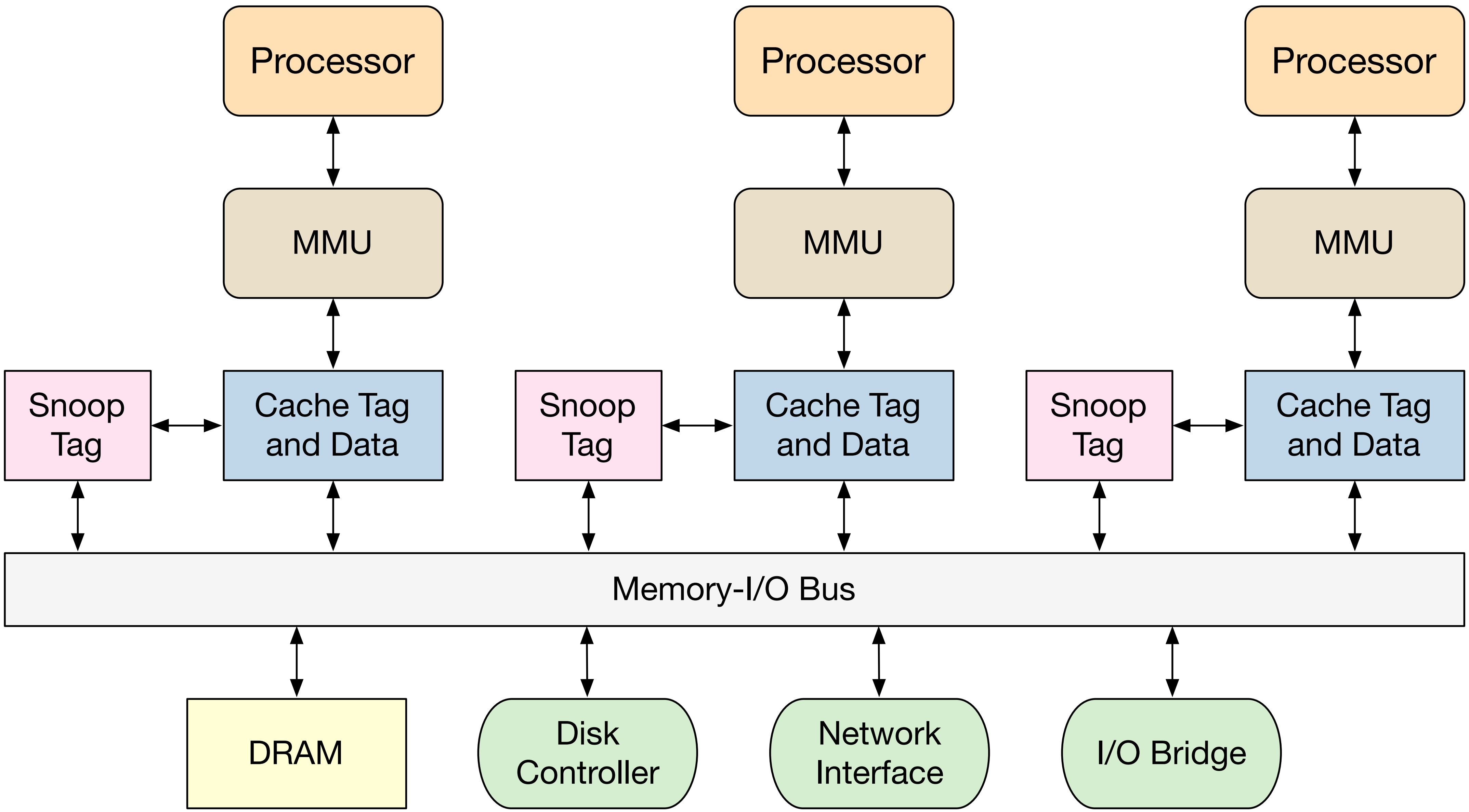
Lecture 24: Bus Interconnects

Fall 2023
Jason Tang

Topics

- Bus types
- Bus performance
- Bus arbitration

System Interconnect



- Bus: shared communication link, using a set of wires to connect multiple subsystems

Buses

- Consists of control and data lines
- Bus protocol: defines semantics of bus transactions and arbitrate usage
- As compared to a point-to-point interconnect, a bus is:
  - *More versatile*: easier to add new devices
  - *Lower cost*: single set of wires, shared in multiple ways
  - *Potential bottleneck*: limiting throughput when simultaneously communicating with multiple devices

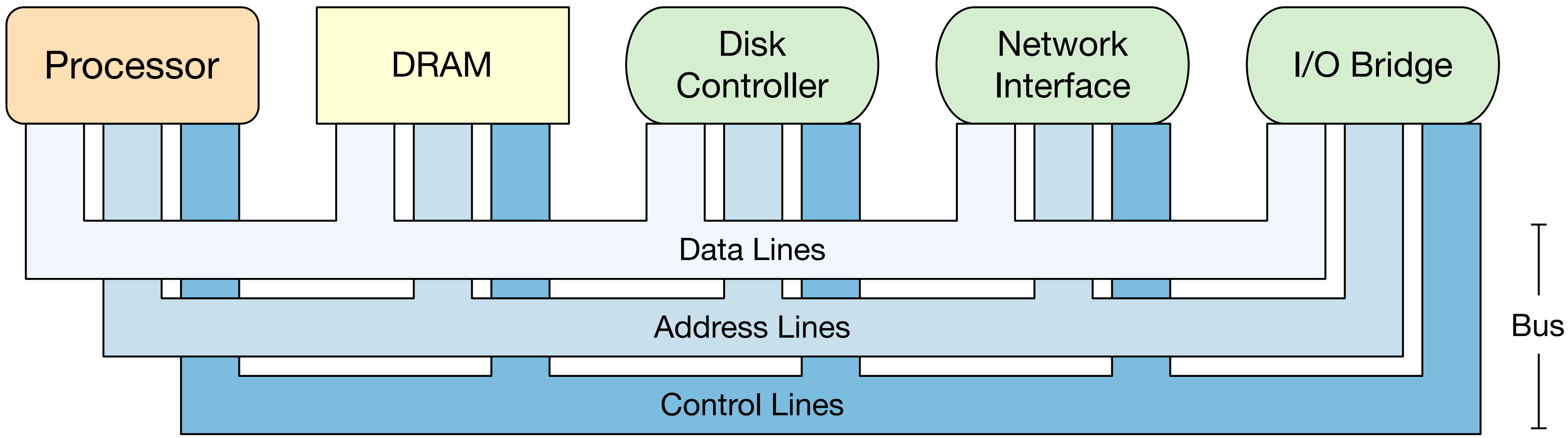
Bus Performance

- Bus performance can be measured using **throughput** and **response time**
    - Increasing bus bandwidth (throughput) can be achieved using buffering, which slows down bus access (response time)
- Maximum bus speed largely limited by physical factors
    - Length of bus
    - Number of devices

Bus Lines

- Data lines: used to move data
    - Width of data bus is number of lines
- Address lines (optional): used to specify an address in memory or I/O
    - Specifies bus or physical address, not virtual memory address
    - If not present, then devices must multiplex between sending data and address
- Control lines: used to determine direction of data flow, and when each device can access the bus
    - Read (from perspective of CPU), write (from perspective of CPU), interrupt request, clock, reset, power, ground, …

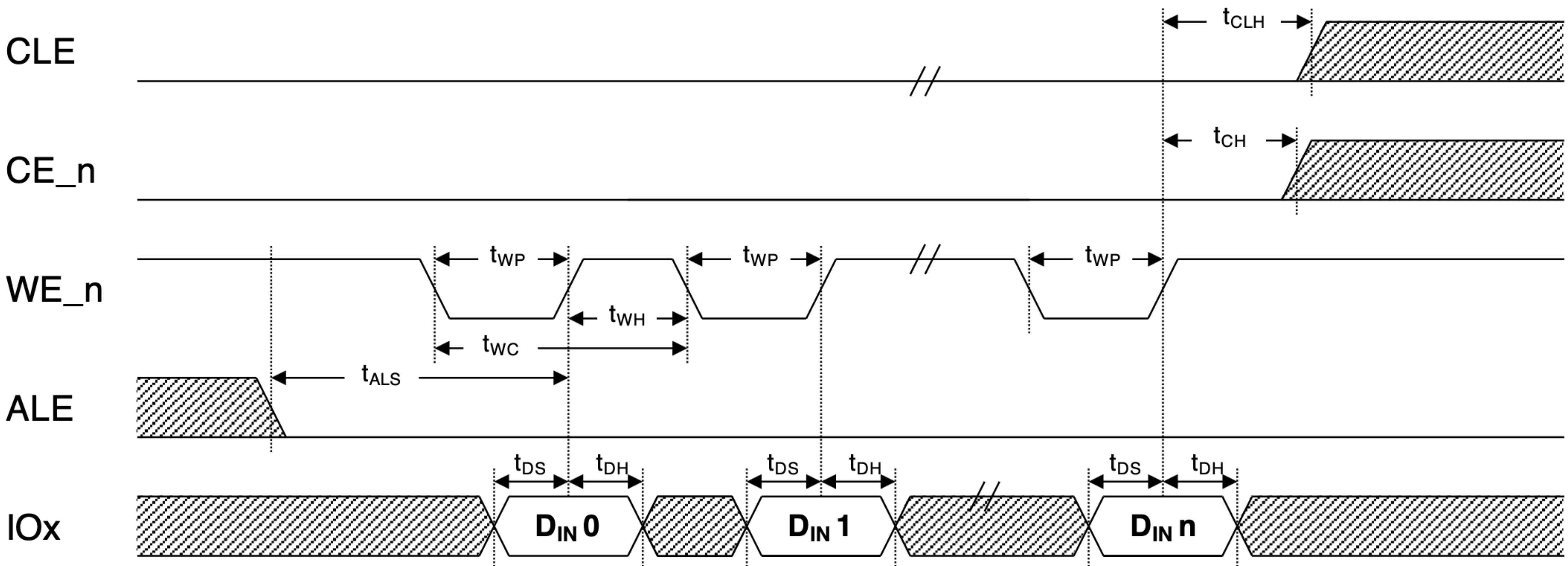
Bus Transactions



- On a read, processor sets address on address lines, and then asserts *read* control line; memory/device then sets data on data lines, and then asserts *data ready* control line

- On a write, processor sets data on data lines and sets address on address lines, and then asserts *write* control line; memory/device asserts *acknowledge* control line after it has consumed data

NAND Flash Timing Diagram



- CE_n is the chip enable line for device *n*
- CLE, WE_n, and ALE are unidirectional; IOx is bidirectional

Bus Types

- Processor-Memory ("Local") Bus: short and high speed to maximize bandwidth
- I/O Bus: lengthy to supportive multiple data rates and devices
  - Handle wide range of device latency, bandwidth, and characteristics
- Backplane Bus: connects memory, processor, and I/O devices
  - Can connect together multiple systems
- Local buses are design specific, while I/O and backplane buses are usually standardized

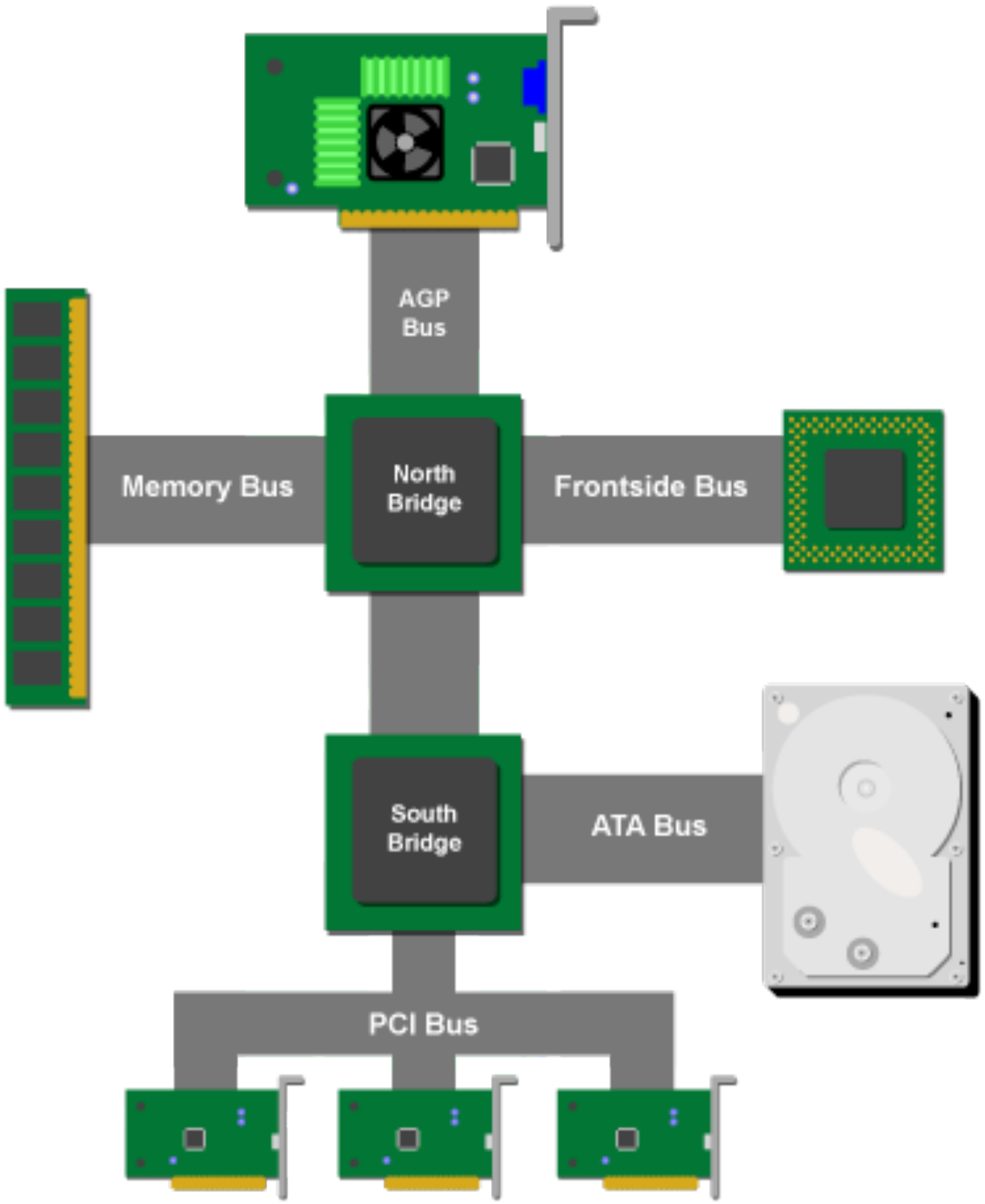
Local Bus

- Directly connects memory to CPU(s), and sometimes GPU(s)

- On Intel systems, northbridge (or "memory controller hub" (MCH)) connects CPU's frontside bus to memory and graphics bus

- Southbridge (or I/O controller hub) connects northbridge to slower devices

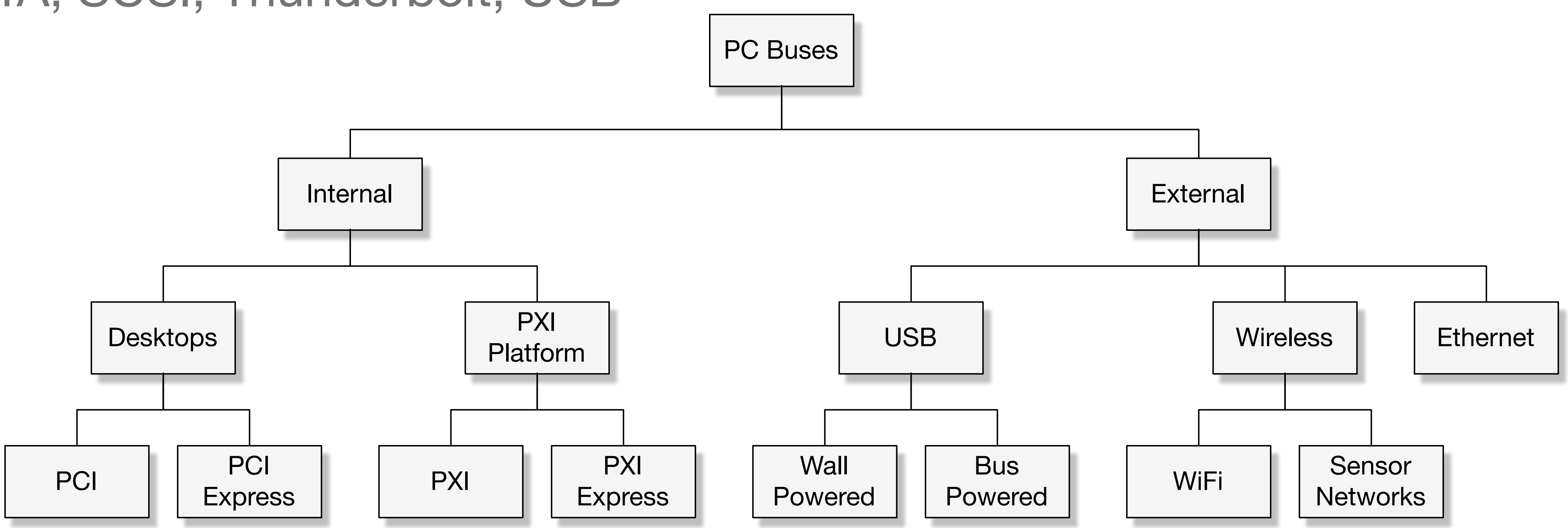
    - Additional I/O buses hang off of southbridge



https://arstechnica.com/technopaedia/2008/04/chipset/

I/O Bus

- Varying physical connections, depending upon needs and operating environment
    - Throughput, latency, size, weight, power, cost, …
- Common modern I/O buses: CAN, IEEE 1394, Fibre Channel, MIDI, PCI, SATA, SCSI, Thunderbolt, USB



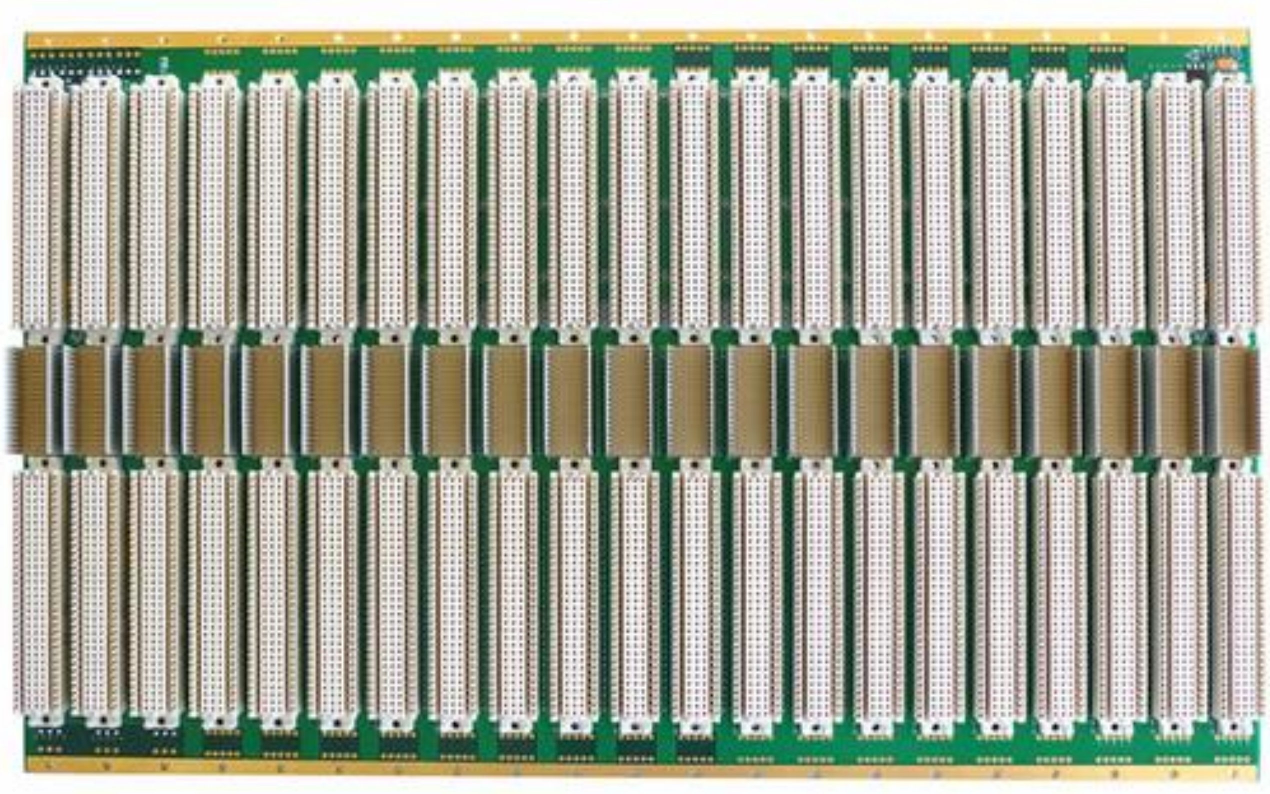
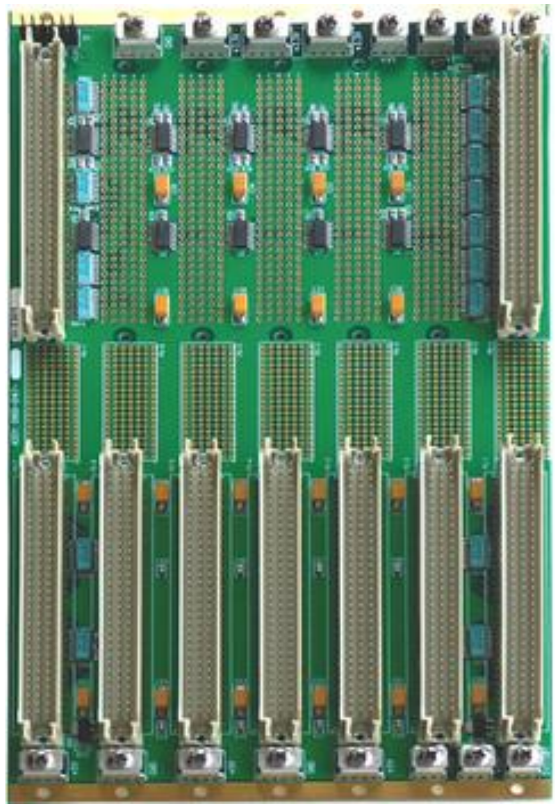
http://www.ni.com/white-paper/9401/en/

Backplane Bus

- Parallel wires, such that each pin of each connector is linked to relative pin of all other connectors

- Compute nodes connect to a backplane connector, to create a full system

    - Each node can send data to any other node on the backplane

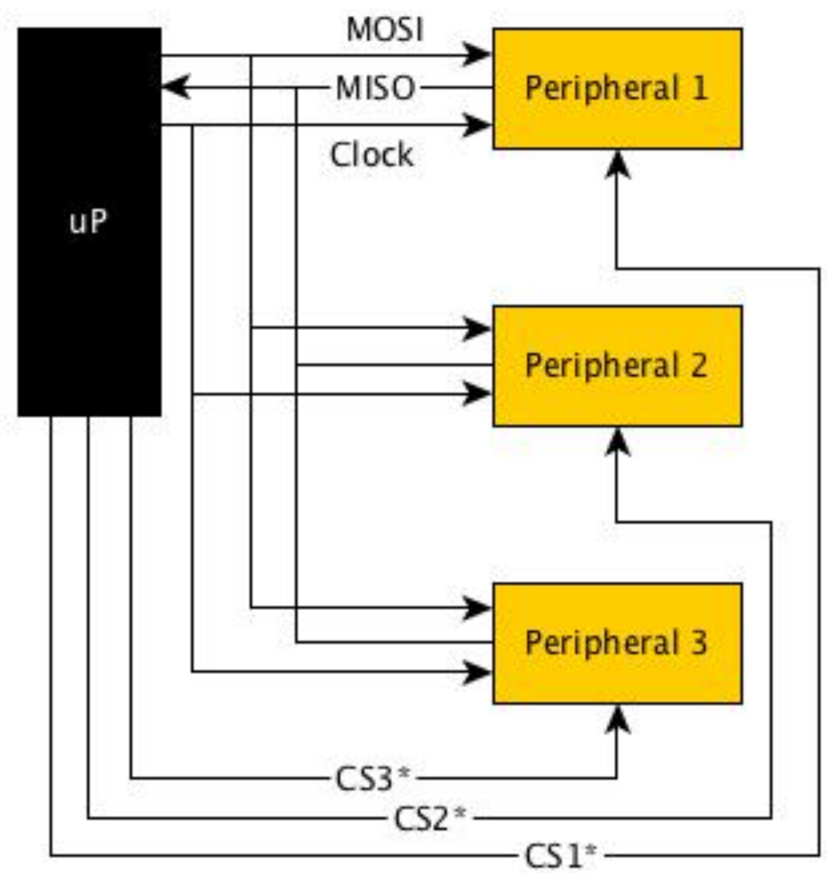



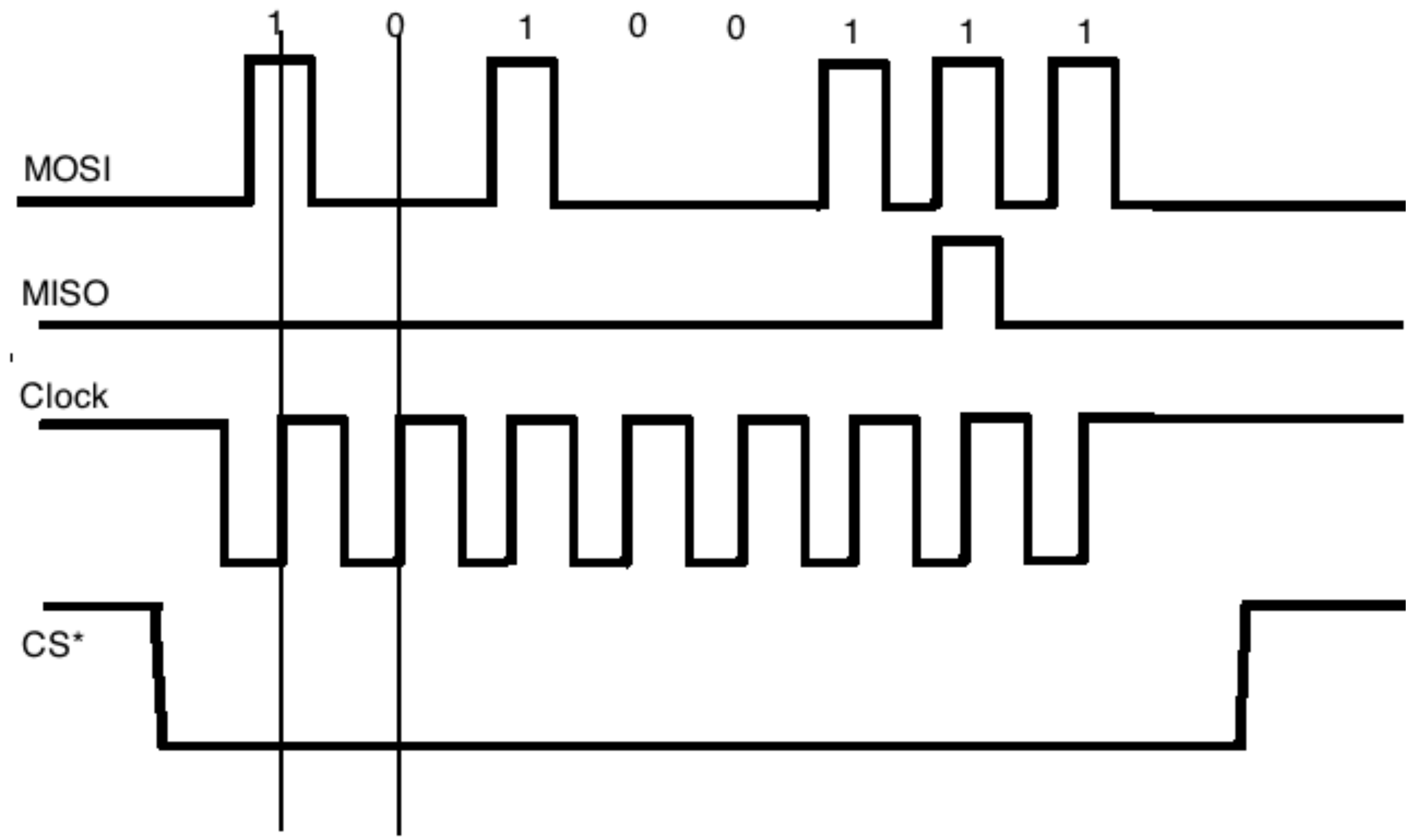
https://www.vectorelect.com/backplanes-vme64x.html

Synchronous vs. Asynchronous

- Synchronous Bus: clock-based protocol and time-based control lines
  - Master clock broadcasts a signal to all devices; all devices are synchronized to those clock pulses (via internal phase-lock loops)
  - Subject to clock skew
  - Local buses are often synchronous
- Asynchronous Bus: devices have their own clocks
  - Handshaking protocol to coordinate data signals

SPI Clocking Diagram





- All signals are unidirectional:
  - MOSI, Clock, and Chip Select are from Controller to Peripheral
  - MISO is from Peripheral to Controller

https://www.embedded.fm/blog/2017/3/24/i-spi

Bus Performance Example

- A bus has a clock cycle time of 50 ns, with each transaction taking 1 clock cycle. An asynchronous bus requires 40 ns per handshake. The data port of both buses is 32-bit wide. What is the bandwidth of each bus for one-word reads from 200 ns memory?

- For synchronous bus, total time is 300 ns (50 ns to send address, 200 ns to read memory, then 50 ns to return data)

    - 4 bytes every 300 ns = 13.3 MB / sec

- For asynchronous bus, total time is 360 ns (40 ns for RRQ, 200 ns to read memory, 40 ns for DataReady, 40 ns for DataAck, 40 ns to drop DataReady)

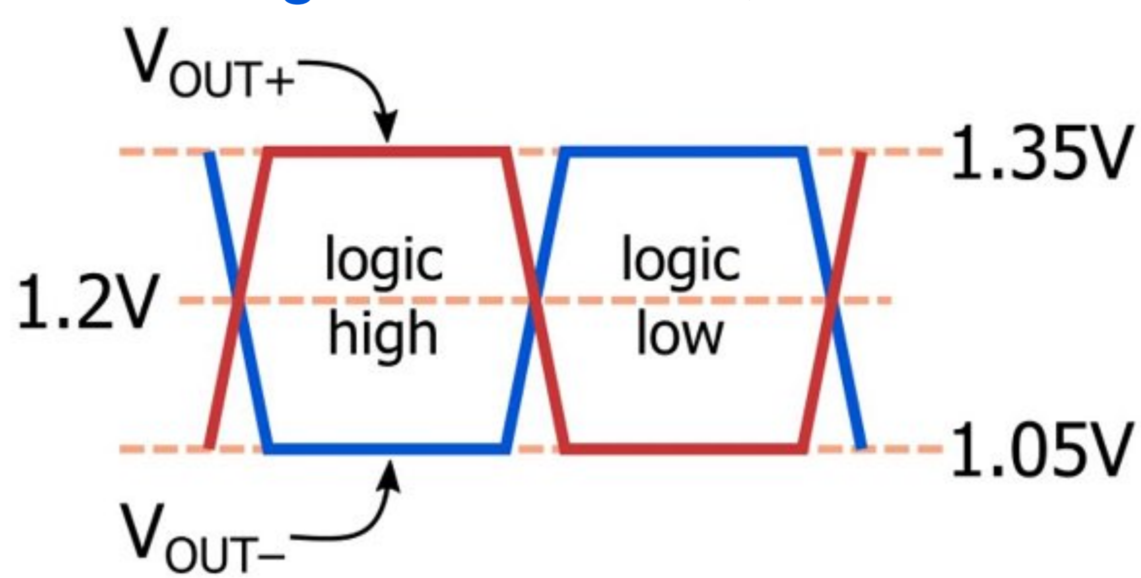
    - 4 bytes every 360 ns = 11.1 MB / sec

Increasing Bus Bandwidth

- Data bus width: transfer multiple words with fewer cycles
  - Increases number of bus lines
- When writing, send address and data simultaneously with separate lines
  - Simplifies bus control logic, but increases number of bus lines
- Burst Write: write multiple words, in successive cycles, without sending a new address for each word
  - Increases complexity of bus control logic

Serial Buses

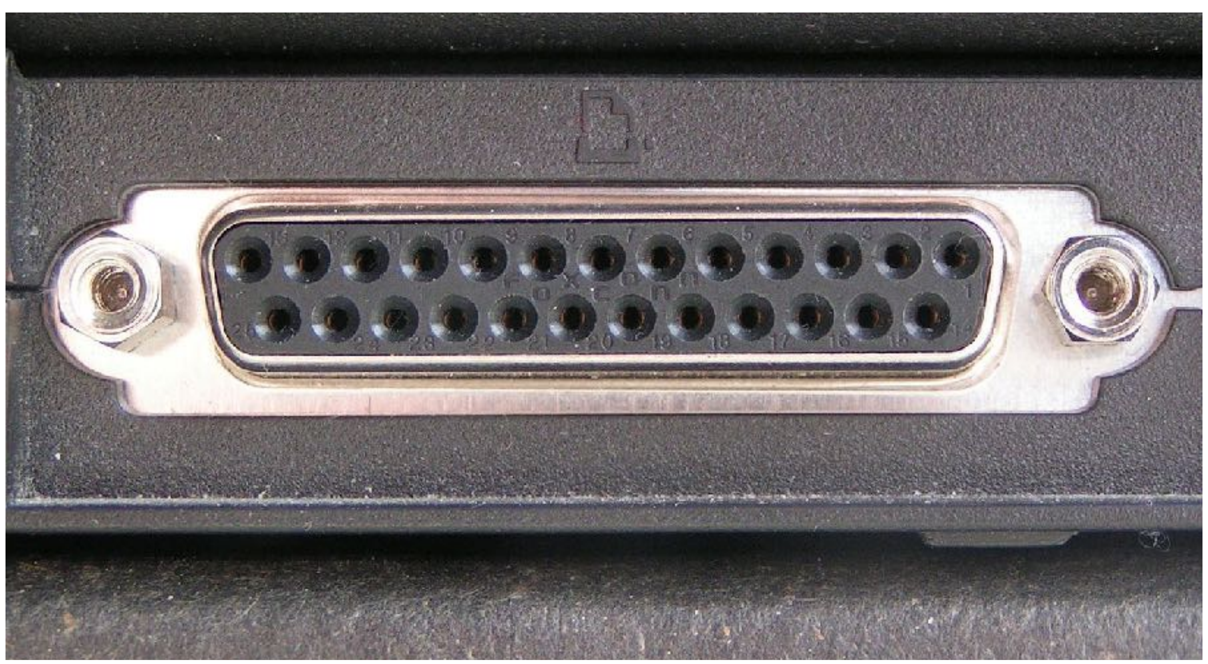
- Serial bus: only one data line, and thus sends only one bit at a time
- For a serial line, a voltage above a certain threshold (often 1.8 volts) is considered a one, and voltage below that threshold is a zero
- Single lines are susceptible to static, interference, etc
  - Low voltage differential signal: two wires, where second wire mirrors first

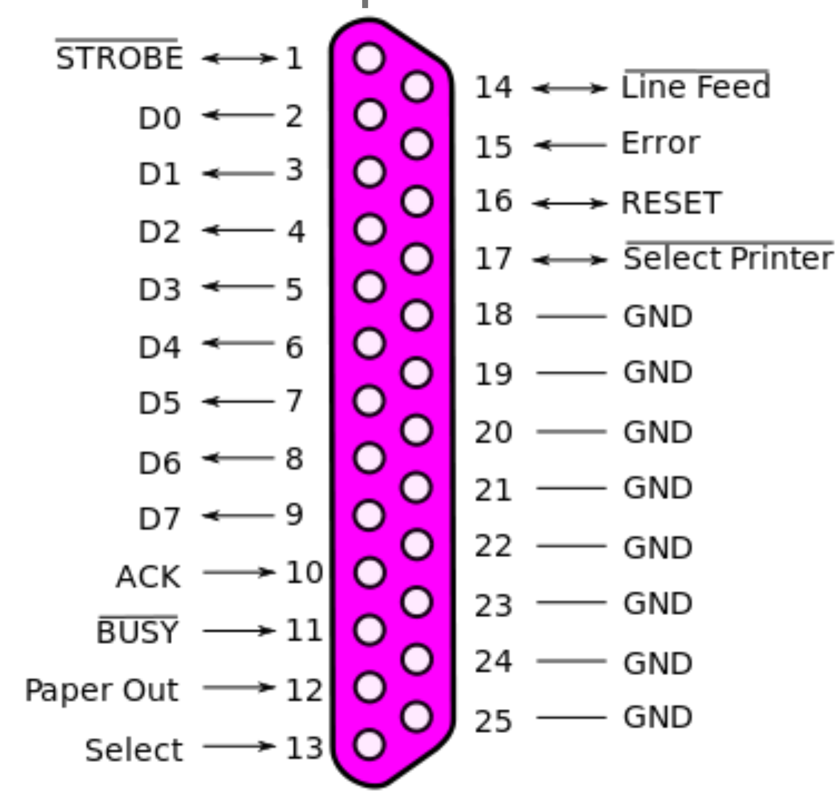


https://www.allaboutcircuits.com/technical-articles/lvds-high-speed-low-power-robust-data-transfer/

Parallel Buses

- Parallel bus: multiple data lines, to send multiple bits simultaneously
  - Common parallel buses have 8, 32, or more data lines
- Simple way to increase bus performance, on older computers





https://en.wikipedia.org/wiki/Parallel_port

Serial vs Parallel Buses

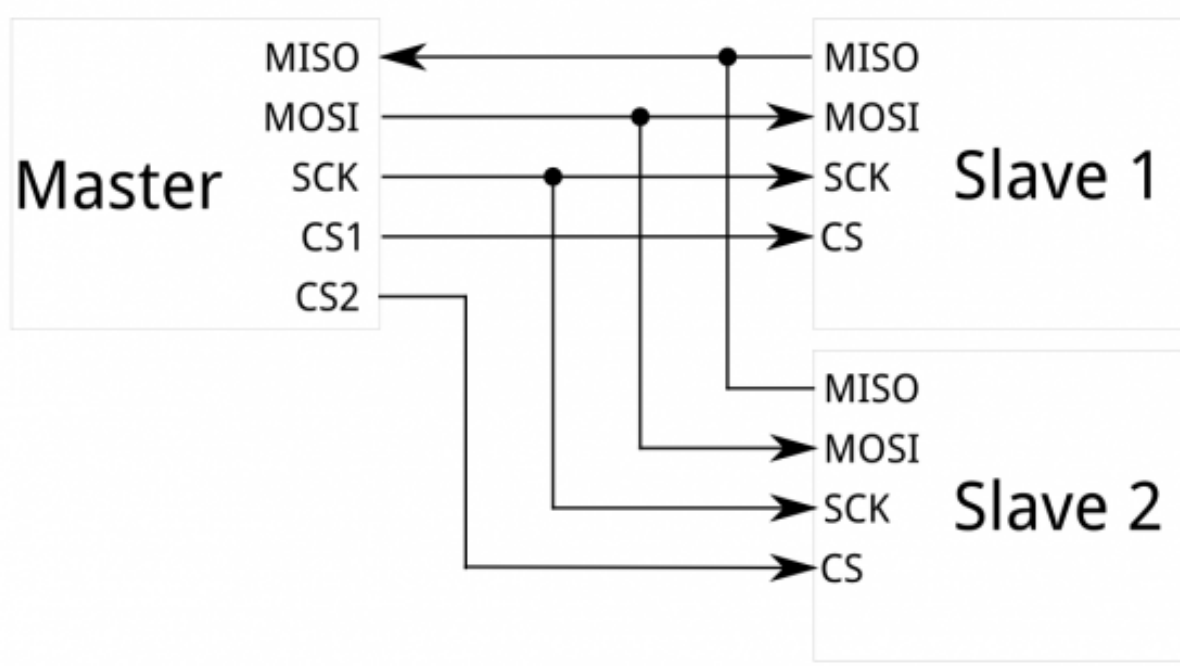
- For the same clock frequency, serial buses have *lower* bus bandwidth versus parallel buses
- But as clock frequencies increase, parallel buses perform *worse*
    - Crosstalk: interference between data lines, especially unshielded wires
    - Synchronization: signals from one wire arrive faster than another wire, because of differences of wire length
- Modern trend is to move from parallel to serial (PATA → SATA, PCI → PCI Express)

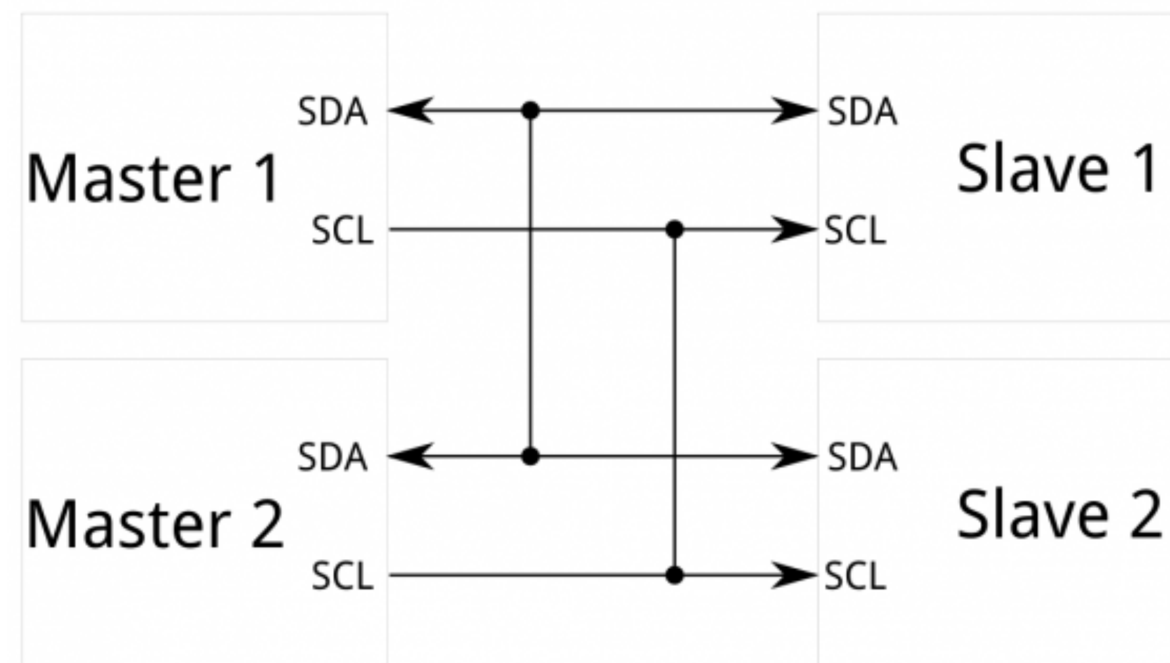
https://www.howtogeek.com/171947/why-is-serial-data-transmission-faster-than-parallel-data-transmission/

Bus Access

- When multiple devices are sharing bus lines, need a way to decide which device gets to write to bus, otherwise bus contention occurs
  - Bus master: controls access to the bus and initiates and manage all bus requests
  - Bus slave: can only respond to a request, but never initiates on its own
- If processor is the only bus master, then it is required for every transaction (but peripherals cannot use DMA between themselves or to the processor)
- If a bus has multiple bus masters, then there must be arbitration to coordinate bus usage among potential access requesters

Bus Access





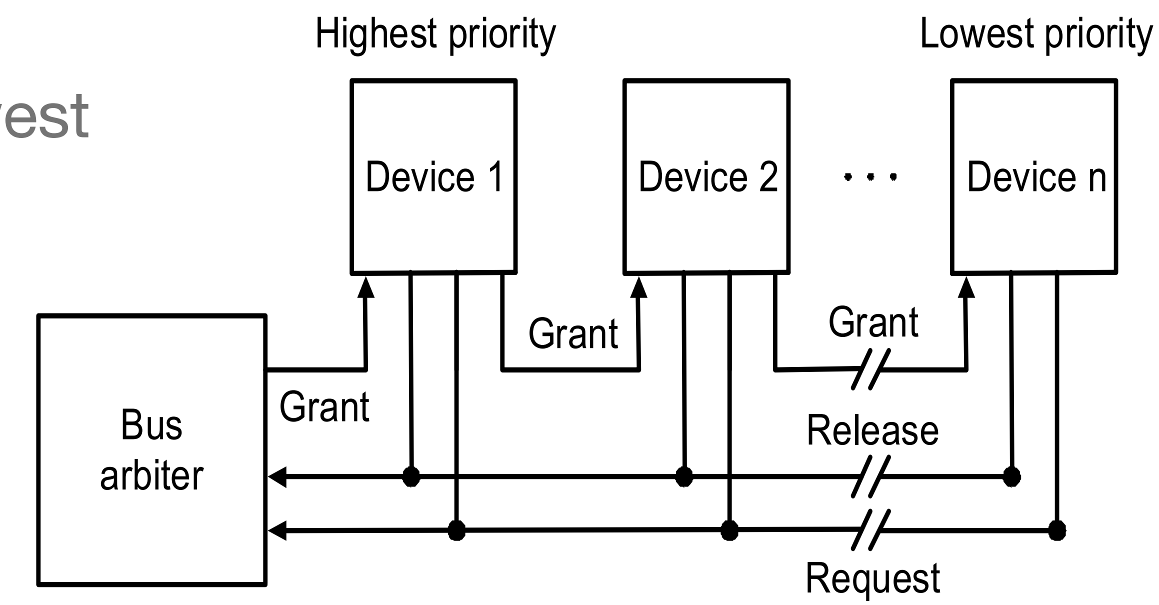
- Serial Peripheral Interface (SPI):
    - Full duplex communication
    - Single master initiates all traffic; slaves cannot send data between themselves
- Inter-Integrated Circuit (I^2C):
    - Half duplex communication
    - Multiple masters, arbitrated based upon which device first pulls SDA low

Bus Arbitration

- Bus arbitration coordinates bus usage among multiple devices using *request*, *grant*, *release* mechanism
- When multiple devices contend to be the master:
  - Devices with higher priority should be serviced first
  - But need to avoid starving lower priority devices
- Can either arbitrate by self-selection (such at I^2C) or by collision detection (such as Ethernet)
  - Example: for an Ethernet collision, each device delays a *random* amount of time before resending

Bus Arbitration

- Daisy chain arbitration: bus grant line runs through devices from highest priority to lowest



    - High priority devices simply intercept grant signal, to prevent lower priority device from seeing signal
    - Limited bus speed (grant signals take long time to reach last device in the chain)

- Centralized, parallel arbitration: multiple request lines, one per device
    - Centralized arbiter selects a device and notifies it to be the next bus master
    - Arbiter can be a bottleneck for bus usage