connects the processor to devices that transmit only one bit at a time.**
- Here on the device side, the data is transferred in the bit-serial pattern, and on the processor side, the data is transferred in the bit-parallel pattern.
- The transformation of the format from serial to parallel i.e., from device to processor, and from parallel to serial i.e., from processor to device is made possible with the help of shift registers (input shift register & output shift register).

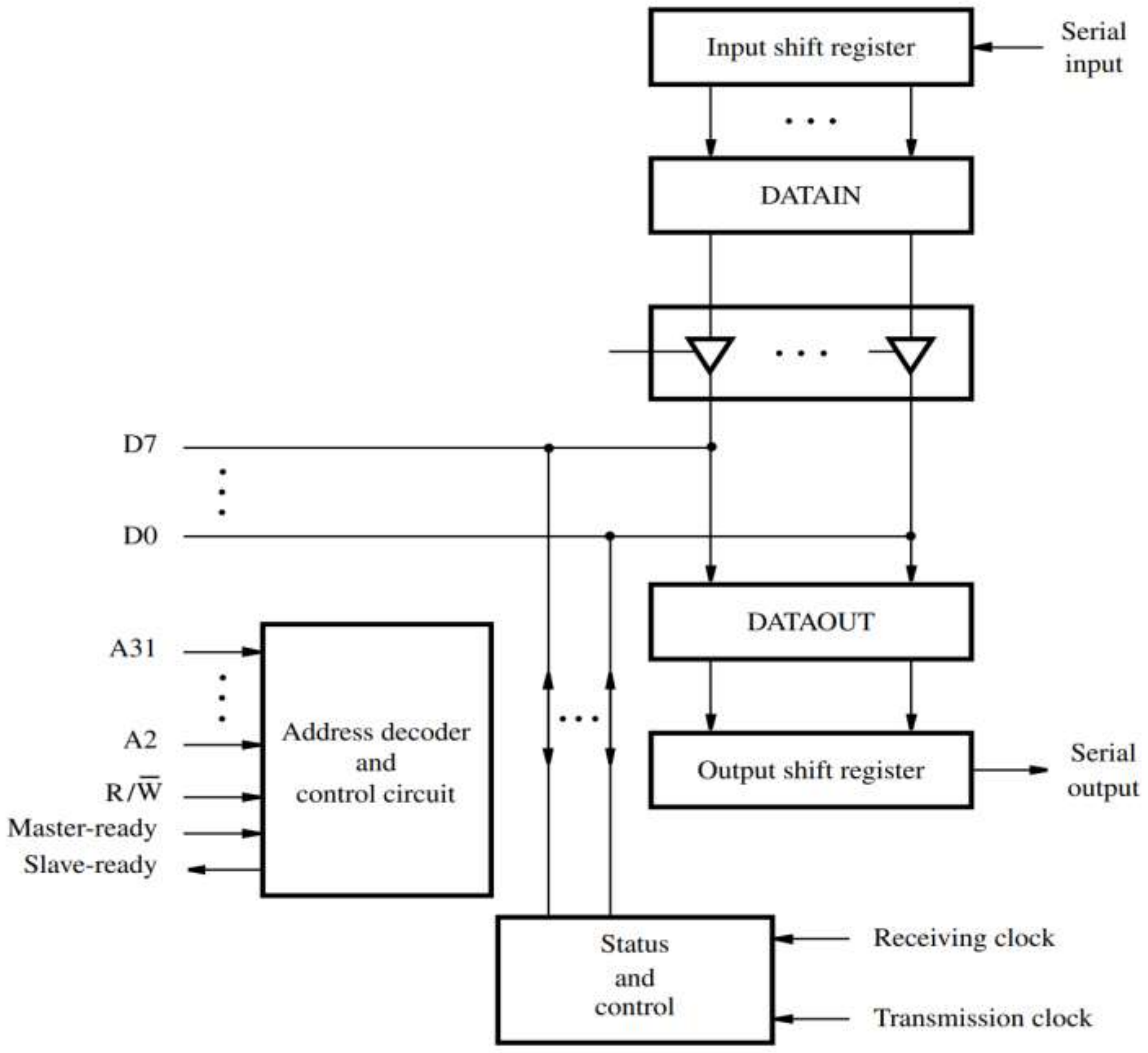

Figure 7.15 A serial interface.

- The above figure to understand the functioning of the serial interface at the device side. The input shift register accepts the one bit at a time in a bit-serial fashion till it receives all 8 bits. When all the 8 bits are received by the input shift register it loads its content into the DATA IN register parallelly. In a similar fashion, the content of the DATA OUT register is transferred in parallel to the output shift register.
- The serial interface port connected to the processor via system bus functions similarly to the parallel port. The status and control block has two status flags SIN and SOUT.
- The SIN flag is set to 1 when the I/O device inputs the data into the DATA IN register through the input shift register and the SIN flag is cleared to 0 when the processor reads the data from the DATA IN register.
- When the value of the SOUT register is 1 it indicates to the processor that the DATA OUT register is available to receive new data from the processor.
- The processor writes the data into the DATA OUT register and sets the SOUT flag to 0 and when the output shift register reads the data from the DATA OUT register sets back SOUT to 1.
- This makes the transmission convenient between the device that transmits and receives one bit at a time and the processor that transmits and receives multiple bits at a time.

Interconnection Standards

A typical desktop or notebook computer has several ports that can be used to connect I/O devices, such as a mouse, a memory key, or a disk drive.

Standard interfaces have been developed to enable I/O devices to use interfaces that are independent of any particular processor.

For example, a memory key that has a USB connector can be used with any computer that has a USB port.

Most standards are developed by a collaborative effort among a number of companies. In many cases, the IEEE (Institute of Electrical and Electronics Engineers) develops these standards further and publishes them as IEEE Standards.

PCI Bus

The PCI (Peripheral Component Interconnect) bus was developed as a low-cost, processor-independent bus. It is housed on the motherboard of a computer and used to connect I/O interfaces for a wide variety of devices.

A device connected to the PCI bus appears to the processor as if it is connected directly to the processor bus. Its interface registers are assigned addresses in the address space of the processor.

Bus Structure

The use of the PCI bus in a computer system is illustrated in Figure.

The PCI bus is connected to the processor bus via a controller called a bridge. The bridge has a special port for connecting the computer's main memory.

It may also have another special high speed port for connecting graphics devices. The bridge translates and relays commands and responses from one bus to the other and transfers data between them.

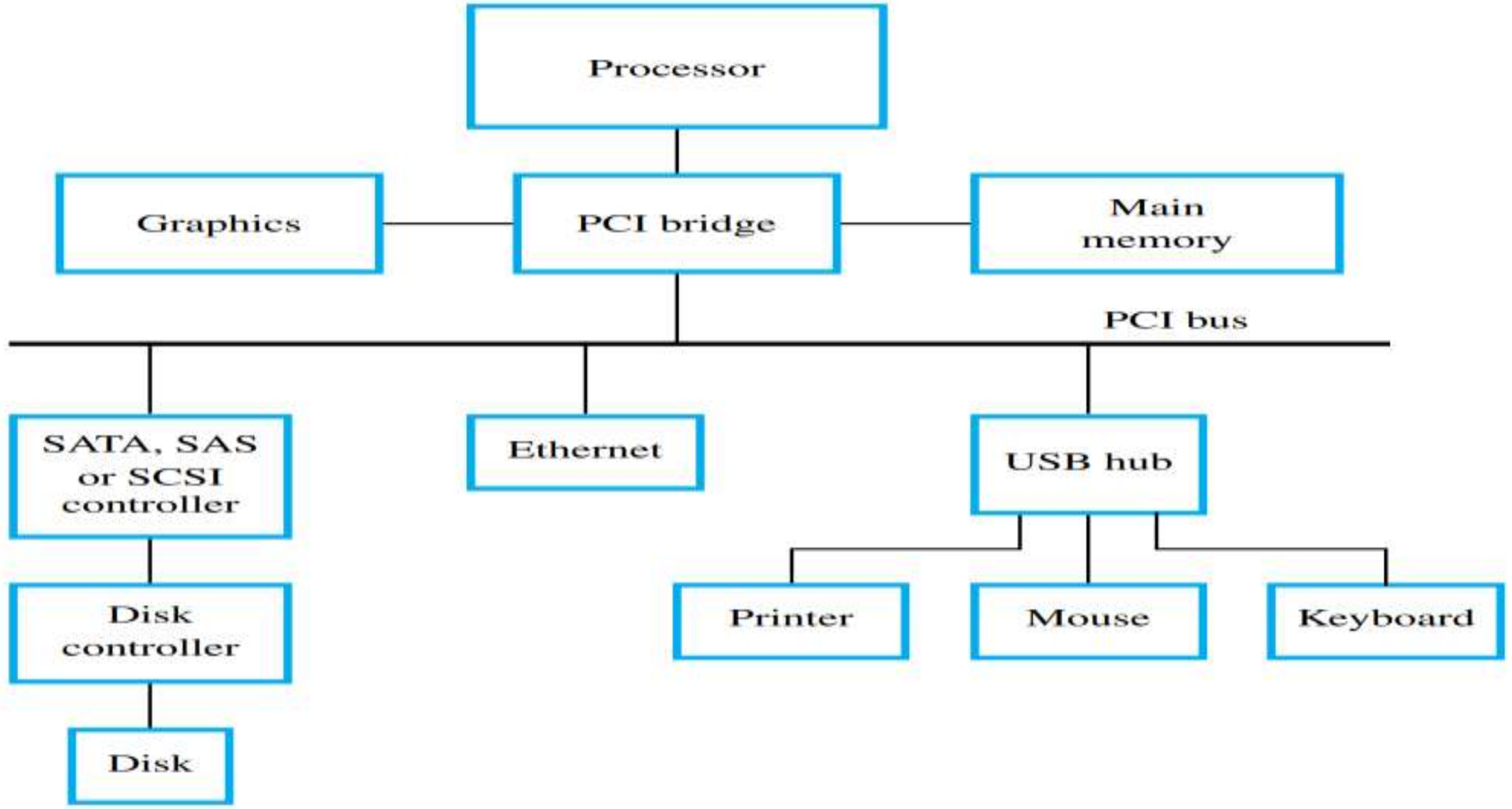

Figure 7.18 Use of a PCI bus in a computer system.

For example, when the processor sends a Read request to an I/O device, the bridge forwards the command and address to the PCI bus.

When the bridge receives the device’s response, it forwards the data to the processor using the processor bus.

I/O devices are connected to the PCI bus, possibly through ports that use standards such as Ethernet, USB, SCSI, or SAS.

The PCI bus supports three independent address spaces: memory, I/O, and configuration.

The system designer may choose to use memory-mapped I/O even with a processor that has a separate I/O address space.

In fact, this is the approach recommended by the PCI standard for wider compatibility. The configuration space is intended to give the PCI its plug-and-play capability.

Data Transfer

The bus master, which is the device that initiates data transfers by issuing Read and Write commands, is called the initiator in PCI terminology.

The addressed device that responds to these commands is called a target. The main bus signals used for transferring data are listed in below table

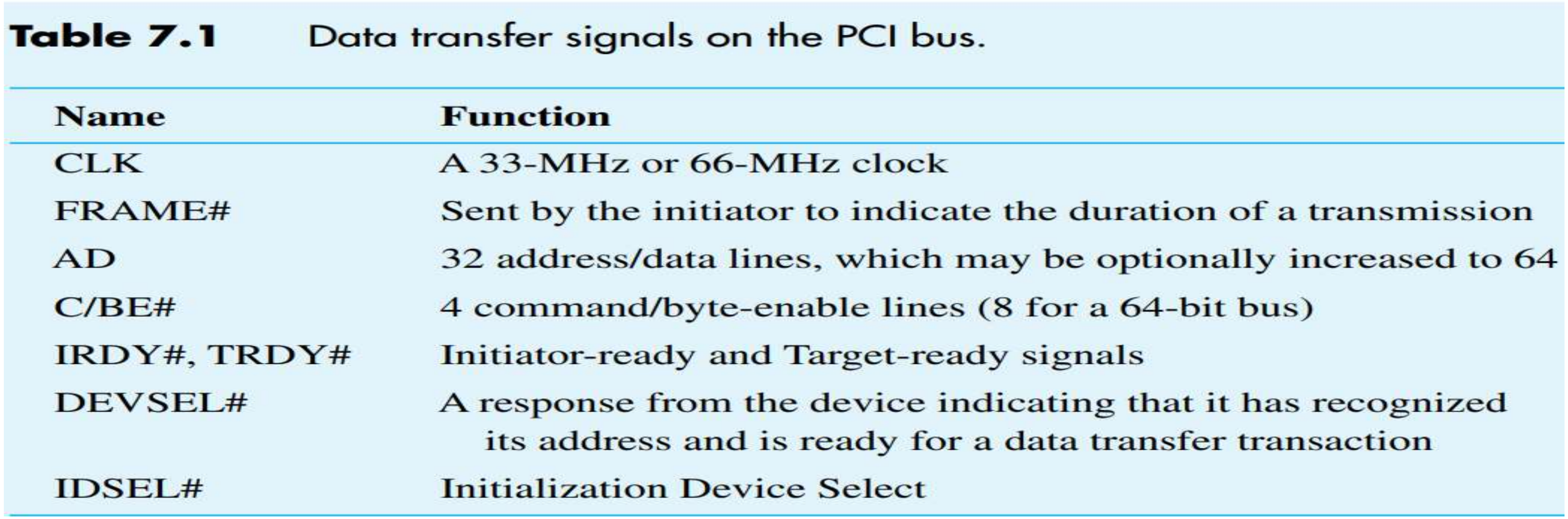

Table 7.1 Data transfer signals on the PCI bus.

Name	Function
CLK	A 33-MHz or 66-MHz clock
FRAME#	Sent by the initiator to indicate the duration of a transmission
AD	32 address/data lines, which may be optionally increased to 64
C/BE#	4 command/byte-enable lines (8 for a 64-bit bus)
IRDY#, TRDY#	Initiator-ready and Target-ready signals
DEVSEL#	A response from the device indicating that it has recognized its address and is ready for a data transfer transaction
IDSEL#	Initialization Device Select

There are 32 or 64 lines that carry address and data using a synchronous signaling scheme.

The target-ready, TRDY#, signal is equivalent to the Slave-ready signal in that figure. In addition, PCI uses an initiator-ready signal, IRDY#, to support burst transfers.

A signal whose name ends with the symbol # is asserted when in the low-voltage state.

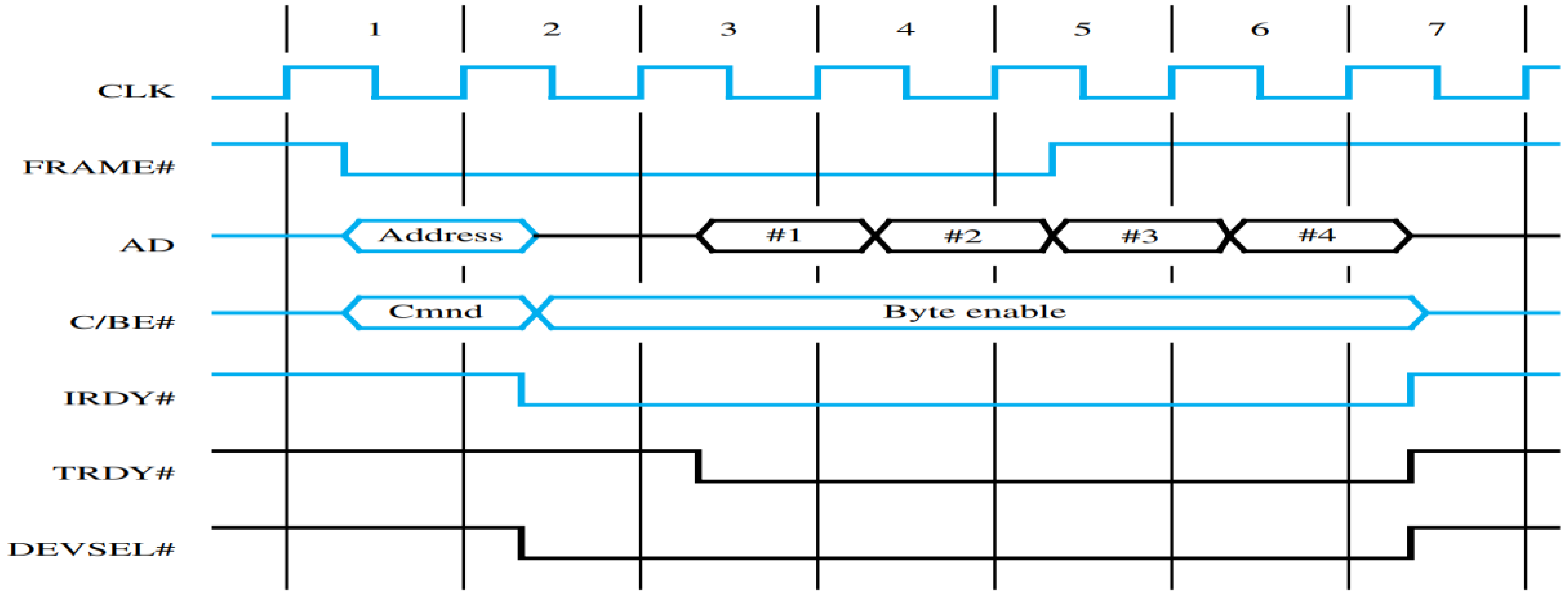

Figure 7.19 A Read operation on the PCI bus.

The bus master, acting as the initiator, asserts FRAME# in clock cycle 1 to indicate the beginning of a transaction. At the same time, it sends the address on the AD lines and a command on the C/BE# lines. In this case, the command will indicate that a Read operation is requested and that the memory address space is being used.

SCSI Bus

The acronym SCSI stands for Small Computer System Interface . It refers to a standard bus defined by the American National Standards Institute (ANSI).

The SCSI bus may be used to connect a variety of devices to a computer. It is particularly well-suited for use with disk drives. It is often found in installations such as institutional databases or email systems where many disks drives are used.

In the original specifications of the SCSI standard, devices are connected to a computer via a 50-wire cable, which can be up to 25 meters in length and can transfer data at rates of up to 5 Megabytes/s.

The standard has undergone many revisions, and its data transfer capability has increased rapidly. SCSI-2 and SCSI-3 have been defined, and each has several options.

Data are transferred either 8 bits or 16 bits in parallel, using clock speeds of up to 80 MHz. There are also several options for the electrical signaling scheme used.

The bus may use single-ended transmission, where each signal uses one wire, with a common ground return for all signals.

In another option, differential signaling is used, with a pair of wires for each signal.

Data Transfer

Devices connected to the SCSI bus are not part of the address space of the processor in the same way as devices connected to the processor bus or to the PCI bus.

A SCSI bus may be connected directly to the processor bus, or more likely to another standard I/O bus such as PCI, through a SCSI controller. Data and commands are transferred in the form of multi-byte messages called packets.

To send commands or data to a device, the processor assembles the information in the memory then instructs the SCSI controller to transfer it to the device.

Similarly, when data are read from a device, the controller transfers the data to the memory and then informs the processor by raising an interrupt.

The operations of the SCSI bus

let us consider how it may be used with a disk drive. Communication with a disk drive differs substantially from communication with the main memory.

Data are stored on a disk in blocks called sectors, where each sector may contain several hundred bytes.

1. The SCSI controller contends for control of the SCSI bus.

2. When it wins the arbitration process, the SCSI controller sends a command to the disk controller, specifying the required Read operation.

3. The disk controller cannot start to transfer data immediately. It must first move the read head of the disk to the required sector. Hence, it sends a message to the SCSI controller indicating that it will temporarily suspend the connection between them. The SCSI bus is now free to be used by other devices.

4. The disk controller sends a command to the disk drive to move the read head to the first sector involved in the requested Read operation. It reads the data stored in that sector and stores them in a data buffer.

When it is ready to begin transferring data, it requests control of the bus. After it wins arbitration, it re-establishes the connection with the SCSI controller, sends the contents of the data buffer, then suspends the connection again.

5. The process is repeated to read and transfer the contents of the second disk sector.

6. The SCSI controller transfers the requested data to the main memory and sends an interrupt to the processor indicating that the data are now available.