System Bus

Computer systems generally consist of three main parts, the central processing unit (CPU) to process data, main memory to hold the data to be processed, and a variety of peripherals to communicate that data with the outside world. An early computer might use a hand-wired CPU of vacuum tubes, a magnetic drum for main memory, and a punch tape and printer for reading and writing data. In a modern system we might find a multi-core CPU, DDR3 SDRAM for memory, a hard drive for secondary storage, a graphics card and LCD display as a display system, a mouse and keyboard for interaction, and a Wi-Fi connection for networking. In both examples, computer buses of one form or another move data between all of these devices.

In most traditional computer architectures, the CPU and main memory tend to be tightly coupled. A microprocessor conventionally is a single chip which has a number of electrical connections on its pins that can be used to select an "address" in the main memory and another set of pins to read and write the data stored at that location. In most cases, the CPU and memory share signaling characteristics and operate in synchrony. The bus connecting the CPU and memory is one of the defining characteristics of the system, and often referred to simply as the system bus.

In modern systems the performance difference between the CPU and main memory has grown so great that increasing amounts of high-speed memory is built directly into the CPU, known as a **cache**. In such systems, CPUs communicate using high-performance buses that operate at speeds much greater than memory, and communicate with memory using protocols similar to those used solely for peripherals in the past. These system buses are also used to communicate with most (or all) other peripherals, through adaptors, which in turn talk to other peripherals and controllers. Such systems are architecturally more similar to multicomputer, communicating over a **bus** rather than a network. In these cases, expansion buses are entirely separate and no longer share any architecture with their host CPU (and may in fact support many different CPUs, as is the case with **PCI**). What would have formerly been a system bus is now often known as a front-side bus.

Given these changes, the classical terms "system", "expansion" and "peripheral" no longer have the same connotations. Other common categorization systems are based on the buses primary role, connecting devices internally or externally, PCI vs. SCSI for instance. However, many common modern bus systems can be used for both; SATA and the associated eSATA are one example of a system that would formerly be described as internal, while in certain automotive applications use the primarily external IEEE 1394 in a fashion more similar to a system bus.

Internal bus

The internal bus, also known as internal data bus, memory bus, system bus or Front-Side-Bus, connects all the internal components of a computer, such as CPU and memory, to the motherboard. Internal data buses are also referred to as a local bus, because they are intended to connect to local devices. This bus is typically rather quick and is independent of the rest of the computer operations.

The internal bus are - PCI , AGP.

External bus

The external bus, or expansion bus, is made up of the electronic pathways that connect the different external devices, such as printer etc., to the computer.

What is a Bus? It is means of getting data from one point to another, point A to point B, one device to another device , or one device to multiple devices.

The **bus** includes not only the actual capability to transfer data between devices, **but also** :

1.all appropriate signaling information to ensure complete movement of the data from point A to point B.

2.To avoid loss of data, a bus must include a means of controlling the flow of data between two devices.

3. To ensure that both devices are ready to send / receive information.

4.Both ends must understand the speed with which data is to be exchanged.

5.A bus for all of these elements , and include a port definition to allow physical interfacing or connecting of two or more devices.

Typically , more than two devices are involved in a bus.The channels connecting only two devices are some times referred as a ports instead of buses. Buses often include **wires** to carry signals of addresses, data , control , status ,clock , and power.

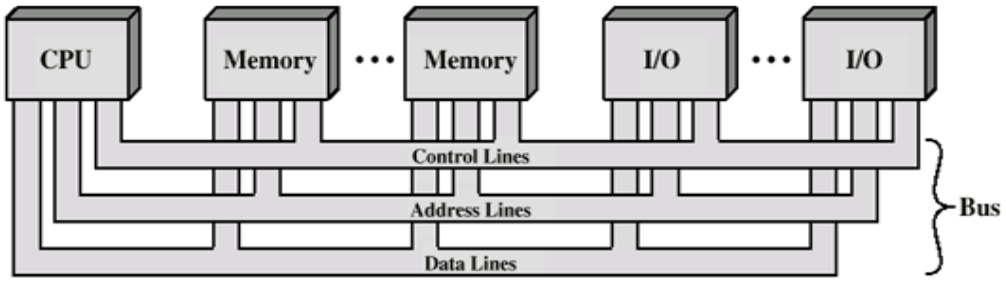
- The **address lines** indicate the source or destination of the data on the data lines.
- **Control lines** are used to implement the bus protocol.
- **Status lines** indicate the progress of the current transaction.
- **Clock signals** are used in synchronous bus systems to synchronize bus operations.

Bus is shared system interconnection :

- CPU to Cache
- Cache to main Memory
- CPU to I/O
- Main Memory to I/O

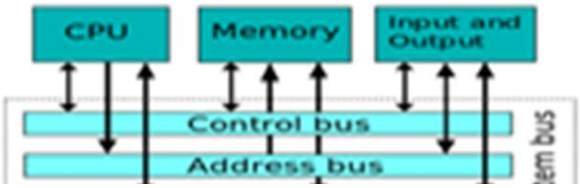
Connecting

- All the units must be connected
- Different type of connection for different type of unit
  - Memory, Input/Output, CPU

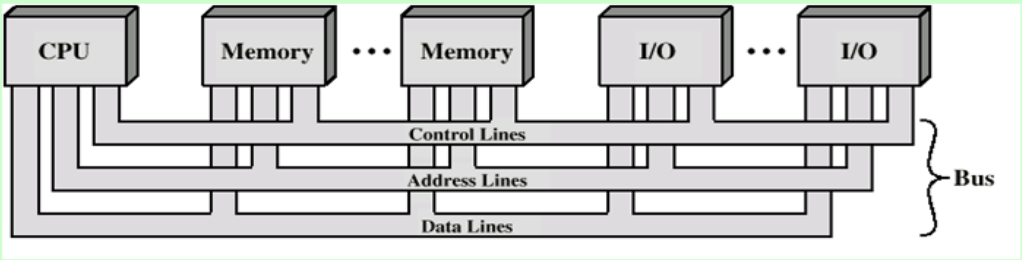


System bus consists of :

- Address Bus
- Data Bus
- Control Bus



Bus Interconnection Scheme



Parallel and Serial bus

Parallel buses are transmitting multiple bits at a time. Buses can be parallel buses, which carry data words in parallel on multiple wires. Parallel buses typically have 8,16,32,and 64 data lines. The PCI , ISA, VESA are examples for Parallel buses.

Serial buses are transmitting one bit at a time. , or serial buses, which carry data in bit-serial form. The speed of serial bus is generally expressed in bits per second(bps). Serial buses use the same line to transfer different data bits of the same byte/word. Typically they have only one data line and the bits are sent one after the other , as a packet. The Universal Serial Bus (USB) and IEEE 1394 bus architecture are examples of serial buses.

Serial buses are less expensive than parallel buses , but parallel buses have higher throughput.

In computer architecture, a bus is a communication system that transfers data between components inside a computer, or between computers. This expression covers all related hardware components (wire, optical fiber, etc.) and software, including communication protocols.

Early computer buses were parallel electrical wires with multiple connections, but the term is now used for any physical arrangement that provides the same logical functionality as a parallel electrical bus.

Modern computer buses can use both parallel and serial connections, and can be wired in either a multidrug (electrical parallel) or daisy chain topology, or connected by switched hubs, as in the case of USB.

For Example : with each character containing 8 bits , the character is sent between devices.

In serial mode, sending the first bit, then the second bit, third bit , and so on until the eighth bit is sent.

In parallel mode, allowing the devices transmit all of a character's bits simultaneously instead of one at a time.

Bus Protocols

Protocol refers to the set of rules agreed upon by both the bus master and bus slave as how data is to be transferred over the bus. Flow control is an important aspect of a protocol. Flow control is used to regulate the flow of information between devices.

Flow control is the ability of a receiving device to regulate the flow of data from the sending device. Protocols break the data into blocks or frames of data, hence the term block size. Some protocols support multiple block sizes, requiring the two communicating devices to agree on the block size before transmission. Typical block sizes are 128, 256, 512, 1024 (1K), and 2048 (2K) bytes. The sending device will send the information a block at a time. In addition, the sending device will perform a calculation on the bits of the data in the block. The result of this calculation is some form of check character or frame-check sequence. The block-

1.3.1. Synchronous Buses

With a synchronous protocol, data transfers occur in relation to successive edges of the system clock. Inherent in this type of protocol is the assumption that data will arrive within a certain time window (if it does not, then the data is lost).

1.3.2. Asynchronous Buses

Asynchronous bus transfers bear no particular timing relation to the system clock; transfers can take place at any time. Additional handshake lines are required in order to guarantee data transfers between master and slave. Synchronous bus transfers, by way of contrast, only depend on the system clock (the protocol being built into the system).

Asynchronous buses are useful when matching the different speeds of the CPU and peripheral chips. For example, a processor can interface to both slow and fast semiconductor memories using an asynchronous protocol. For a write operation, the protocol involves the bus master advising the slave that it has some data ready to send. The slave advises the master upon receipt of the data. The master then sends a message back to the slave acknowledging advice of the successful transfer. Finally, the slave responds to this acknowledgement from the master; a read operation follows a similar protocol. As with the synchronous bus, there will be delays due to address decoding, setup times and skew.

In synchronous buses, a bus clock signal provides the timing information for all actions on the bus. Change in other signals is relative to the falling or rising edge of the clock. In asynchronous buses, there is no clock signal. Instead, they use four-way handshaking to perform a bus transaction. Asynchronous buses allow more flexibility in timing.

The main advantage of asynchronous buses is that they eliminate this dependence on the bus clock. However, synchronous buses are easier to implement, as they do not use handshaking. Almost all system buses are synchronous, partly for historical reasons. In the early days, the difference between the speeds of various devices was not so great as it is now. Since synchronous buses are simpler to implement, designers chose them.

In synchronous buses, all timing must be a multiple of the bus clock. For example, if memory requires slightly more time than the default amount, we have to add a complete bus cycle. Of course, we can increase the bus frequency to counter this problem. But that introduces problems such as bus skew, increased power consumption, the need for faster circuits, and so on. Thus, choosing an appropriate bus clock frequency is very important for synchronous buses. In determining the clock frequency, we have to consider all devices that will be attached to the bus. When these devices are heterogeneous in the sense that their operating speeds are different, we have to operate the bus at the speed of the slowest device in the system.

Centralized implementations suffer from single-point failures due to the presence of the central arbiter. This causes two main problems:

1. If the central arbiter fails, there will be no arbitration.
2. The central arbiter can become a bottleneck limiting the performance of the whole system.

The distributed implementation avoids these problems. However, the arbitration logic has to be distributed among the masters. In contrast, in the centralized organization, the bus masters don't have the arbitration logic.

Modern Buses

1. PCI (Peripheral Component Interconnect Bus)

The PCI local bus with great performance capabilities met many demands. PCI is now a well-defined open standard. PCI is processor independent and is used on a number of different CPU-based systems, such as Intel.

Today you can find PCI video card. Most systems today include a PCI bus with slots.

PCI allows hierarchical PCI bus systems, which are typically built using PCI-to-PCI bridges (e.g., using the Intel 21152 PCI-to-PCI bridge chip). This chip connects two independent PCI buses: a primary and a secondary. The bridge improves performance due to the following reasons:

1. It allows concurrent operation of the two PCI buses. For example, a master and target on the same PCI bus can communicate while the other PCI bus is busy.
2. The bridge also provides traffic filtering which minimizes the traffic crossing over to the other side.

Obviously, this traffic separation along with concurrent operation improves overall system performance for bandwidth-hungry applications such as multimedia.

1.4.4. FireWire and USB

1.4.4.1. FireWire

The IEEE-1394 standard, known as FireWire, is a serial SCSI-bus standard supporting transfer rates from 100 to 400 MBps, expanding eventually up to 1.6 Gbps (gigabits per second) or faster. We will come back to this a bit later.

1.4.4.2. Universal Serial Bus

The Universal Serial Bus (USB) standard is for connecting keyboards, monitors, input devices, and digital cameras over a 12-Mbps bus network. USB is designed to simplify the connection of peripheral devices, provide increased I/O capacity, and provide maximum flexibility for the continued evolution of the PC. As connectivity demands increase, USB is to provide for this expansion.

USB should eventually replace the PC's keyboard, serial, and parallel connections with a simple jack architecture, with autodetection capabilities to know when a device is attached or unattached, complete with configuration support. Another goal of USB is to improve the connectivity of new peripherals by placing jacks in convenient locations. With today's computers, the main connection location is on the back of the computer, which is generally not conveniently accessible. With USB, the potential exists to put ports on the front of the computer, on the monitor, keyboard, etc., making it extremely convenient for the typical user to connect more peripherals. It also offers great flexibility and scalability.

USB is a cable bus that supports data exchange between a host computer and a wide range of simultaneously accessible peripherals. The attached peripherals share USB bandwidth through a host scheduled token based protocol. The bus allows peripherals to be attached, configured, used, and detached while the host and other peripherals are in operation. This is referred to as dynamic (or hot) attachment and removal.

The USB interconnect is the manner in which USB devices are connected to and communicate with the host. This includes:

- Bus Topology: Connection model between USB devices and the host.
- Inter-layer Relationships: In terms of a capability stack, the USB tasks that are performed at each layer in the system.
- Data Flow Models: The manner in which data moves in the system over the USB between producers and consumers.
- Scheduling the USB: USB provides a shared interconnect. Access to the interconnect is scheduled in order to support isochronous data transfers.

Table: Bus Architectures Comparisons

Bus Type	Industry Reference	Bus Width (Bits)	Bus Speed (MHz)	Transfer Rate (MBS)
ISA 8-BIT	Industry Standard Architecture	8	8	4
ISA 16-BIT	Industry Standard Architecture	16	8	8
MCA (32 bit)	Microchannel Architecture	32	8	33
EISA	Extended Industry Standard Architecture	32	8.33	33.3
VL-BUS	VESA LocalBus	33	33 40 50	128-132
PCI (32 bit)	Peripheral Component Interconnect	32	33	132
PCI (64 bit)	Peripheral Component Interconnect	64	33	264
SCSI	Small Computer System Interface	(see above)		
1394	FireWire		100-400	100-400 Mbps
USB	Universal Serial Bus		12	12 Mbps

Example Buses

- We look at five buses
  - * ISA
    - » Supports older text-based applications
  - * PCI
    - » Supports modern window-based systems
  - * AGP
    - » Supports high-performance graphics and full-motion video
  - * PCI-X
    - » Improved and faster PCI
  - * PCMCIA
    - » Useful for laptops

Summary

A bus is a common pathway to connect various subsystems in a computer system. A bus consists of the connection media like wires and connectors, and a bus protocol. Buses can be serial or parallel, synchronous or asynchronous. Depending on these and other features, several bus architectures have been devised in the past. The Universal Serial Bus (USB) and IEEE 1394 are examples of serial buses while the ISA and PCI buses are examples of popular parallel buses. This article first describes fundamental information on bus architectures and bus protocols, and then provides specific information on various industry standard bus architectures from the past and the present, and their advantages and disadvantages. It also describes how different types of bus architectures are used simultaneously in different parts of a modern personal computer.