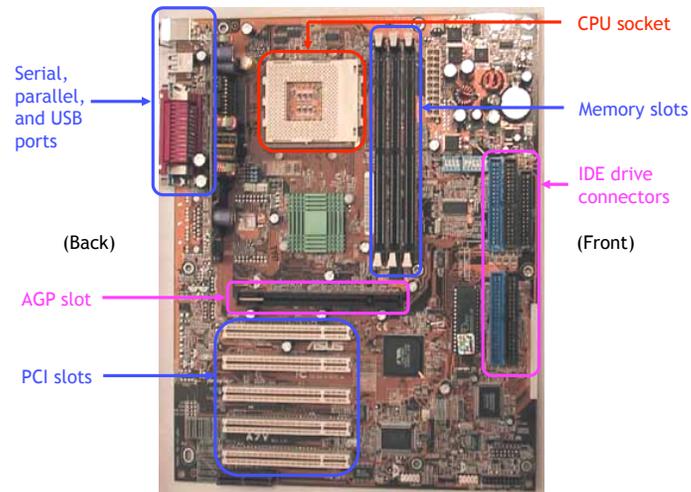


## The Hardware (the motherboard)



1

## What is all that stuff?

- Different motherboards support different CPUs, types of memories, and expansion options.
- The picture is an Asus A7V.
  - The **CPU socket** supports AMD Duron and Athlon processors.
  - There are three **DIMM slots** for standard PC100 memory. Using 512MB DIMMs, you can get up to 1.5GB of main memory.
  - The **AGP slot** is for video cards, which generate and send images from the PC to a monitor.
  - **IDE ports** connect internal storage devices like hard drives, CD-ROMs, and Zip drives.
  - **PCI slots** hold other internal devices such as network and sound cards and modems.
  - **Serial, parallel and USB ports** are used to attach external devices such as scanners and printers.

2

## Network (and I/O) Performance

There are two fundamental performance metrics for I/O systems:

- **Bandwidth**: the amount of data that can be transferred in unit time (units = bytes/time)
  - This is a primary concern for applications which transfer large amounts of data in big blocks.
  - If you download large files, bandwidth will be the limiting factor.
- **Latency**: the time taken for the smallest transfer (units = time)
  - This is a primary concern for programs that do many small dependent transfers.
  - It takes time for bits to travel across states, countries and oceans!

```
>ping www.uiuc.edu
Approximate round trip times in milli-seconds:
  Minimum = 104ms, Maximum = 115ms, Average = 112ms

>ping www.stanford.edu
Approximate round trip times in milli-seconds:
  Minimum = 160ms, Maximum = 170ms, Average = 164ms

>ping nus.edu.sg
Approximate round trip times in milli-seconds:
  Minimum = 410ms, Maximum = 437ms, Average = 420ms
```

December 4, 2009

3

## Back of the Envelope Calculation

- Because the transmission of network packets can be pipelined, the time for a transfer can be estimated as:

$$\begin{aligned} \text{Transfer time} &= \text{latency} + \text{transfer\_size} / \text{bandwidth} \\ &= \text{time} + \text{bytes} / (\text{bytes}/\text{time}) \end{aligned}$$

Dominant term for  
small transfers

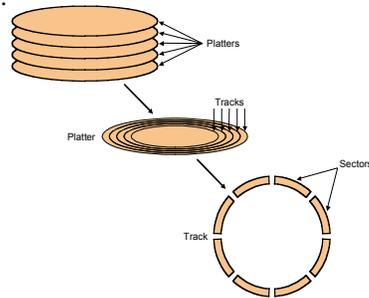
Dominant term for  
large transfers

December 4, 2009

4

## Hard drives

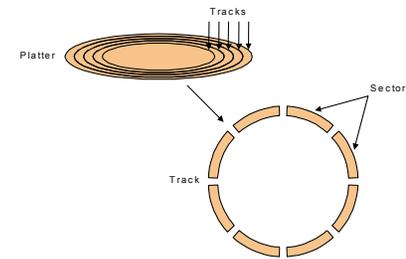
- The textbook shows the ugly guts of a hard disk.
  - Data is stored on double-sided magnetic disks called **platters**.
  - Each platter is arranged like a record, with many concentric **tracks**.
  - Tracks are further divided into individual **sectors**, which are the basic unit of data transfer.
  - Each surface has a read/write head like the arm on a record player, but all the heads are connected and move together.
- A 75GB IBM Deskstar has roughly:
  - 5 platters (10 surfaces),
  - 27,000 tracks per surface,
  - 512 sectors per track, and
  - 512 bytes per sector.



5

## Accessing data on a hard disk

- Accessing a sector on a track on a hard disk takes a lot of time!
  - **Seek time** measures the delay for the disk head to reach the track.
  - A **rotational delay** accounts for the time to get to the right sector.
  - The **transfer time** is how long the actual data read or write takes.
  - There may be additional **overhead** for the operating system or the controller hardware on the hard disk drive.
- **Rotational speed**, measured in revolutions per minute or RPM, partially determines the rotational delay and transfer time.



6

### Estimating disk latencies (seek time)

---

- Manufacturers often report *average* seek times of 8-10ms.
  - These times average the time to seek from any track to any other track.
- In practice, seek times are often much better.
  - For example, if the head is already on or near the desired track, then seek time is much smaller. In other words, **locality** is important!
  - Actual average seek times are often just 2-3ms.

7

### Estimating Disk Latencies (rotational latency)

---

- Once the head is in place, we need to wait until the right sector is underneath the head.
  - This may require as little as **no time** (reading consecutive sectors) or as much as **a full rotation** (just missed it).
  - On **average**, for **random** reads/writes, we can assume that the disk spins halfway on average.
- Rotational delay depends partly on how fast the disk platters spin.
  - Average rotational delay = 0.5 x rotations x rotational speed**
  - For example, a 5400 RPM disk has an average rotational delay of:  
**0.5 rotations / (5400 rotations/minute) = 5.55ms**

8

## Estimating disk times

- The overall **response time** is the sum of the seek time, rotational delay, transfer time, and overhead.
- Assume a disk has the following specifications.
  - An average seek time of 9ms
  - A 5400 RPM rotational speed
  - A 10MB/s average transfer rate
  - 2ms of overhead
- How long does it take to read a random 1,024 byte sector?
  - The average rotational delay is 5.55ms.
  - The transfer time will be about  $(1024 \text{ bytes} / 10 \text{ MB/s}) = 0.1\text{ms}$ .
  - The response time is then  $9\text{ms} + 5.55\text{ms} + 0.1\text{ms} + 2\text{ms} = 16.7\text{ms}$ .  
That's 16,700,000 cycles for a 1GHz processor!
- One possible measure of throughput would be the number of random sectors that can be read in one second.

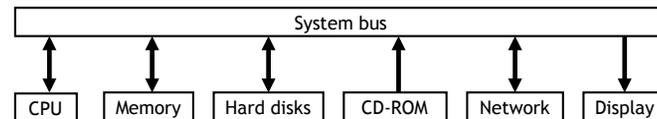


$$(1 \text{ sector} / 16.7\text{ms}) \times (1000\text{ms} / 1\text{s}) = 60 \text{ sectors/second.}$$

9

## Computer buses

- Every computer has several small “networks” inside, called **buses**, to connect processors, memory, and I/O devices.
- The simplest kind of bus is linear, as shown below.
  - All devices share the same bus.
  - Only one device at a time may transfer data on the bus.
- Simple is not always good!
  - With many devices, there might be a lot of contention.
  - The distance from one end of the bus to the other may also be relatively long, increasing latencies.

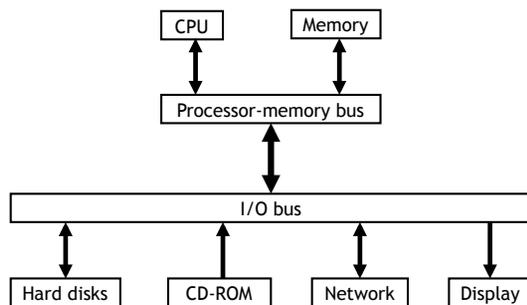


December 4, 2009

10

## Hierarchical buses

- We could split the bus into different segments.
  - Since the CPU and memory need to communicate so often, a shorter and faster **processor-memory bus** can be dedicated to them.
  - A separate **I/O bus** would connect the slower devices to each other, and eventually to the processor.

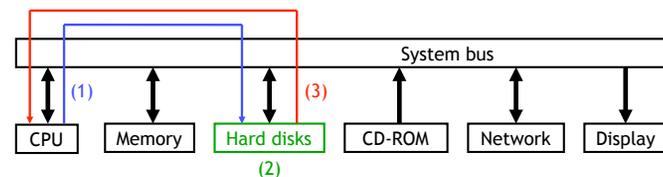


December 4, 2009

11

## Basic bus protocols

- Although physically, our computer may have a hierarchy of buses (for performance), logically it behaves like a single bus
- I/O reads and writes can be programmed like loads and stores, using addresses.
- Two devices might interact as follows.
  1. An initiator sends an address and data over the bus to a target.
  2. The target processes the request by “reading” or “writing” data.
  3. The target sends a reply over the bus back to the initiator.
- The **bus width** limits the number of bits transferred per cycle.

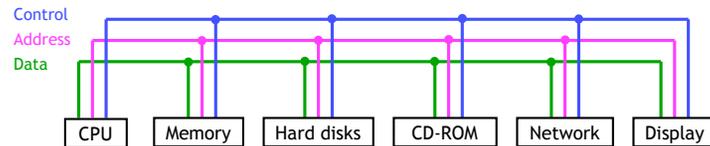


December 4, 2009

12

## What is the bus anyway?

- A bus is just a bunch of wires which transmits three kinds of information.
  - **Control signals** specify commands like “read” or “write.”
  - The location on the device to read or write is the **address**.
  - Finally, there is also the actual **data** being transferred.
- Some buses include separate control, address and data lines, so all of this information can be sent in one clock cycle.

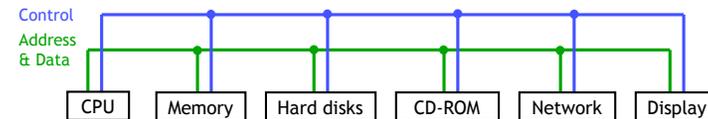


December 4, 2009

13

## Multiplexed bus lines

- Unfortunately, this could lead to many wires and wires cost money.
  - Many buses transfer 32 to 64 bits of data at a time.
  - Addresses are usually at least 32-bits long.
- Another common approach is to **multiplex** some lines.
  - For example, we can use the same lines to send both the address and the data, one after the other.
  - The drawback is that now it takes *two* cycles to transmit both pieces of information.



December 4, 2009

14

### Example bus problems

---

- I/O problems always start with some assumptions about a system.
  - A CPU and memory share a 32-bit bus running at 100MHz.
  - The memory needs 50ns to access a 64-bit value from one address.
- Then, questions generally ask about the latency or throughput.
  - How long does it take to read one address of memory?
  - How many random addresses can be read per second?
- You need to find the total time for a single transaction.
  1. It takes one cycle to send a 32-bit address to the memory.
  2. The memory needs 50ns, or 5 cycles, to read a 64-bit value.
  3. It takes two cycles to send 64 bits over a 32-bit wide bus.
- Then you can calculate latencies and throughputs.
  - The time to read from one address is eight cycles or 80ns.
  - You can do 12.5 million reads per second, for an **effective bandwidth** of  $(12.5 \times 10^6 \text{ reads/second}) \times (8 \text{ bytes/read}) = 100\text{MB/s}$ .

December 4, 2009

15

### Example Bus Problems, cont.

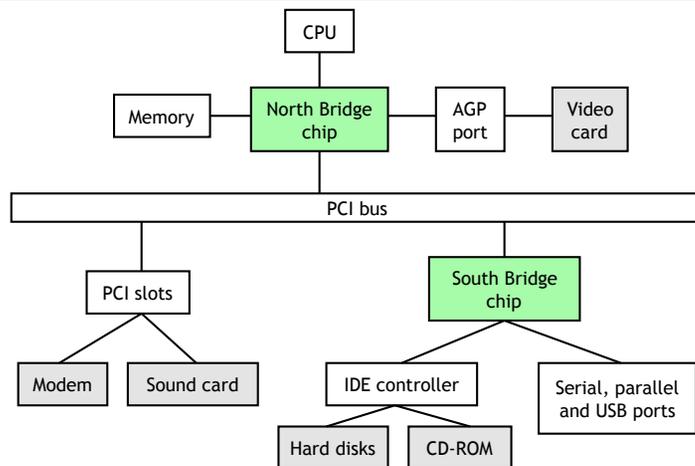
---

- 2) Assume the following system:
- A CPU and memory share a 32-bit bus running at 100MHz.
  - The memory needs 50ns to access a 64-bit value from one address.
- For this system, a single read can be performed in eight cycles or 80ns for an effective bandwidth of  $(12.5 \times 10^6 \text{ reads/second}) \times (8 \text{ bytes/read}) = 100\text{MB/s}$ .
- A) If the memory was widened, such that 128-bit values could be read in 50ns, what is the new effective bandwidth?
- B) What is the bus utilization (fraction of cycles the bus is used) to achieve the above bandwidth?
- C) If utilization were 100% (achievable by adding additional memories), what effective bandwidth would be achieved?

December 4, 2009

16

## Buses in modern PCs



December 4, 2009

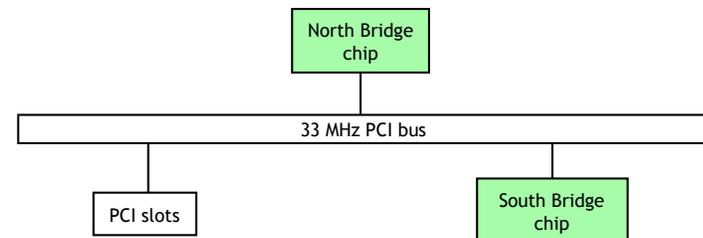
17

## PCI

- **Peripheral Component Interconnect** is a synchronous 32-bit bus running at 33MHz, although it can be extended to 64 bits and 66MHz.
- The **maximum bandwidth** is about 132 MB/s.

$$33 \text{ million transfers/second} \times 4 \text{ bytes/transfer} = 132\text{MB/s}$$

- Cards in the motherboard PCI slots plug directly into the PCI bus.
- Devices made for the older and slower ISA bus standard are connected via a “south bridge” controller chip, in a hierarchical manner.



December 4, 2009

18

## Frequencies

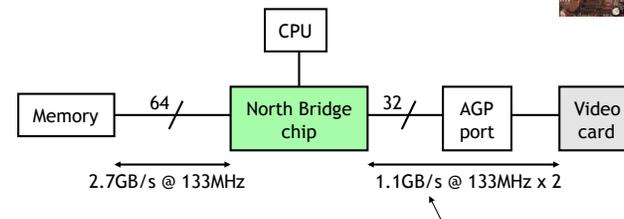
- CPUs actually operate at two frequencies.
  - The **internal frequency** is the clock rate inside the CPU, which is what we've been talking about so far.
  - The **external frequency** is the speed of the processor bus, which limits how fast the CPU can transfer data.
- The internal frequency is usually a multiple of the external bus speed.
  - A 2.167 GHz Athlon XP sits on a 166 MHz bus ( $166 \times 13$ ).
  - A 2.66 GHz Pentium 4 might use a 133 MHz bus ( $133 \times 20$ ).
    - You may have seen the Pentium 4's bus speed quoted at 533MHz. This is because the Pentium 4's bus is "**quad-pumped**", so that it transfers 4 data items every clock cycle.
- Processor and Memory data rates far exceed PCI's capabilities:
  - With an 8-byte wide "533 MHz" bus, the Pentium 4 achieves 4.3GB/s
  - A bank of 166MHz Double Data Rate (DDR-333) Memory achieves 2.7GB/s

December 4, 2009

19

## The North Bridge

- To achieve the necessary bandwidths, a "**frontside bus**" is often dedicated to the CPU and main memory.
  - "bus" is actually a bit of a misnomer as, in most systems, the interconnect consists of point-to-point links.
  - The video card, which also need significant bandwidth, is also given a direct link to memory via the Accelerated Graphics Port (AGP).
- All this CPU-memory traffic goes through the "north bridge" controller, which can get very hot (hence the little green heatsink).



December 4, 2009

20

## External buses

---

- **External buses** are provided to support the frequent plugging and un-plugging of devices
  - As a result their designs significantly differ from internal buses
- Two modern external buses, **Universal Serial Bus (USB)** and **FireWire**, have the following (desirable) characteristics:
  - **Plug-and-play** standards allow devices to be configured with software, instead of flipping switches or setting jumpers.
  - **Hot plugging** means that you don't have to turn off a machine to add or remove a peripheral.
  - The cable transmits **power!** No more power cables or extension cords.
  - **Serial links** are used, so the cable and connectors are small.



December 4, 2009

21

## Serial/Parallel

---

- Why are modern external buses **serial** rather than **parallel**?
- Generally, one would think that having more wires would increase bandwidth and reduce latency, right?
  - Yes, but only if they can be clocked at comparable frequencies.
- Two physical issues allow serial links to be clocked significantly faster:
  - On parallel interconnects, **interference** between the signal wires becomes a serious issue.
  - **Skew** is also a problem; all of the bits in a parallel transfer could arrive at slightly different times.
- Serial links are being increasingly considered for internal buses:
  - **Serial ATA** is a new standard for hard drive interconnects
  - **PCI-Express** (aka 3GI/O) is a PCI bus replacement that uses serial links

December 4, 2009

22