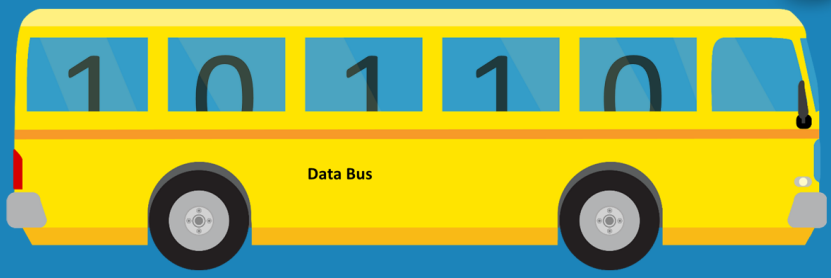
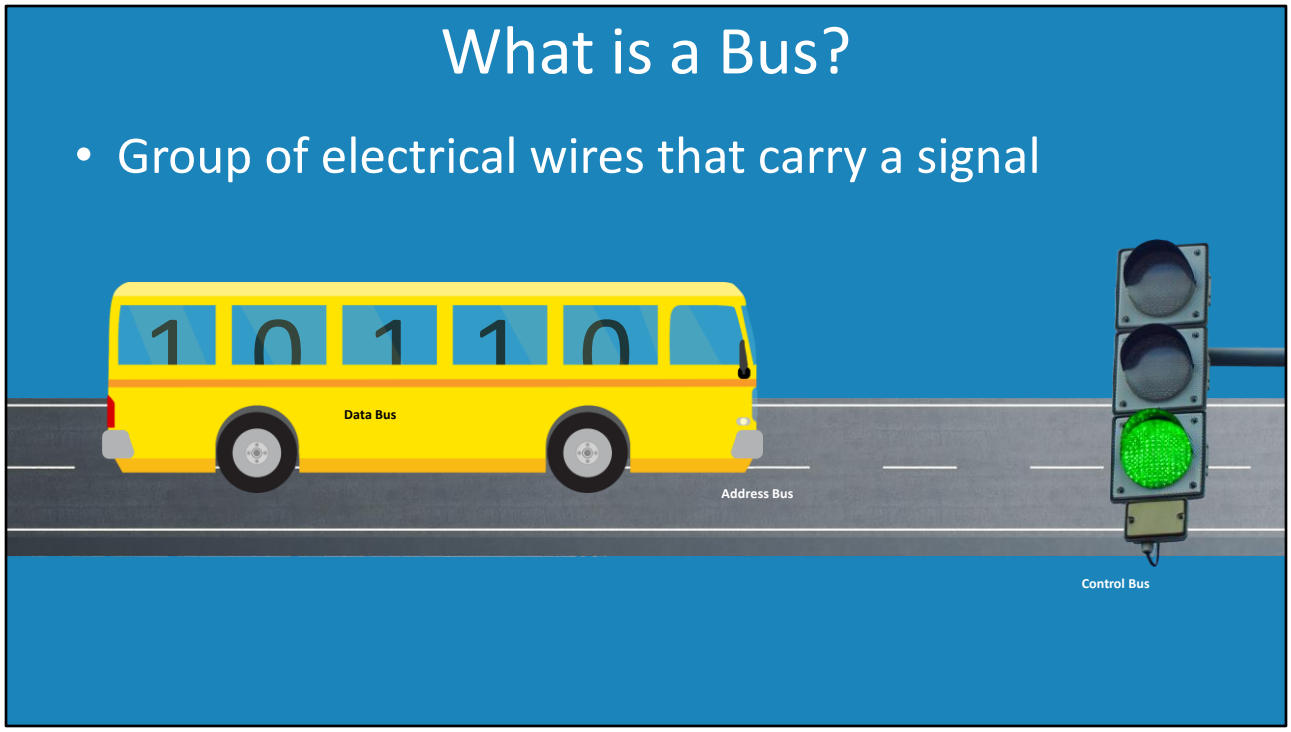
ITFreeTraining



Computer Buses

For the free video please see
http://itfreetraining.com/ap/1b75

In this video I will look at the buses that are used inside a computer. A bus is a communication pathway that allows data to travel between different components inside a computer. Understanding how buses work will give you a better understanding of what you can achieve with a computer. This will help you determine where potential bottlenecks in system performance may occur when the computer is put under load.



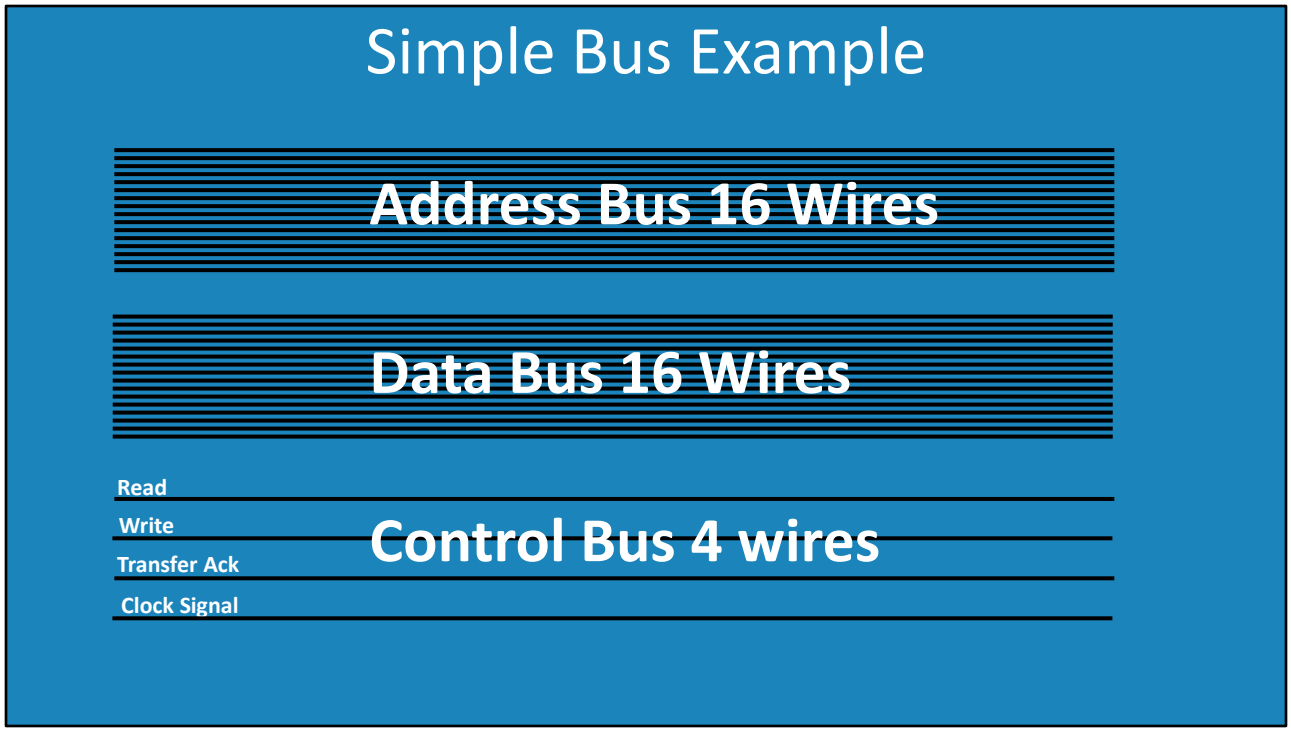
0:23 To start with, let's consider the question what is a computer bus or as it's better known, a system bus? A bus inside a computer is essentially a group of electrical wires that carry a signal. On a motherboard, these wires are called traces.

The bus carries data, but you need to control the movement of the data and determine where it is going.

In order to do this, a typical system bus is divided into three parts. The first part is the data bus. This bus carries the data. A computer bus works just like a regular bus used to carry people. The regular bus will have a limited number of seats. A system bus will have a limited number of bits it can send at once. For example, a 32-bit bus can send 32 bits at once. Think of it like a bus with 32 seats and, in order for the bus to leave, it needs to have all 32 seats occupied. A 32-bit bus always sends 32 bits of data at once, no more, no less.

The next step is that the data needs to know where it is going. This is achieved by using the address bus. The address bus essentially tells the computer where the data is going. For example, going to the main memory or to memory on a device. More on that later in the video.

Lastly, there is the control bus. The control bus determines when data is sent. Without any control the computer would not be able to determine when data is sent and when it is complete. Previous data would get mixed up with new data and thus transfer errors would occur.



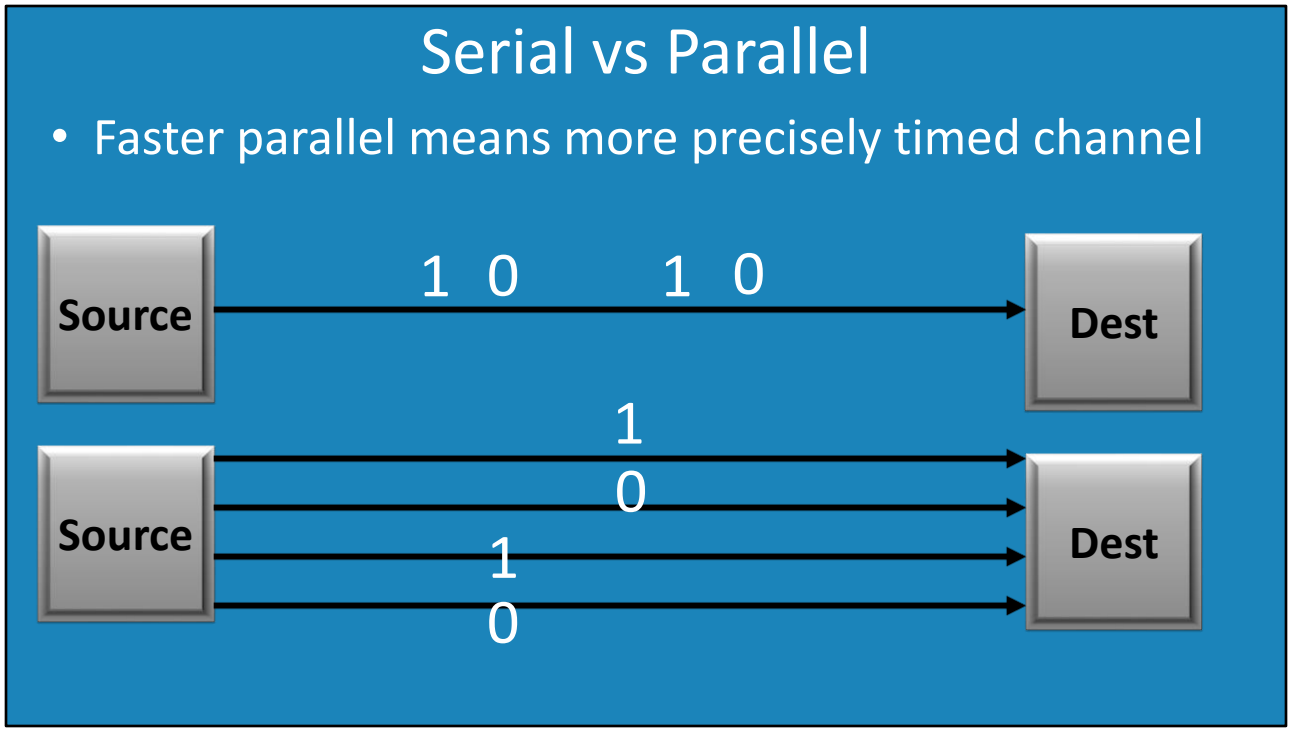
2:04 To understand the bus better, let's consider a simple example of how a bus may work. In the old days of computing this is how buses worked. Later in the video, we will see the problems with using a bus designed like this and how modern buses work differently to overcome these problems.

To start with, let's consider there are 16 wires for the address bus. This will allow us to access 64 kilobytes of memory, not a lot by today's standards. Now let's consider that we have a 16-bit data bus. 16 bits can be sent or received on the data bus at once, which is two bytes.

Lastly, we have our control bus. In this example there are only four wires. This is a simple example and you would generally have more. But for this example, there is a read, write, transfer acknowledgment and a clock signal wire.

The clock signal will determine when data is sent. The read and write wires determine if data is being sent and received and the transfer acknowledgment wire determines if the data went through o.k. So essentially, set the address bus to the address you want to read or write to, place or receive your data on the data bus and use the control wires to determine when to start the transfer, what sort of transfer, be it either read or write and if it was received.

This is a very simple example that you may find in a very old computer, but it is not without its problems. Consider that with this example we have 36 wires. Consider now that we wanted to send 32 or 64 bits at once. This would greatly increase the number of wires required, however having extra wires is not the only problem this would create.



3:52 This brings us to the topic of serial versus parallel. Serial transmission is when data is transferred using one channel. Each bit follows the previous bit, as in 'follow the leader'. Parallel is when multiple channels are used at the same time. Think of it like line dancing, where everyone is in a line and dancing in sync with each other

In the old days of computing, serial was the first technology to be used but was very slow. In order to improve speed, multiple channels were combined together to increase the speed. It makes sense, as sending more data at once greatly increases speed. Thus, old computers started using parallel transmissions to increase speed.

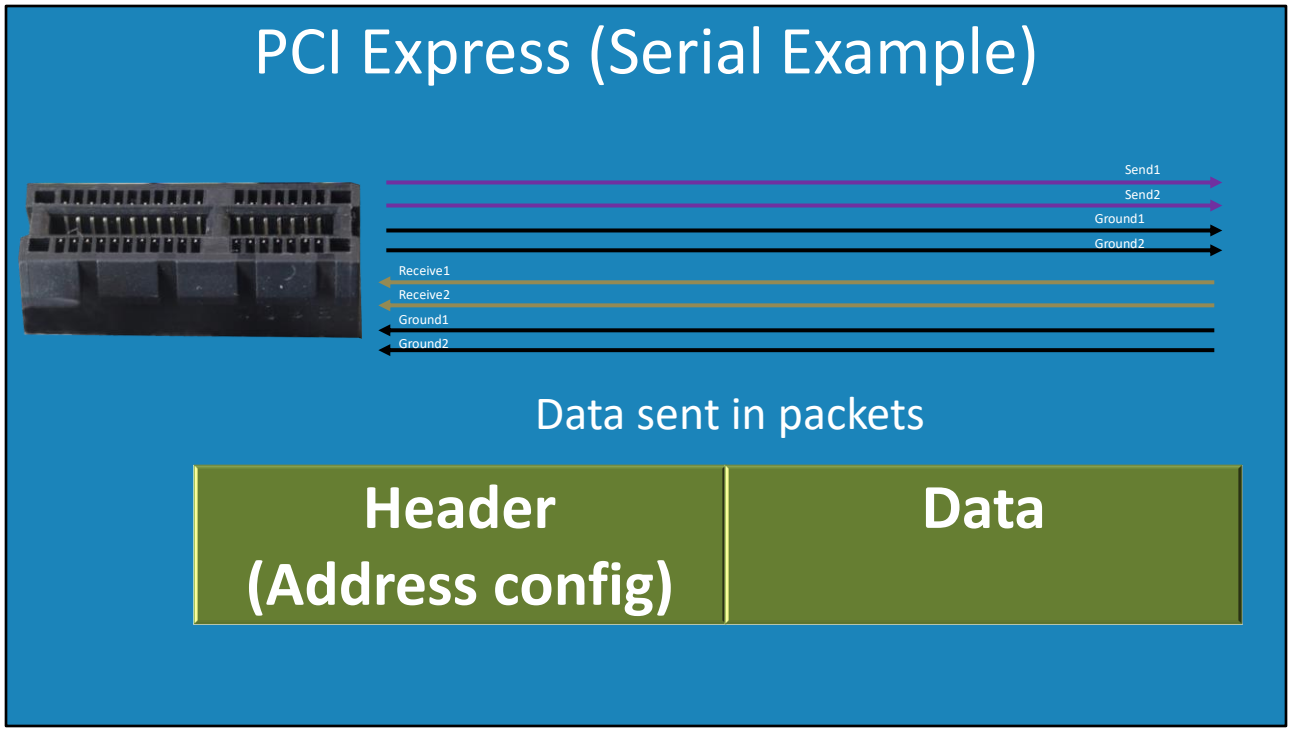
Consider this example, there are four bits being transmitted. The first using serial and the second using parallel. There is a small gap in transmission in-between the bits. It is difficult to precisely time electronics at high speed so sometimes there will be small delays in transmissions at high speed.

Now consider our line dancing example. If you were to speed the music up everyone in the line would need to match the music. As the music gets faster this gets harder and harder. With parallel communication, all data must arrive at the destination before any more can be sent. If all channels are not exactly in sync, the transmission needs to slow down to keep it in sync, this limits the speed parallel can run at. Essentially, faster parallel communication requires each channel to be precisely timed with every other channel. Now consider you have 64 channels, that is a lot of channels to have precisely timed with each other. If the channels get out of sync with each other this is referred to as clock skew.

In the case of serial, if you consider a ‘follow the leader’ example, if a person in the line were to go slower than the others, the effect would be that everyone behind that person would also slow down. It is a lot easier to time one channel then trying to keep multiple channels in sync with each other.

As electronics improved, serial communication speeds increased and thus started getting faster than parallel communication. For these reasons, serial transmissions have started to replace parallel communication.

In newer computers, you will find serial communication has replaced parallel for most of the buses in the computer including the buses from the CPU. This allows higher speed and also requires less wires to connect devices together. Memory modules still use parallel communication; maybe one day that will change, but you will find that just about everything else nowadays is serial communication.



6:43 To better understand how serial communication works, I will use PCI Express as an example. There are a number of different ways that serial communication can be implemented; however, you will find they are similar to how PCI Express is implemented.

In this example, I will use the PCI Express by 1 slot. This slot only has one lane of traffic, making it easy to understand. Each PCI Express slot has a number of pins for power and other functions. Each lane will have eight wires that are used for data transmission.

The first four are for sending data. Of these four wires, the first two are for sending and the last two are ground wires. In some cases, there may only be one ground wire and the second wire may be reserved.

You may be wondering why there are two send wires when it is using serial communication. The reason is that the two wires are run next to each other to reduce data corruption from interference. When sending data, the difference between the two signals is measured. Since both wires are next two each other, they will both equally be affected by interference and thus the difference will remain the same even with interference quite high. Running an extra wire is a small price to pay to reduce interference and thus increase the amount of data that can be sent.

The same principal applies to the receive wires – four wires, two of which are for receiving and two are for ground. Using two receive wires electrically reduces interference and thus increases the speed data can be transmitted at.

In old buses, to save wires the bus may work in both directions. Essentially a switch would be flipped to determine if the bus was receiving or sending. Nowadays you will find that buses will use separate wires for sending and receiving.

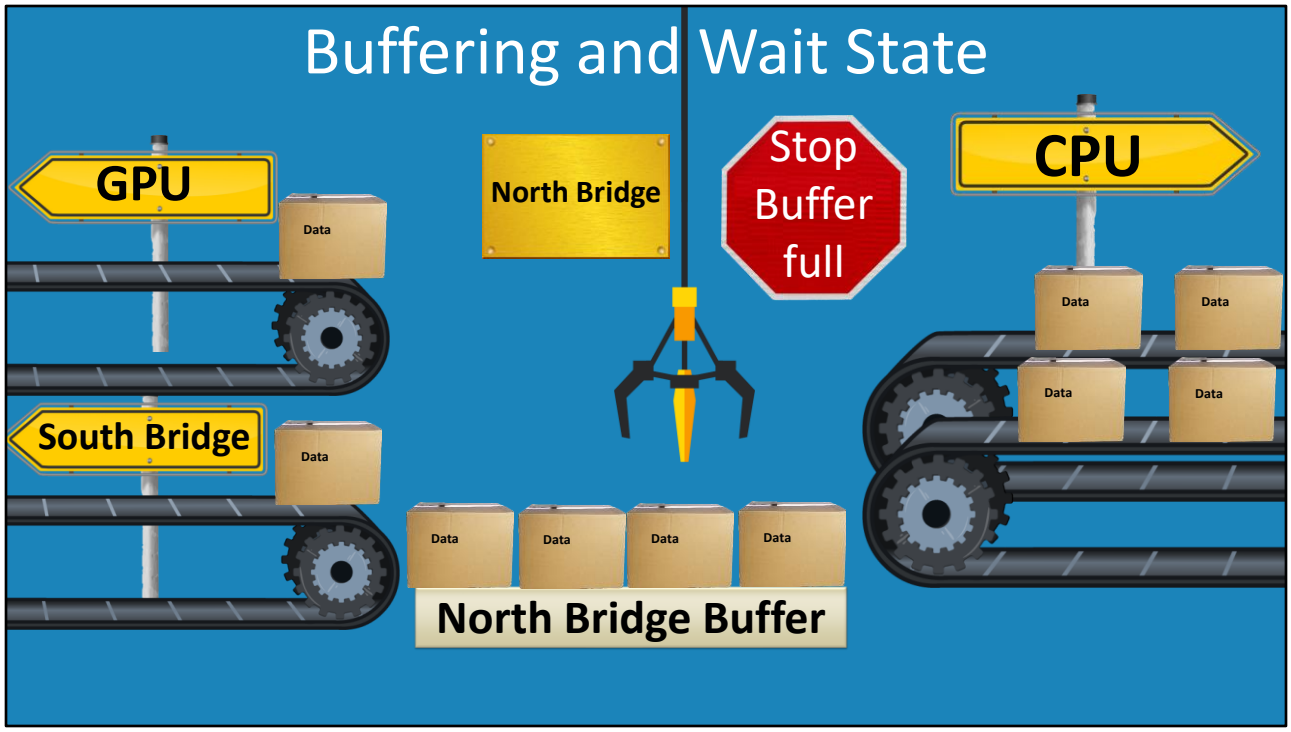
The next point to consider is how data is sent from the PCI Express slot. The way this is achieved is by using packets. The packet contains a 96 to 128-bit header and a variable length amount of data up to four kilobytes. The data sent is always a multiple of 32 bits.

The header contains the address and other configuration information. All data sent contains extra bits which are used for error checking. The version of PCI Express being used will determine how much extra data is needed for error checking.

Having the address in the header means separate wires are not required for the address bus as would have been required in previous computers. You can see that using serial communication has significantly reduced the number of wires required. If you want to increase the speed further, you just add additional lanes. For example, with a PCI Express by 16 slot, there are 16 lanes of traffic going into and out of the slot.

You will notice that there is no wire for the clock signal. With PCI Express, there is no external clock signal since the timing signal is embedded in the data signal itself. This is one of the big advantages of serial communication. Since data does not need to line up with the clock signal, it does not matter if the data traveling over it runs at slightly different speeds. The other end will simply process it at the speed it receives it.

The next point to consider is what happens when you have multiple devices, chips and buses which are connected together. Here there are potentially different devices of different speeds and there may be large amounts of data being transferred around.



10:22 In order to match the speeds between different devices and components, buffering methods and wait states are used. Consider this example.

Consider that you have a North Bridge chip. The North Bridge is connected to the CPU. In this example, there are two conveyor belts which represent data channels. The two conveyor belts are exiting the North Bridge which transports data to the GPU and the South Bridge. Since the GPU or Graphics Processing Unit requires a lot of data, it is not uncommon for it to be directly connected to the North Bridge or the CPU itself.

As the CPU is the fastest component in any computer system, it will be sending data to the North Bridge quite fast. The buses to other components will be slower than the CPU and therefore, in order to match the speeds, a buffer is used.

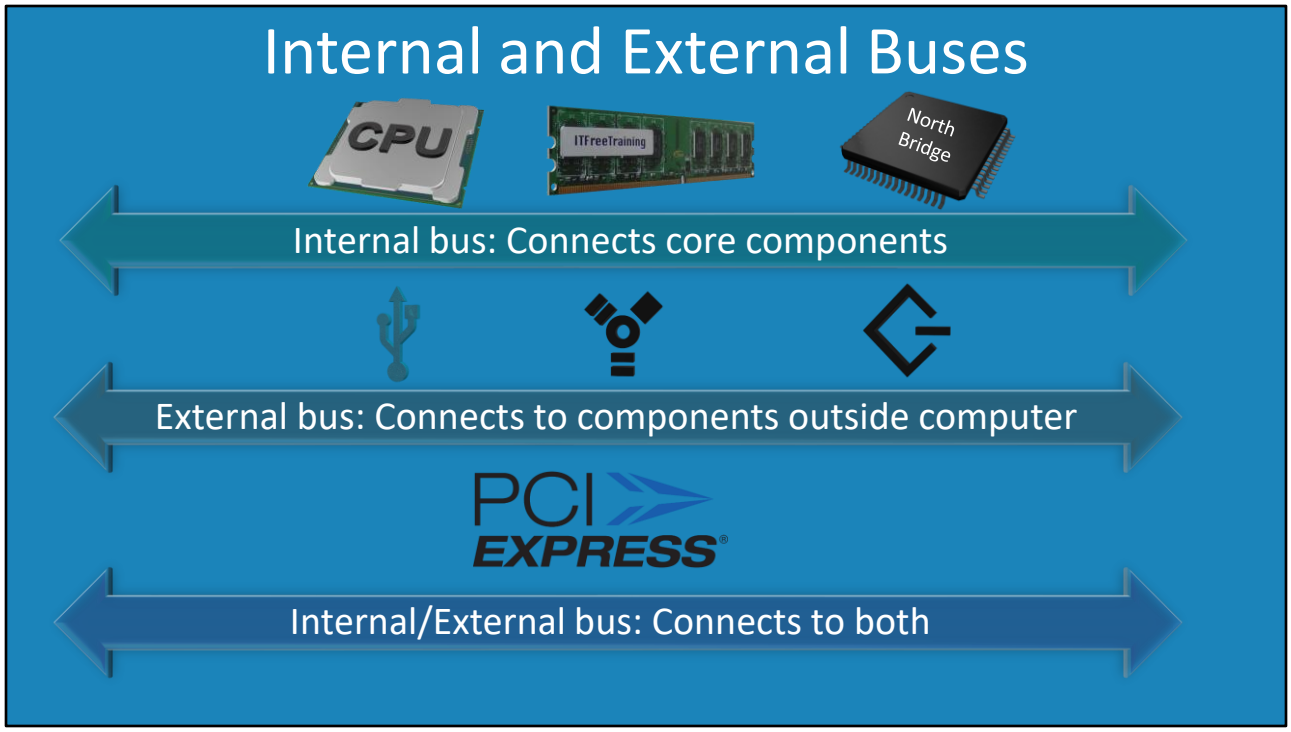
Essentially what happens is, as the data comes in, the North Bridge takes the data from the buffer and places it on the bus. If the buses are slightly out of sync, the data stays in the buffer until it can be placed on the bus. By having small delays in the buffers, this gets around the buses being out of sync with each other.

The problem occurs when the buffer is full. When this occurs, the incoming data needs to be stopped, otherwise it will be lost. This is done by sending a control signal on the bus telling it to stop. This is called a wait state. Once the buffer is freed up, the bus starts again.

The CPU, being the fastest component in a computer, is often placed into a wait state. A

number of different design techniques are performed to prevent the CPU from going into a wait state, but you can only do so much.

You can start to understand why a computer can be under a heavy load and for the CPU utilization to be low. The computer is under load, but the load is not in the CPU. Bottlenecks can potentially occur anywhere in a computer system. Understanding how the buses work gives you a better understanding of where these bottlenecks may occur. Sometimes because of the different speeds of buses, upgrading from SATA 2 to SATA 3 or changing an expansion card to a different slot may make a big difference to your computer's performance.



12:41 Buses in the computer can be either internal or external. Internal buses are generally high-speed buses that are inside the computer. Examples include, the bus that connects the CPU, memory and the chips like the North Bridge.

External buses connect devices outside the computer. These buses are generally slower than the internal buses. Examples include, USB, Firewire and SCSI. SCSI can be used to connect devices outside a computer; however, it can also be used to connect devices inside the computer.

Some buses can be both internal and external. An example of this is PCI Express. PCI Express can be used internally in a computer and also can be used to connect devices external to the computer. You will find that, in a bus that supports both, external speeds will generally be slower than what can be achieved internally inside the computer.

Storage Bus

- A bus to communicate with storage devices
- Host Bus Adapters (HBA)

(PATA) Parallel ATA
(IDE) Integrated Drive Electronics
(EIDE) Enhanced IDE

(SATA) Serial ATA

(SCSI) Small Computer System Interface

13:38 You may also come across the term “storage bus”. A storage bus is simply a bus designed for the purpose of communicating with storage devices such as hard disks, optical devices and solid-state drives. You may also come across the term “Host Bus Adapter” or HBA. This refers to the adapter or device that supplies the bus for the storage devices. The term is commonly used in servers and storage devices.

To access devices, the parallel ATA interface or PATA was very popular in the 80s. It was also known as IDE standing for Integrated Drive Electronics. There were improvements made to it after it was introduced so it may also be called Enhanced IDE.

The interface is obsolete nowadays, but you may find that your motherboard has this connection on it. It may have either one of the three names. Regardless of what name is used, since it is old technology, modern motherboards will support all the old and new features of the interface. So, if you are using a modern motherboard, don’t worry about what the connection is called, it will support all the new and old features of the interface.

Nowadays, you will most likely be using a SATA connection to connect the storage devices in your computer. SATA is a serial connection while IDE was parallel. Previously in computing parallel was faster than serial. However, as computers started to increase in speed, timing parallel connections became harder to achieve. It was found that better results could be achieved by using serial connections rather than parallel.

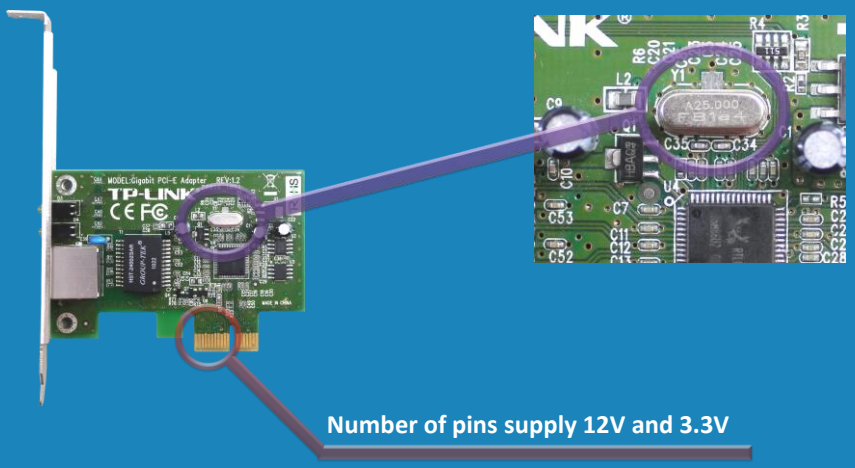
The other connection that is commonly used is SCSI. This stands for Small Computer System

Interface. Traditionally SCSI devices cost more, so you would only see SCSI used in servers or high-end work stations.

The SCSI interface has changed a large number of times. Therefore, you need to make sure you have the right interface and cable for the SCSI device you are using. Traditionally, SCSI was parallel, but now it has made the change to serial. There are other kinds of storage buses that are used, but these are the main ones you are going to come across.

Expansion Cards

- Many expansion cards have their own crystal



15:50 The last point that I would like to cover is expansion cards. Shown here is an example of a PCI Express by 1 expansion card. The expansion card will receive a clock signal from the motherboard. All expansion slots have a standardized clock rate. If different motherboards used different clock rates then a different expansion card would need to be manufactured to support each different clock rate. This now means that only one expansion card needs to be made for each slot type.

This creates the problem that the manufacturer is stuck with that clock rate and it may not be suitable for their needs. For this reason, you will find that a lot of expansion cards will have their own crystal oscillator. A crystal is used to generate a clock rate. Having a crystal on the expansion card allows the manufacturer of the expansion card to create their own clock rate totally independently of the clock rates used on the motherboard.

The expansion card will still need to sync its transfers up with the clock rate supplied by the motherboard. It does this using buffers and wait states as in the previous example of the North Bridge.

In order to power the crystal, the expansion card receives 12 and 3.3 volt power from the motherboard. This will be the first set of pins found on the expansion card. Not all the pins will be power, but a large number of them are. This power allows the expansion card to power the crystal to generate its own clock signal.

You should now understand that a computer system uses a number of buses for devices to

communicate with each other. Buffers are used so different speed devices can communicate with each other. If too much data comes in at once, a wait state is used to temporarily stop the data from being transmitted.

That concludes this video from ITFreeTraining. I hope that has given you a better understanding of what buses do inside a computer system. Until the next video from us, I would like to thank you for watching.

References

"The Official CompTIA A+ Core Study Guide (Exam 220-1001)" Chapter 3 position 6269-6554, 7999-8198
"CompTIA A+ Certification Exam Guide Ninth Edition" pages 240-243
"Bus (computing)" https://en.wikipedia.org/wiki/Bus_(computing)
"Address bus" https://en.wikipedia.org/wiki/Address_bus
"Control Bus" https://en.wikipedia.org/wiki/Control_bus
"Wait state" https://en.wikipedia.org/wiki/Wait_state
"Fun and Easy PCIE - How the PCI Express Protocol works" https://www.youtube.com/watch?v=sRx2YLzBIqk
"Picture PCI Express" https://de.wikipedia.org/wiki/Datei:PCI_Express.svg

Credits

Trainer: Austin Mason http://ITFreeTraining.com
Voice Talent: HP Lewis http://hplewis.com
Quality Assurance: Brett Batson http://www.pbb-proofreading.uk