Announcements

- Lab 4 is out
  - Putting the pipeline to work!
- Homework 4 will be out soon
  - No, we did not forget about it ☺
- Pick up your midterm

Networks and Buses

- There are two main ingredients to I/O systems.
  - Devices like hard drives that we discussed last time.
  - Buses/Networks connect devices to each other and the processor.
    - Back of the envelope performance metrics ✓
    - Bus organization and Performance
    - Serial vs. Parallel

Networks (e.g., the Internet)

- When communicating over a network, typically your communication is broken into a collection of “packets”
  - Each packet carries ~1kB of data
  - Packets are reassembled into the original message at the destination.

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.washington.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
```

Back of the Envelope Calculation

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

$\underline{\text{Transfer time}}$ = latency + $\underline{\text{transfer\_size}}$ / $\underline{\text{bandwidth}}$

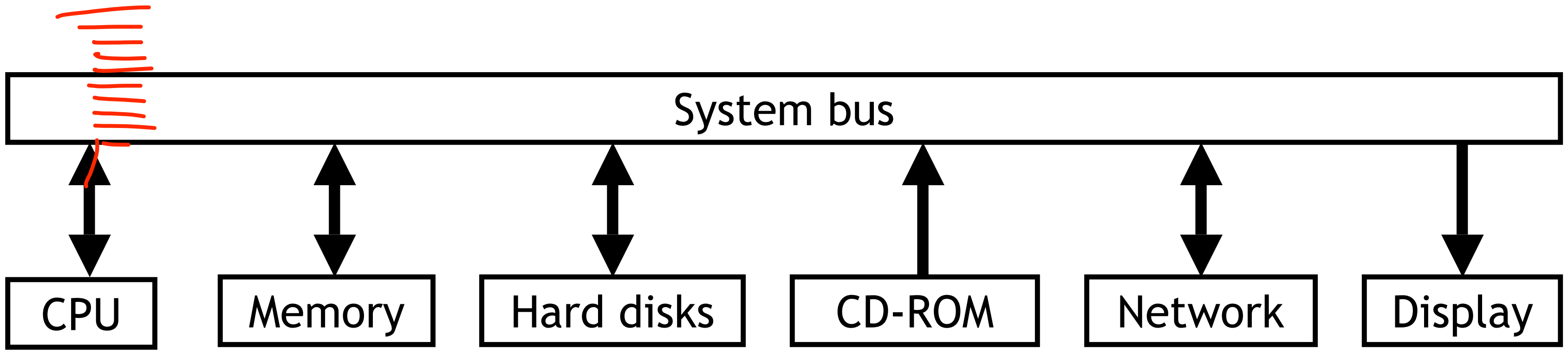
= time + bytes / (bytes/time)

Dominant term for small transfers

Dominant term for large transfers

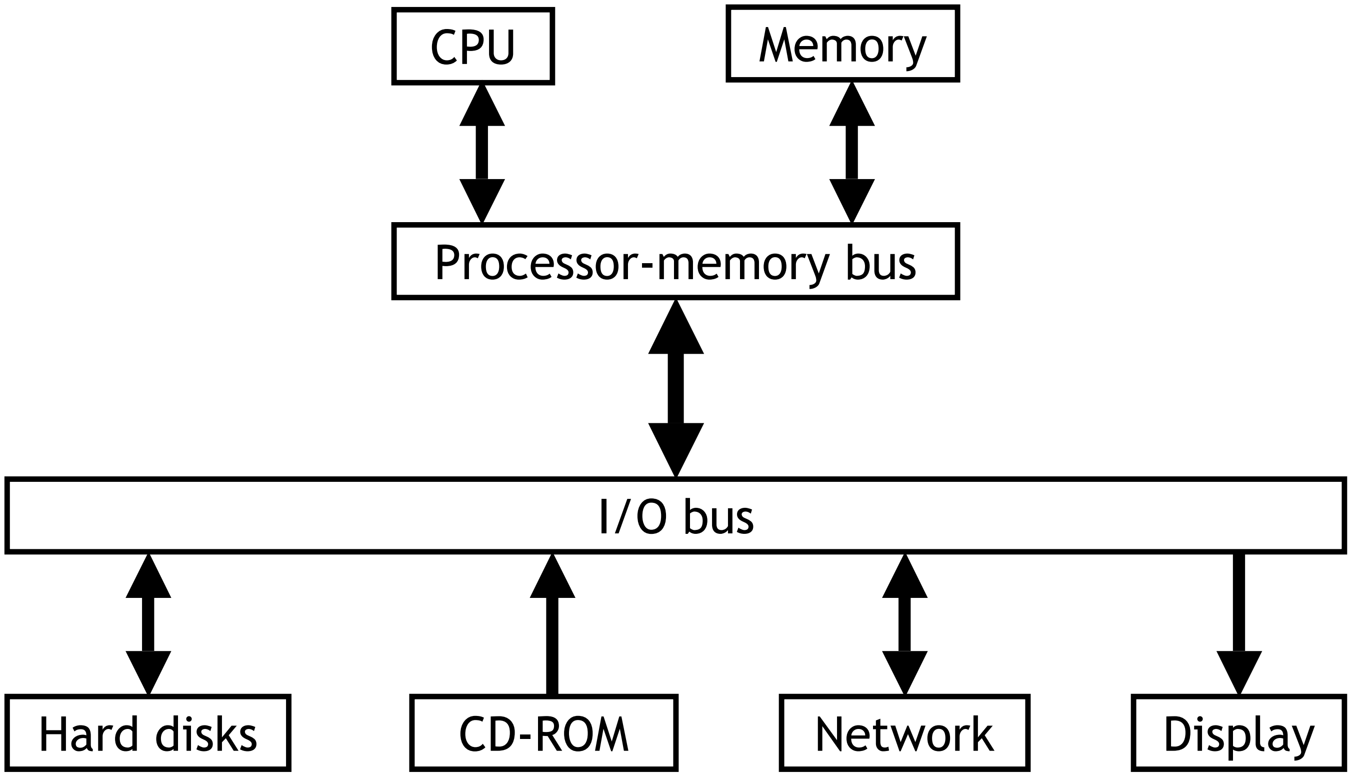
Computer buses

- 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.



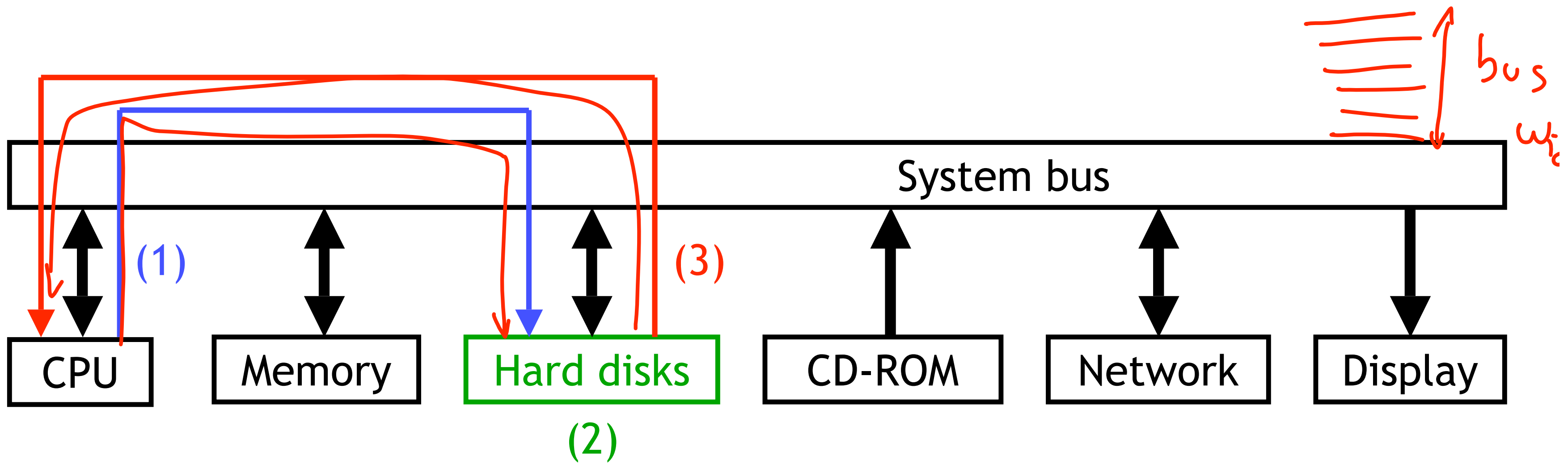
Hierarchical buses

- 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.



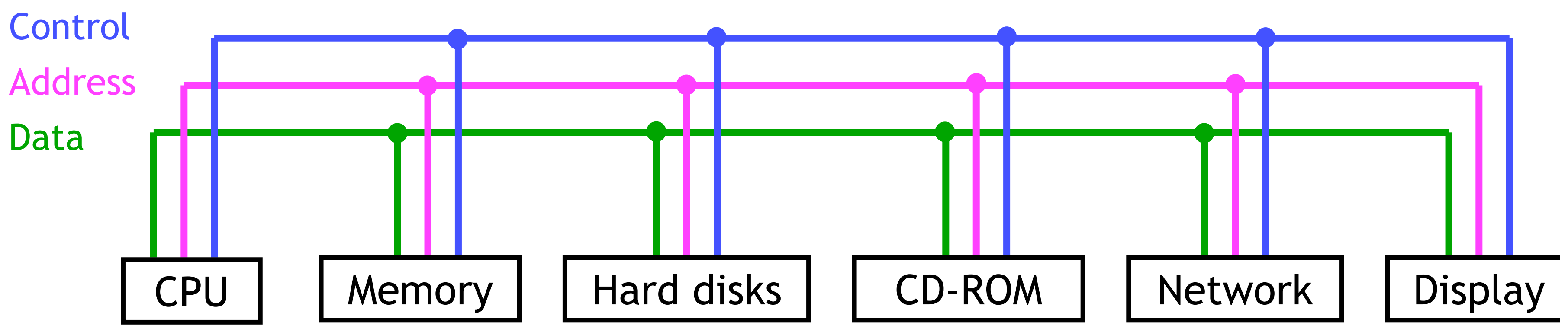
Basic bus protocols

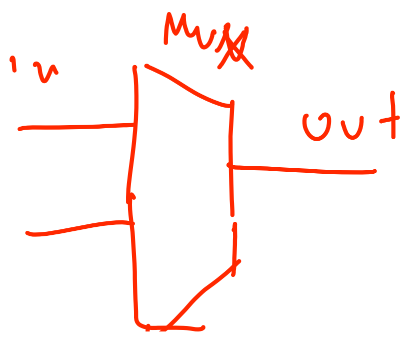
- Although physically, our computer may have a hierarchy of buses (for performance), logically it behaves like a single bus
- A few lectures ago we discussed how I/O reads and writes can be programmed like loads and stores, using addresses.
- Two devices might interact as follows.
  1. An initiator sends 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.

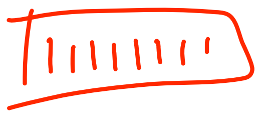


What is the bus anyway?

- 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 thi information can be sent in one clock cycle.

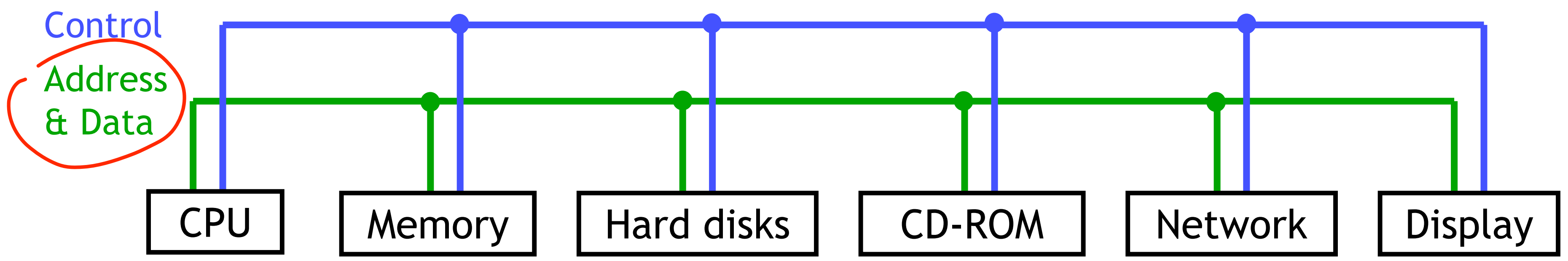






Multiplexed bus lines

- 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.



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. 10ns
  - 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. 1
  2. The memory needs 50ns, or 5 cycles, to read a 64-bit value. 5
  3. It takes two cycles to send 64 bits over a 32-bit wide bus. 2

$$\overline{8}$$

- 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 x 10^6 reads/second) x (8 bytes/read) = 100MB/s.

buff shen

Example Bus Problems, cont.

2) Assume the following system:

→ – A CPU and memory share a 32-bit bus running at 100MHz. 10ns

– 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 a effective bandwidth of (12.5 x 10^6 reads/second) x (8 bytes/read) = 100MB/s.

A) If the memory was widened, such that 128-bit values could be read in 50ns, what is the new effective bandwidth?

$1 + 5 + 4^{\text{4 xfer}} = 10_{\text{cyc}} \Rightarrow 100\text{ns}$ $10^7 \times 16 = 160\text{MB/s}$

B) What is the bus utilization (fraction of cycles the bus is used) to achieve the above bandwidth?

$\frac{1+4}{10} = \frac{1}{2} = 50\%$

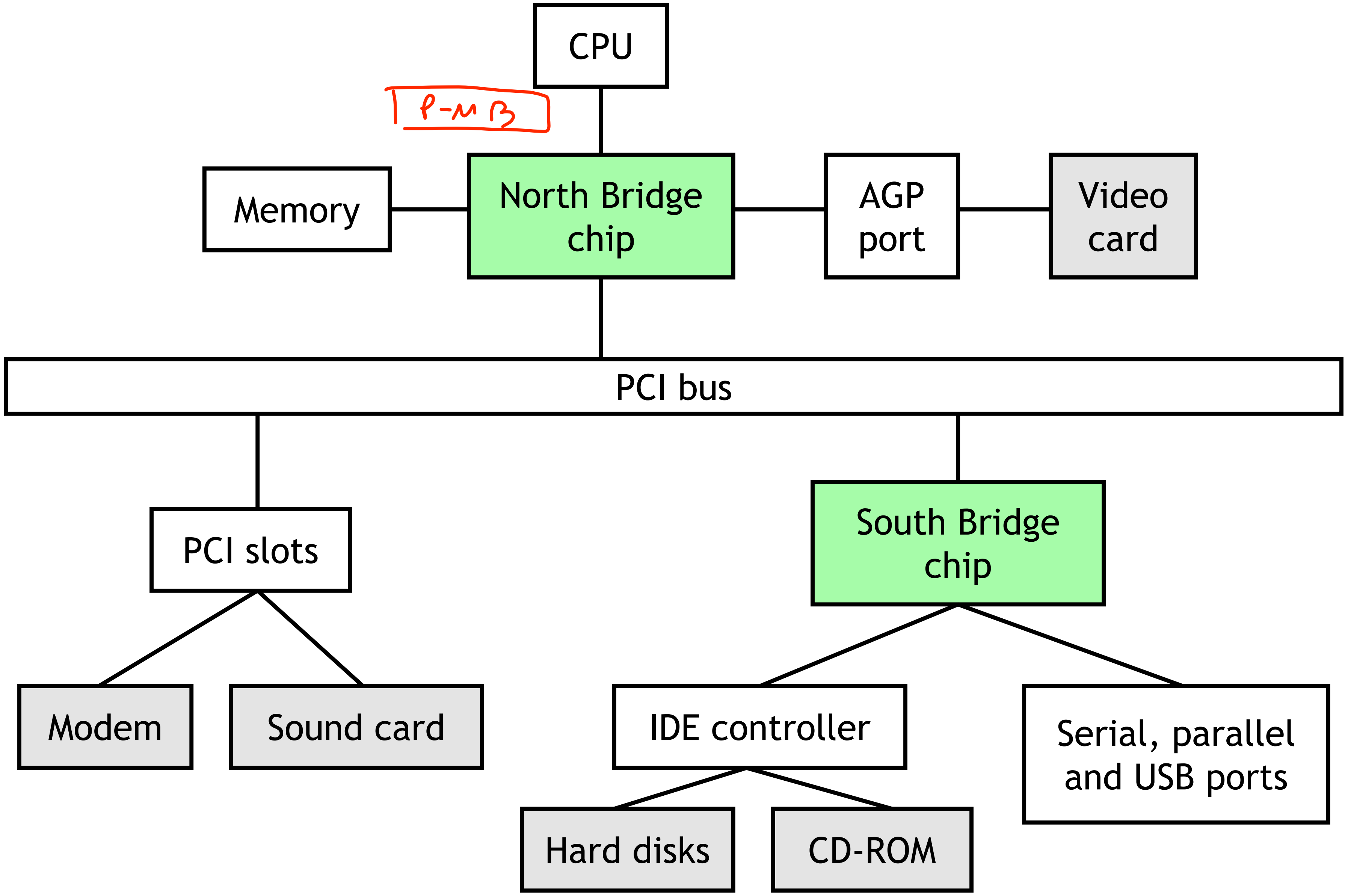
C) If utilization were 100% (achievable by adding additional memories), wha effective bandwidth would be achieved?

400 MB/s

Synchronous and asynchronous buses

- A synchronous bus operates with a central clock signal.
  – Bus transactions can be handled easily with finite state machines.
  – However, the clock rate and bus length are inversely proportional; faster clocks mean less time for data to travel. This is one reason why PCs never have more than about five expansion slots.
  – All devices on the bus must run at the same speed, even if they are capable of going faster.
- An asynchronous bus does not rely on clock signals.
  – Bus transactions rely on complicated handshaking protocols so each device can determine when other ones are available or ready.
  – On the other hand, the bus can be longer and individual devices can operate at different speeds.
  – Many external buses like USB and Firewire are asynchronous.

Buses in modern PCs

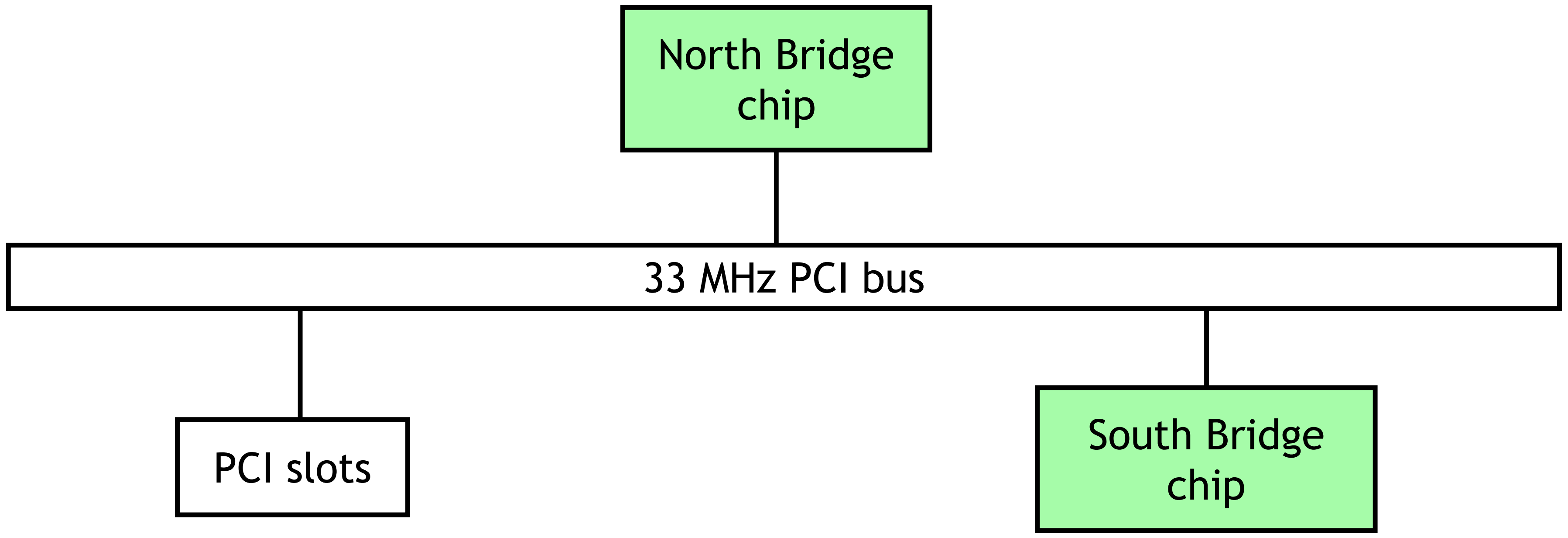


PCI

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

33 million transfers/second x 4 bytes/transfer = 132MB/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 vi a “south bridge” controller chip, in a hierarchical manner.



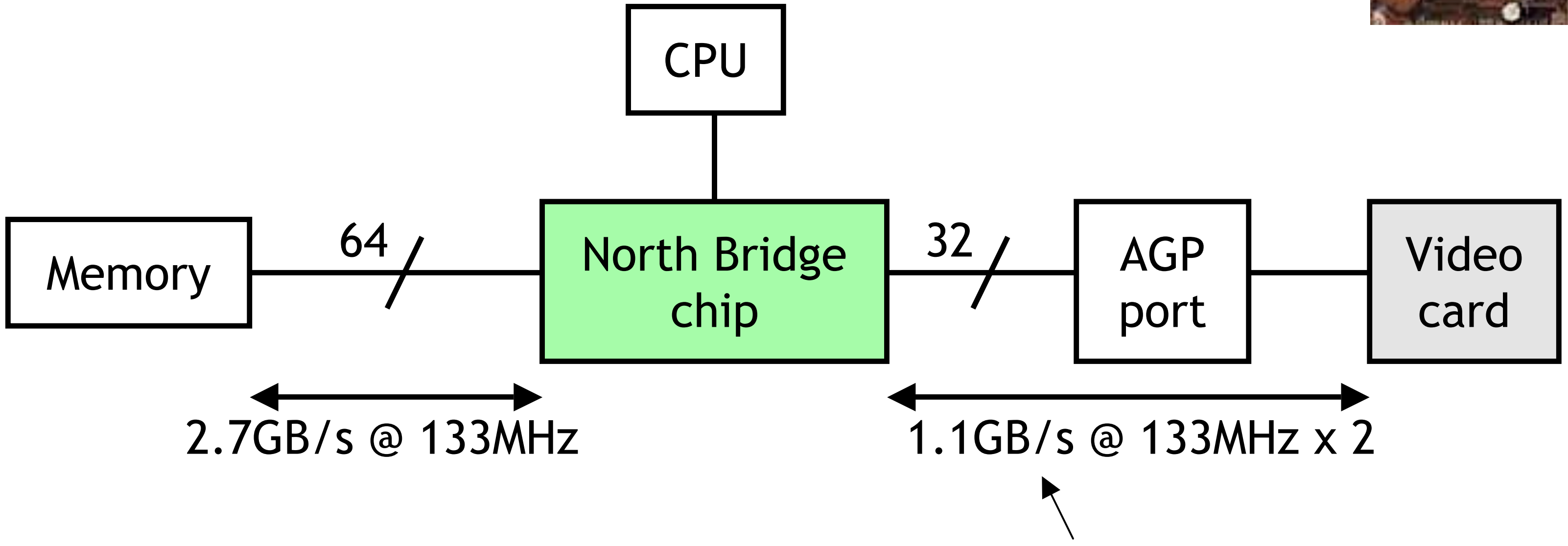
Frequencies

- 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 limit 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 x 13).
  - A 2.66 GHz Pentium 4 might use a 133 MHz bus (133 x 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 dat 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

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).



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, hav 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.

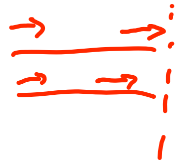
FireWire

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 link