Lecture 6

Lecture Outlines

Learning Objectives

After studying this lecture, you should be able to:

- Understand the basic elements of an instruction cycle and the role of interrupts.
- Describe the concept of interconnection within a computer system.
- Assess the relative advantages of point-to-point interconnection compared to bus interconnection.

3.3 INTERCONNECTION STRUCTURES

A computer consists of a set of components or modules of three basic types (processor, memory, I/O) that communicate with each other. In effect, a computer is a network of basic modules. Thus, there must be paths for connecting the modules.

The collection of paths connecting the various modules is called the *interconnection structure*. The design of this structure will depend on the exchanges that must be made among modules.

Figure 3.15 suggests the types of exchanges that are needed by indicating the major forms of input and output for each module type:

- **Memory:** Typically, a memory module will consist of N words of equal length. Each word is assigned a unique numerical address (0, 1, . . . , $N-1$). A word of data can be read from or written into the memory. The nature of the operation is indicated by read and write control signals. The location for the operation is specified by an address.
- **I/O module:** From an internal (to the computer system) point of view, I/O is functionally similar to memory. There are two operations; read and write. Further, an I/O module may control more than one external device. We can refer to each of the interfaces to an external device as a *port* and give each a unique address (e.g., 0, 1, c , M-1). In addition, there are external data paths for the input and output of data with an external device. Finally, an I/O module may be able to send interrupt signals to the processor.

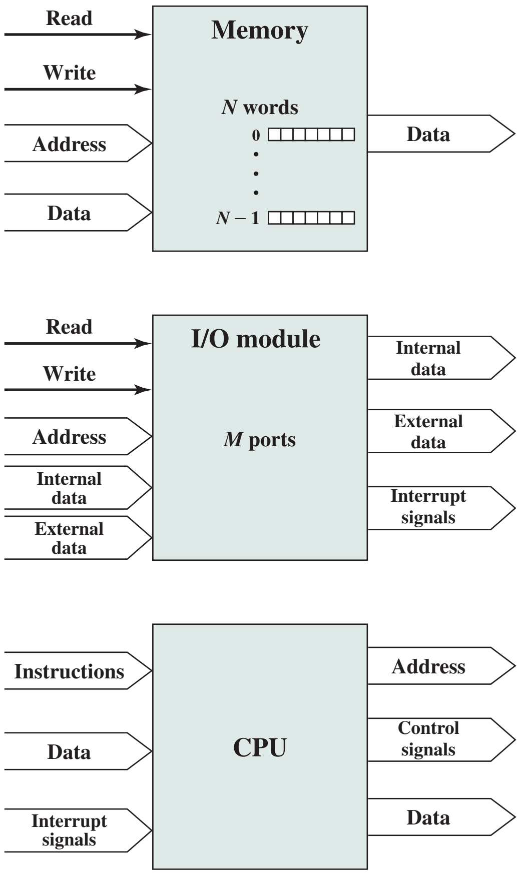

Figure 3.15 Computer Modules

- **Processor:** The processor reads in instructions and data, writes out data after processing, and uses control signals to control the overall operation of the system. It also receives interrupt signals.

The preceding list defines the data to be exchanged. The interconnection structure must support the following types of transfers:

- **Memory to processor:** The processor reads an instruction or a unit of data from memory.
- **Processor to memory:** The processor writes a unit of data to memory.
- **I/O to processor:** The processor reads data from an I/O device via an I/O module.
- **Processor to I/O:** The processor sends data to the I/O device.
- **I/O to or from memory:** For these two cases, an I/O module is allowed to exchange data directly with memory, without going through the processor, using direct memory access.

3.4 BUS INTERCONNECTION

The bus was the dominant means of computer system component interconnection for decades. For general-purpose computers, it has gradually given way to various point-to-point interconnection structures, which now dominate computer system design. However, bus structures are still commonly used for embedded systems, particularly microcontrollers. In this section, we give a brief overview of bus structure. Appendix C provides more detail.

A bus is a communication pathway connecting two or more devices. A key characteristic of a bus is that it is a shared transmission medium. Multiple devices connect to the bus, and a signal transmitted by any one device is available for reception by all other devices attached to the bus. If two devices transmit during the same time period, their signals will overlap and become garbled. Thus, only one device at a time can successfully transmit.

Typically, a bus consists of multiple communication pathways, or lines. Each line is capable of transmitting signals representing binary 1 and binary 0. Over time, a sequence of binary digits can be transmitted across a single line. Taken together, several lines of a bus can be used to transmit binary digits simultaneously (in parallel). For example, an 8-bit unit of data can be transmitted over eight bus lines.

Computer systems contain a number of different buses that provide pathways between components at various levels of the computer system hierarchy. A bus that connects major computer components (processor, memory, I/O) is called a **system bus**. The most common computer interconnection structures are based on the use of one or more system buses.

A system bus consists, typically, of from about fifty to hundreds of separate lines. Each line is assigned a particular meaning or function. Although there are many different bus designs, on any bus the lines can be classified into three functional groups (Figure 3.16): data, address, and control lines. In addition, there may be power distribution lines that supply power to the attached modules.

The **data lines** provide a path for moving data among system modules. These lines, collectively, are called the **data bus**. The data bus may consist of 32, 64, 128, or even more separate lines, the number of lines being referred to as the *width* of the data bus. Because each line can carry only one bit at a time, the number of lines determines how many bits can be transferred at a time. The width of the data bus is a key factor in determining overall system performance. For example, if the

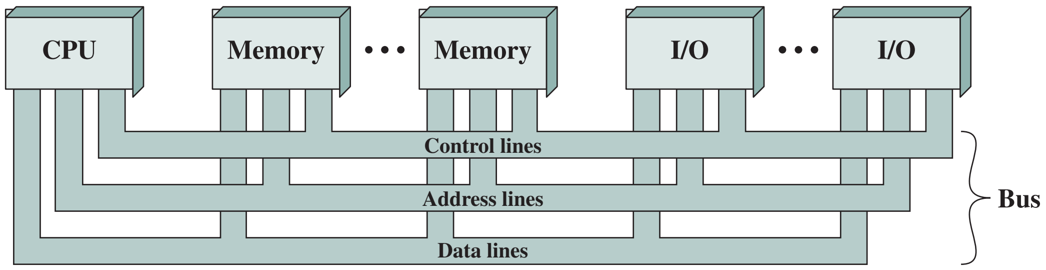

Figure 3.16 Bus Interconnection Scheme

data bus is 32 bits wide and each instruction is 64 bits long, then the processor must access the memory module twice during each instruction cycle.

The **address lines** are used to designate the source or destination of the data on the data bus. For example, if the processor wishes to read a word (8, 16, or 32 bits) of data from memory, it puts the address of the desired word on the address lines. Clearly, the width of the **address bus** determines the maximum possible memory capac-ity of the system. Furthermore, the address lines are generally also used to address I/O ports. Typically, the higher-order bits are used to select a particular module on the bus, and the lower-order bits select a memory location or I/O port within the module. For example, on an 8-bit address bus, address 01111111 and below might reference locations in a memory module (module 0) with 128 words of memory, and address 10000000 and above refer to devices attached to an I/O module (module 1).

The **control lines** are used to control the access to and the use of the data and address lines. Because the data and address lines are shared by all components,

there must be a means of controlling their use. Control signals transmit both command and timing information among system modules. Timing signals indicate the validity of data and address information. Command signals specify operations to be performed. Typical control lines include:

- **Memory write:** causes data on the bus to be written into the addressed location.
- **Memory read:** causes data from the addressed location to be placed on the bus.
- **I/O write:** causes data on the bus to be output to the addressed I/O port.
- **I/O read:** causes data from the addressed I/O port to be placed on the bus.
- **Transfer ACK:** indicates that data have been accepted from or placed on the bus.
- **Bus request:** indicates that a module needs to gain control of the bus.
- **Bus grant:** indicates that a requesting module has been granted control of the bus.
- **Interrupt request:** indicates that an interrupt is pending.
- **Interrupt ACK:** acknowledges that the pending interrupt has been recognized.
- **Clock:** is used to synchronize operations.
- **Reset:** initializes all modules.

The operation of the bus is as follows. If one module wishes to send data to another, it must do two things: (1) obtain the use of the bus, and (2) transfer data via the bus. If one module wishes to request data from another module, it must (1) obtain the use of the bus, and (2) transfer a request to the other module over the appropriate control and address lines. It must then wait for that second module to send the data.

3.5 POINT-TO-POINT INTERCONNECT

The shared bus architecture was the standard approach to interconnection between the processor and other components (memory, I/O, and so on) for decades. But contemporary systems increasingly rely on point-to-point interconnection rather than shared buses.

The principal reason driving the change from bus to point-to-point interconnect was the electrical constraints encountered with increasing the frequency of wide synchronous buses. At higher and higher data rates, it becomes increasingly difficult to perform the synchronization and arbitration functions in a timely fashion. Further, with the advent of multicore chips, with multiple processors and significant memory on a single chip, it was found that the use of a conventional shared bus on the same chip magnified the difficulties of increasing bus data rate and reducing bus latency to keep up with the processors. Compared to the shared bus, the point-to-point interconnect has lower latency, higher data rate, and better scalability.

In this section, we look at an important and representative example of the point-to-point interconnect approach: Intel's **QuickPath Interconnect (QPI)**, which was introduced in 2008.

The following are significant characteristics of QPI and other point-to-point interconnect schemes:

- **Multiple direct connections:** Multiple components within the system enjoy direct pairwise connections to other components. This eliminates the need for arbitration found in shared transmission systems.
- **Layered protocol architecture:** As found in network environments, such as TCP/IP-based data networks, these processor-level interconnects use a layered protocol architecture, rather than the simple use of control signals found in shared bus arrangements.
- **Packetized data transfer:** Data are not sent as a raw bit stream. Rather, data are sent as a sequence of packets, each of which includes control headers and error control codes.

Figure 3.17 illustrates a typical use of QPI on a multicore computer. The QPI links (indicated by the green arrow pairs in the figure) form a switching fabric that enables data to move throughout the network. Direct QPI connections can be established between each pair of core processors. If core A in Figure 3.17 needs to access the memory controller in core D, it sends its request through either cores B or C, which must in turn forward that request on to the memory controller in core D. Similarly, larger systems with eight or more processors can be built using processors with three links and routing traffic through intermediate processors.

In addition, QPI is used to connect to an I/O module, called an I/O hub (IOH). The IOH acts as a switch directing traffic to and from I/O devices. Typically in newer systems, the link from the IOH to the I/O device controller uses an interconnect technology called PCI Express (PCIe).

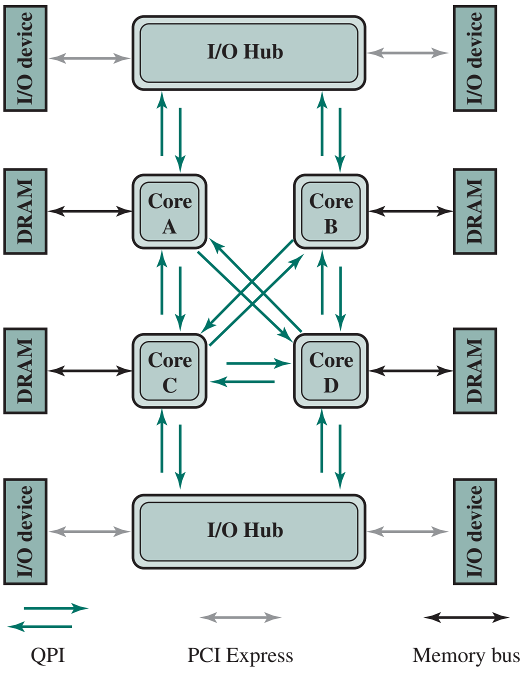

Figure 3.17 Multicore Configuration Using QPI

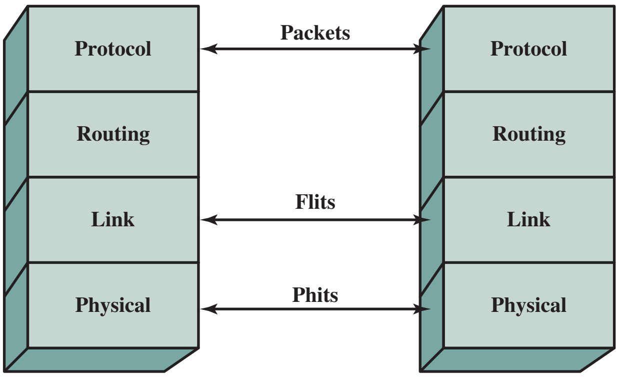

Figure 3.18 QPI Layers

The IOH translates between the QPI protocols and formats and the PCIe protocols and for-mats. A core also links to a main memory module (typically the memory uses dynamic access random memory (DRAM) technology) using a dedicated memory bus.

QPI is defined as a four-layer protocol architecture,[3] encompassing the following layers (Figure 3.18):

- **Physical:** Consists of the actual wires carrying the signals, as well as circuitry and logic to support ancillary features required in the transmission and receipt of the 1s and 0s. The unit of transfer at the Physical layer is 20 bits, which is called a **Phit** (physical unit).
- **Link:** Responsible for reliable transmission and flow control. The Link layer's unit of transfer is an 80-bit **Flit** (flow control unit).
- **Routing:** Provides the framework for directing packets through the fabric.
- **Protocol:** The high-level set of rules for exchanging**packets** of data between devices. A packet is comprised of an integral number of Flits.

QPI Physical Layer

Figure 3.19 shows the physical architecture of a QPI port. The QPI port consists of 84 individual links grouped as follows. Each data path consists of a pair of wires that transmits data one bit at a time; the pair is referred to as a **lane**. There are 20 data lanes in each direction (transmit and receive), plus a clock lane in each direction. Thus, QPI is capable of transmitting 20 bits in parallel in each direction. The 20-bit unit is referred to as a *phit*. Typical signaling speeds of the link in current products calls for operation at 6.4 GT/s (transfers per second). At 20 bits per transfer, that adds up to 16 GB/s, and since QPI links involve dedicated bidirectional pairs, the total capacity is 32 GB/s.

The lanes in each direction are grouped into four quadrants of 5 lanes each. In some applications, the link can also operate at half or quarter widths in order to reduce power consumption or work around failures.

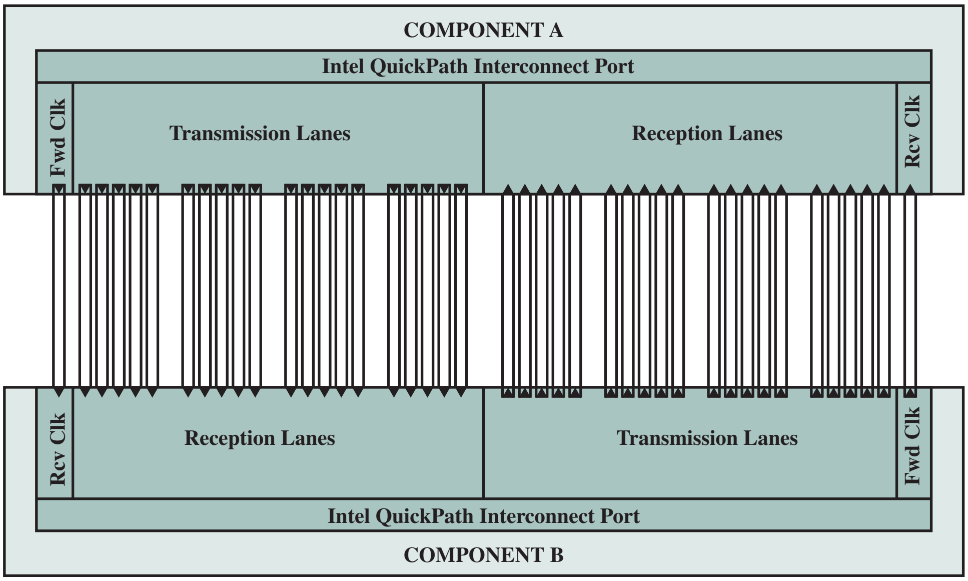

Figure 3.19 Physical Interface of the Intel QPI Interconnect

The form of transmission on each lane is known as **differential signaling**, or **balanced transmission**. With balanced transmission, signals are transmitted as a current that travels down one conductor and returns on the other. The binary value depends on the voltage difference. Typically, one line has a positive voltage value and the other line has zero voltage, and one line is associated with binary 1 and one line is associated with binary 0. Specifically, the technique used by QPI is known as *low-voltage differential signaling* (LVDS). In a typical implementation, the transmitter injects a small current into one wire or the other, depending on the logic level to be sent. The current passes through a resistor at the receiving end, and then returns in the opposite direction along the other wire. The receiver senses the polarity of the voltage across the resistor to determine the logic level.

Another function performed by the physical layer is that it manages the translation between 80-bit flits and 20-bit phits using a technique known as **multilane distribution**. The flits can be considered as a bit stream that is distributed across the data lanes in a round-robin fashion (first bit to first lane, second bit to second lane, etc.), as illustrated in Figure 3.20. This approach enables QPI to achieve very high data rates by implementing the physical link between two ports as multiple parallel channels.

QPI Link Layer

The QPI link layer performs two key functions: flow control and error control. These functions are performed as part of the QPI link layer protocol, and operate on the level of the flit (flow control unit). Each flit consists of a 72-bit message payload and an 8-bit error control code called a cyclic redundancy check (CRC).

A flit payload may consist of data or message information. The data flits transfer the actual bits of data between cores or between a core and an IOH. The message flits are used for such functions as flow control, error control, and cache coherence. We discuss cache coherence in Chapters 5 and 17.

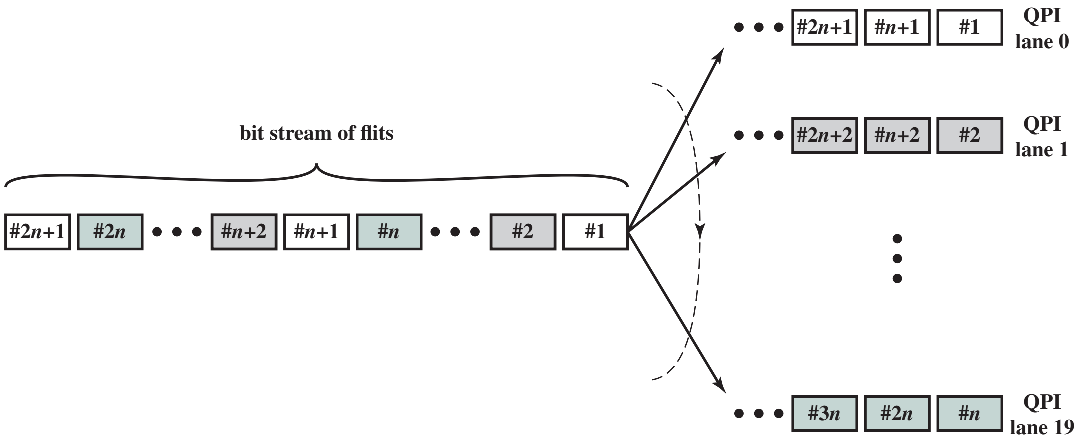

Figure 3.20 QPI Multilane Distribution

The **flow control function** is needed to ensure that a sending QPI entity does not overwhelm a receiving QPI entity by sending data faster than the receiver can process the data and clear buffers for more incoming data. To control the flow of data, QPI makes use of a credit scheme. During initialization, a sender is given a set number of credits to send flits to a receiver. Whenever a flit is sent to the receiver, the sender decrements its credit counters by one credit. Whenever a buffer is freed at the receiver, a credit is returned to the sender for that buffer. Thus, the receiver controls that pace at which data is transmitted over a QPI link.

Occasionally, a bit transmitted at the physical layer is changed during transmission, due to noise or some other phenomenon. The **error control function** at the link layer detects and recovers from such bit errors, and so isolates higher layers from experiencing bit errors. The procedure works as follows for a flow of data from system A to system B:

1. As mentioned, each 80-bit flit includes an 8-bit CRC field. The CRC is a function of the value of the remaining 72 bits. On transmission, A calculates a CRC value for each flit and inserts that value into the flit.
2. When a flit is received, B calculates a CRC value for the 72-bit payload and compares this value with the value of the incoming CRC value in the flit. If the two CRC values do not match, an error has been detected.
3. When B detects an error, it sends a request to A to retransmit the flit that is in error. However, because A may have had sufficient credit to send a stream of flits, so that additional flits have been transmitted after the flit in error and before A receives the request to retransmit. Therefore, the request is for A to back up and retransmit the damaged flit plus all subsequent flits.

QPI Routing Layer

The routing layer is used to determine the course that a packet will traverse across the available system interconnects. Routing tables are defined by firmware and describe the possible paths that a packet can follow. In small configurations, such as a two-socket platform, the routing options are limited and the routing tables quite simple. For larger systems, the routing table options are more complex, giving the flexibility of routing and rerouting traffic depending on how (1) devices are populated in the platform, (2) system resources are partitioned, and (3) reliability events result in mapping around a failing resource.

QPI Protocol Layer

In this layer, the packet is defined as the unit of transfer. The packet contents definition is standardized with some flexibility allowed to meet differing market segment requirements. One key function performed at this level is a cache coherency protocol, which deals with making sure that main memory values held in multiple caches are consistent. A typical data packet payload is a block of data being sent to or from a cache.