CMPE110 Lecture 24
Input/Output (I/O)

Heiner Litz

https://canvas.ucsc.edu/courses/19290

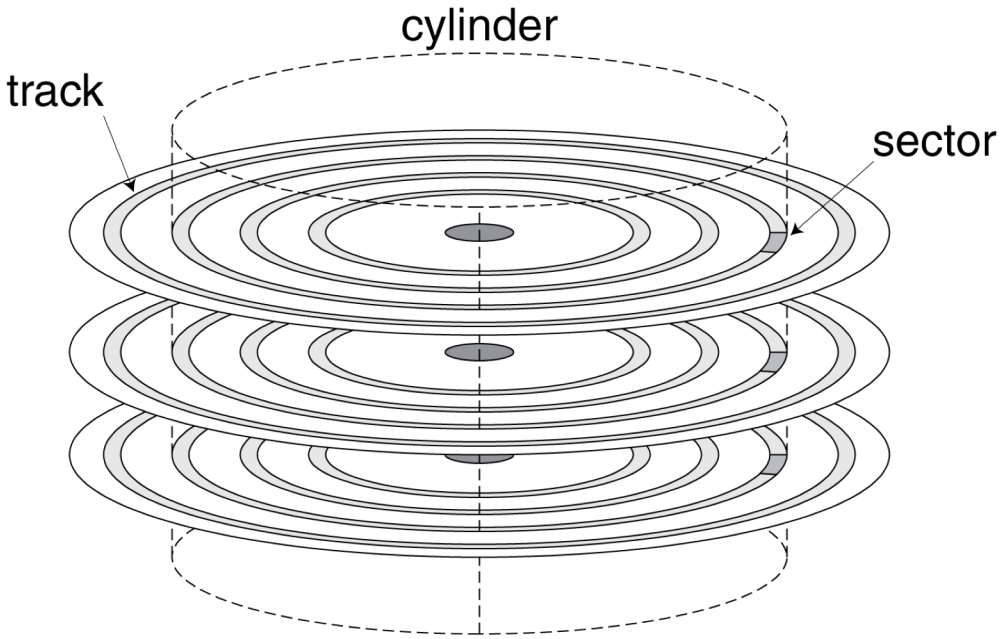
Announcements

- No Quiz on Friday, instead evaluations
- This week is exam relevant

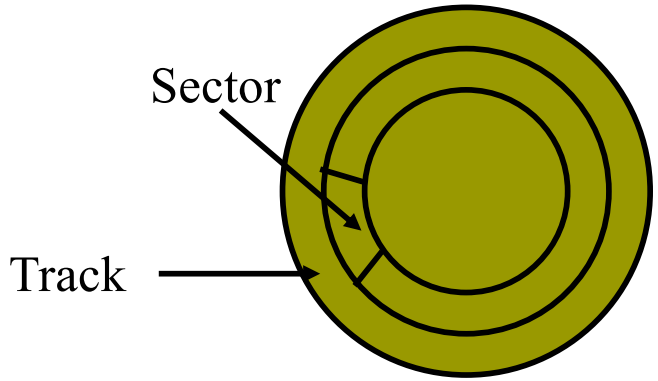
Review

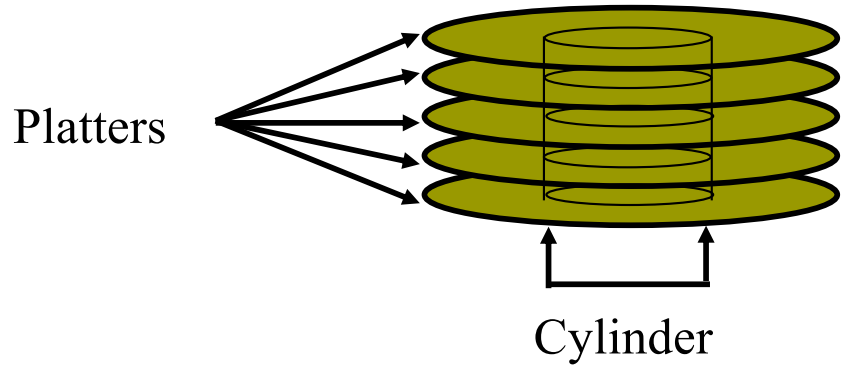
Magnetic Hard Disks

- Nonvolatile, rotating magnetic storage



Hard Disk Organization





- Each sector records
  - Sector ID and data (512 bytes, 4KB proposed)
  - Error correcting code (ECC), synchronization fields and gaps
- Access to a sector involves
  - Queuing delay if other accesses are pending
  - Seek: move the heads
  - Rotational latency
  - Data transfer
  - Controller overheads

Disk Performance Example

- Given
  - 512B sector, 15,000rpm, 4ms average seek time, 100MB/s transfer rate, 0.2ms controller overhead, idle disk
- Average read time
  - 4ms seek time
    + ½ / (15,000/60) = 2ms rotational latency
    + 512 / 100MB/s = 0.005ms transfer time
    + 0.2ms controller delay
    = 6.2ms
- If actual average seek time is 1ms
  - Average read time = 3.2ms

Disk Performance Issues

- Manufacturers quote average seek time
  - Based on all possible seeks
  - Locality and OS scheduling lead to smaller actual average se
- Smart disk controller allocate physical sectors on disk
  - Present logical sector interface to host
  - SCSI, ATA, SATA
- Disk drives include caches
  - Prefetch sectors in anticipation of access
  - Avoid seek and rotational delay

Questions

- How can you minimize seek time?
  - Tip: assume a full queue of pending accesses
- How can you amortize the seek time?
  - Tip: can locality help?

HW/SW Interface for I/O

- Need to communicate control & data to/from device

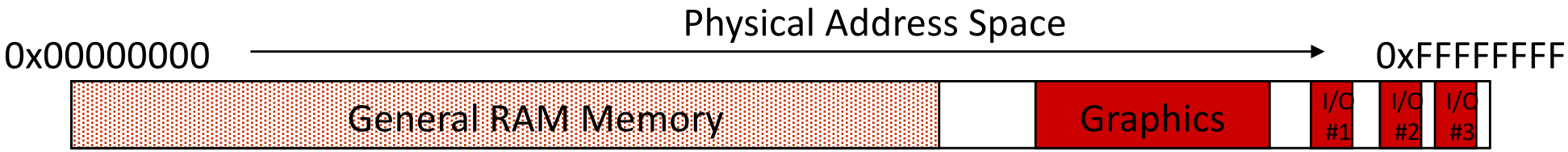
- Common approach
  - Build a design interface using registers & buffers
  - Memory-mapped I/O
    - With some help from virtual memory

Example Register Interface: Serial I/O Controller

- Data registers
  - TxD and RxD
  - Read/write may trigger action/side effects
- Command/status registers
  - Bits for status
    - TxRdy, RxRdy
  - Bits to record errors
    - ECC error, overrun, framing error, etc…
  - Bits to control response
    - Interrupt enables
  - Bits to configure
    - Data rate, framing, etc…

Memory Mapped I/O

- Memory-mapped I/O basics
  - Portions physical address space are assigned to each I/O device
  - I/O addresses correspond to device registers and buffers
  - Read/write to registers to control device
  - User programs prevented form issuing I/O operations directly since I/O address space is protected by the address translation mechanism



Question

- How does software make sense of all these different device interfaces?

OS and Types of Devices

- Abstract I/O devices to fit a couple of standard types
  - No need to rewrite programs for every new device
- Block device (e.g., disks)
  - Seek location, read/write a block of data
  - lseek(fp, offset, reference),
  - read(fp, buf, nbytes), write(fp, buf, nbytes),
- Stream device (e.g., audio I/O, network)
  - No addressing, just read/write next data block, queue interface
  - read(fp, buf, nbytes), write(fp, buf, nbytes),

Device Drivers

- Low level code that exports block/stream interface
  - Often provided by the I/O device vendor
  - Hides implementation details from OS/programs
  - Translates seek(), read(), and write() to low level actions
    - Read/write memory-mapped registers
- Details sometimes leak through abstraction
  - ioctl(fp, code, arg)
  - Used in performance profiling code in Labs (counter interface)

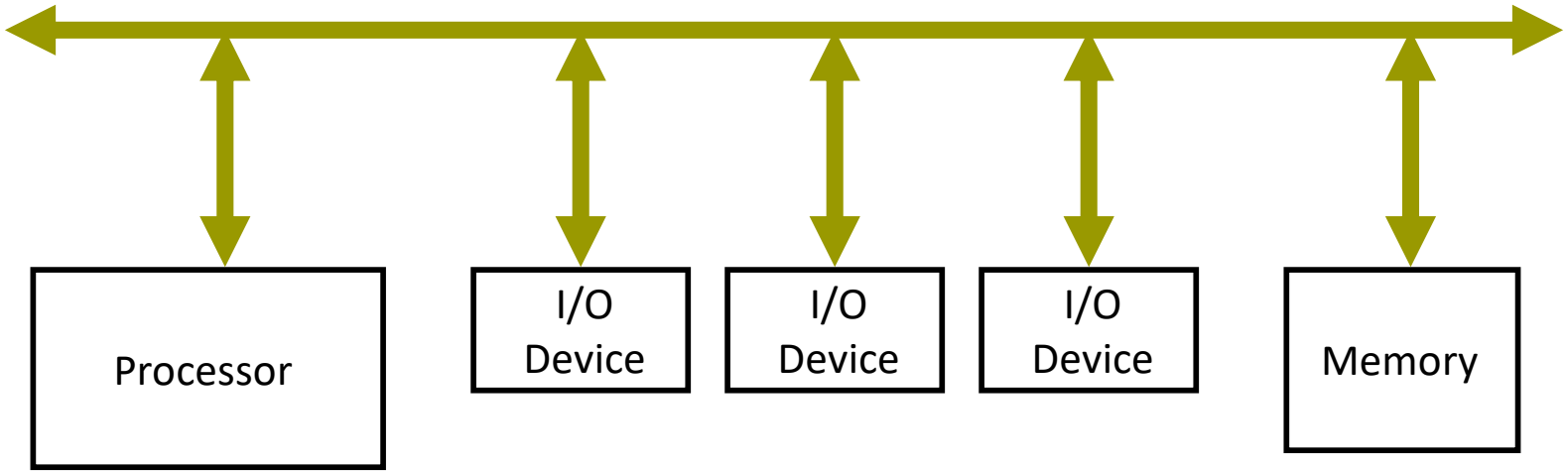
Physical Interface for I/O

- How do we connect cores, memories, and I/O devices physically?

- Common solutions
  - Buses
  - Point-to-point links & switch-based interconnects

Bus

- A shared communication link that connects multiple devices
  - Single set of wires connects multiple “subsystems”
  - As opposed to a point to point link which only connects two components
- Wires connect in parallel
  - E.g., 32 bit bus has 32 wires of data
  - And a few additional wires for control



Bus Transactions

- An initiator acquires the bus
  - May require arbitration
- Initiator starts a transaction
    - E.g., the CPU reads or writes a word to memory
  - Specifies command (read, write, reset, etc…)
  - Specifies address (which memory, device/register to access)
  - Specifies outbound data for a write
- Target responds to the transaction
  - Does the specified action (e.g., reads or writes the data)
  - Responds (possibly with data – for a read)

Bus Components

- Control Lines
  - Signal begin and end of transactions
  - Indicate the type of information on the data line
- Data Lines
  - Carry information between source and destination
  - Can include data, addresses, or complex commands

Generic Types of Buses

- Processor-memory bus (processor-processor bus)
  - Short, high-speed bus
  - Connects memory and processor directly
  - Designed to match the memory system
    - Achieve the maximum bandwidth for cache line transfers
  - Designed for a given processor/memory system (proprietary)

- I/O bus (or peripheral bus)
  - Usually long and slow
  - Connect devices to the processor-memory bus
  - Must match a wide range of I/O device performance characteristics
  - Industry standard

Accessing the Bus

- How is the bus reserved by a device that wishes to use it?
- Master-slave arrangement
  - Only the bus master can control access to the bus
  - The bus master initiates and controls all bus requests
  - A slave responds to read and write requests
- A simple system
  - Processor is the only bus master
  - All bus requests must be controlled by the processor
  - Major drawback is the processor must be involved in every transfer

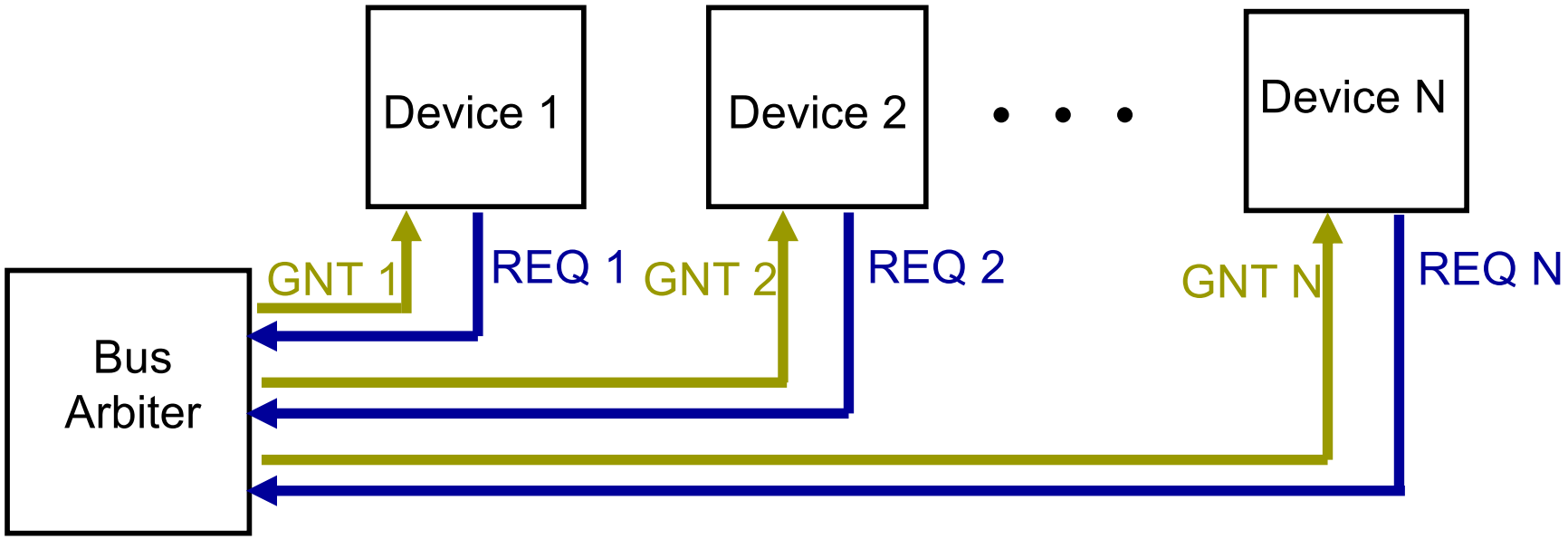
Multiple Masters

- With multiple masters, arbitration must be used so that only one device is granted access to the bus at a given time

- Arbitration
  - The bus master wanting to use the bus asserts a bus request
  - The bus master cannot use the bus until the request is granted
  - The bus master must signal the arbiter when finished using the bus

- Bus arbitration goals
  - Bus priority – Highest priority device should be serviced first
  - Fairness – Lowest priority devices should not starve

Centralized Parallel Arbitration

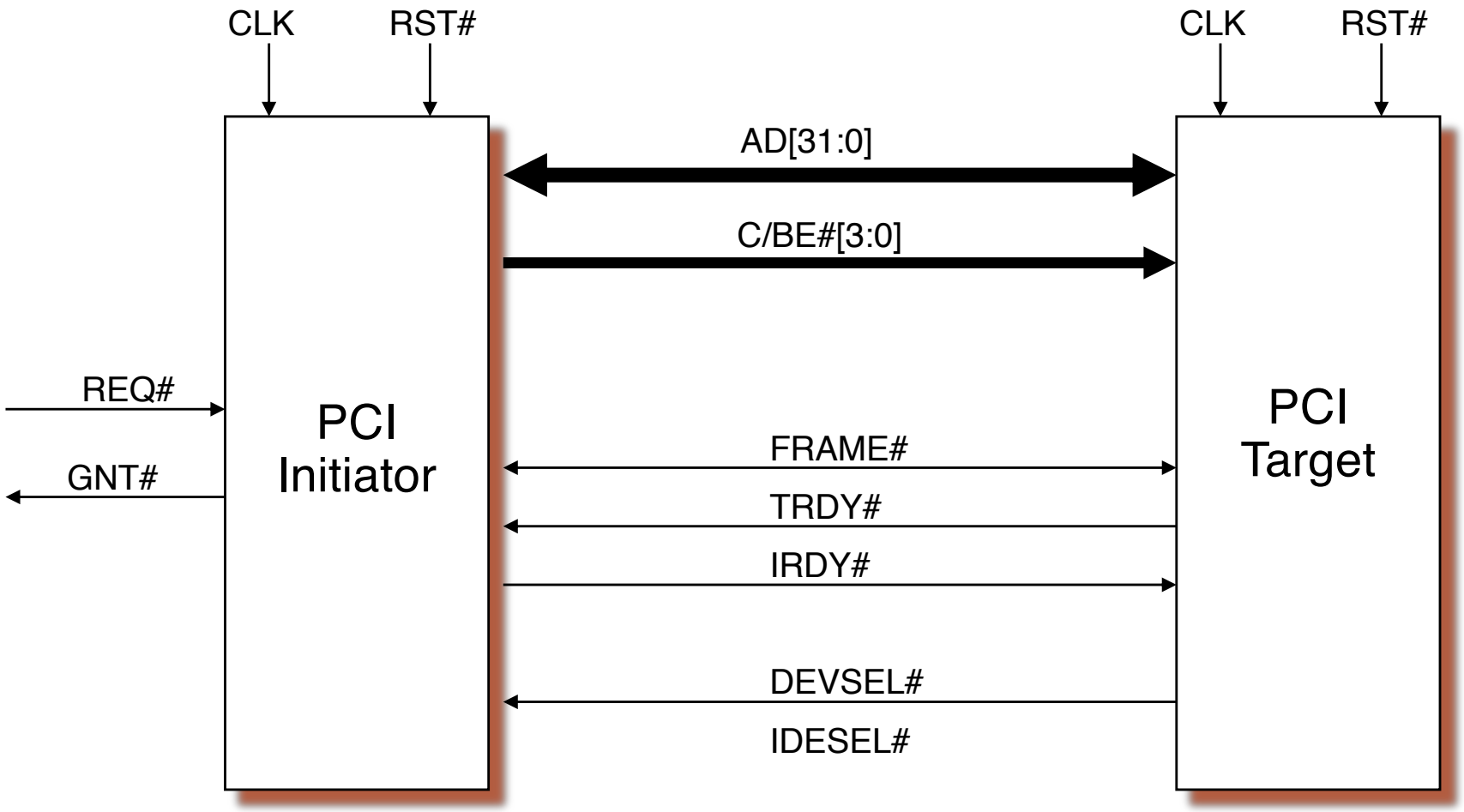


- Advantages
  - Centralized control where all devices submit request
  - Any fair priority scheme can be implemented (FCFS, round-robin)
- Disadvantages
  - Potential bottleneck at central arbiter

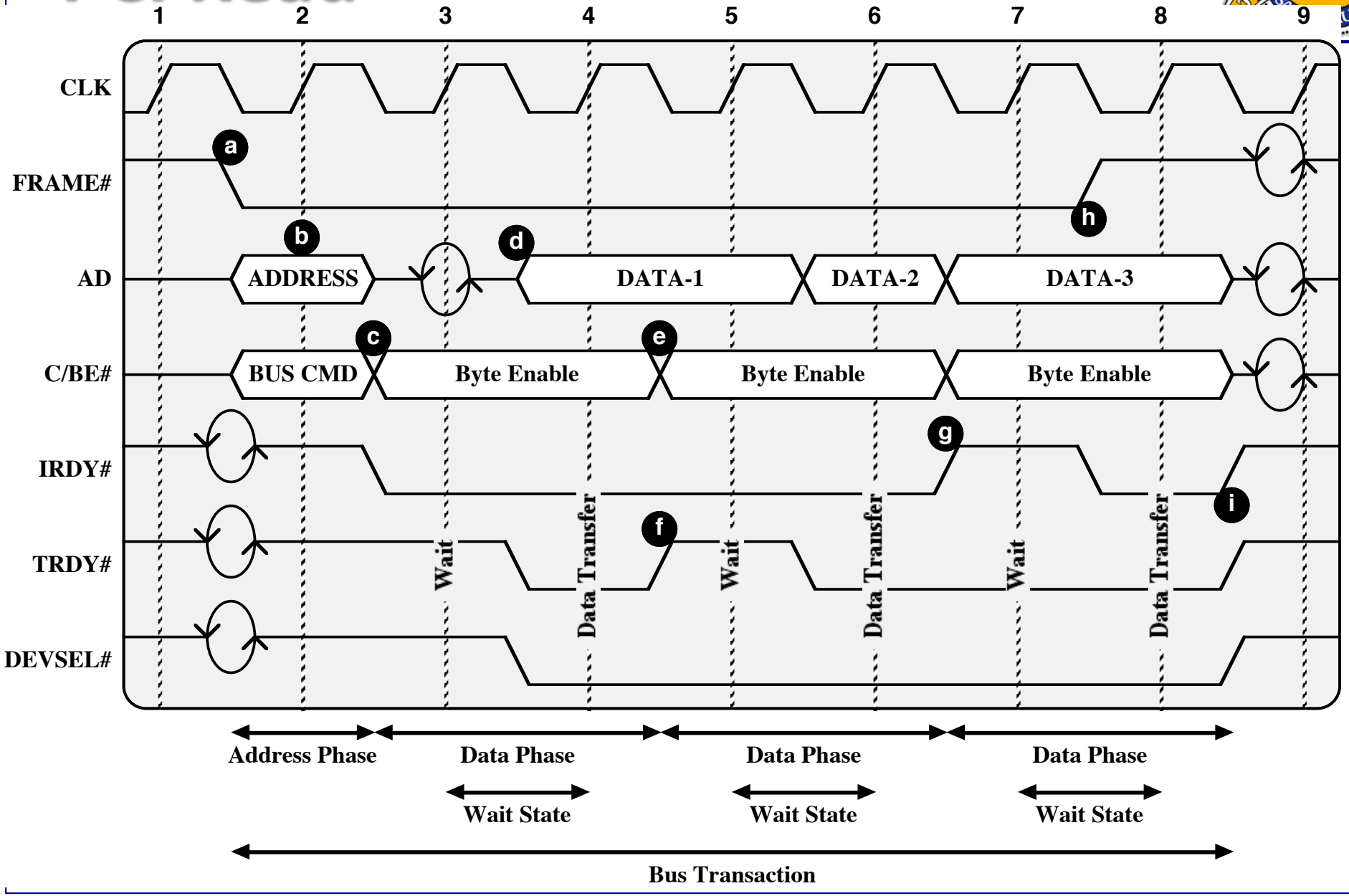
Case Study: PCI

- Peripheral Component Interconnect (PCI) peripheral backplane bus standard
- Clock Rate: 33 MHz (or 66 MHz in PCI Version 2.1) [CLK]
- Central arbitration [REQ#, GNT#]
  - Overlapped with previous transaction
- Multiplexed Address/Data
  - 32 lines (with extension to 64) [AD]
- Maximum bandwidth is 132 MB/s (533 MB/s at 64 bit/ 66 MHz)
- General Protocol
  - Transaction type (command is memory read, memory write, etc) [C/BE#]
  - Address handshake and duration [FRAME#, TRDY#]
  - Data width (byte enable) [C/BE#]
  - Variable length data block handshake between Initiatory Ready and Target Ready [IRDY#, TRDY#]

32 bit PCI Signals



PCI Read


1
2
3
4
5
6
7
8
9
CLK
FRAME#
AD
C/BE#
IRDY#
TRDY#
DEVSEL#
a
b
c
d
e
f
g
h
i
ADDRESS
DATA-1
DATA-2
DATA-3
BUS CMD
Byte Enable
Byte Enable
Byte Enable
Wait
Data Transfer
Wait
Data Transfer
Wait
Data Transfer
Address Phase
Data Phase
Data Phase
Data Phase
Wait State
Wait State
Wait State
Bus Transaction

PCI Read Steps 1

a) Once a bus master has gained control of the bus, it initiates the transaction by asserting FRAME. This line remains asserted until the last data phase. The initiator also puts the start address on the address bus, and the read command on the C/BE lines.

b) The target device recognizes its address on the AD lines.

c) The initiator ceases driving the AD bus. A turnaround cycle (marked with two circular arrows) is required before another device may drive any multiple-source bus. Meanwhile, the initiator changes the C/BE lines to designate which AD lines are to be used for data transfer (from 1-4 bytes wide). The initiator also asserts IRDY to indicate that it is ready for the first data item.

d) The selected target asserts DEVSEL to indicate that it has recognized its address and will respond. It places the requested data on the AD lines and asserts TRDY to indicate that valid data is present on the bus.

PCI Read Steps 2

e) The initiator reads the data at the beginning of clock 4 and changes the byte enable lines as needed in preparation for the next read.

f) In this example, the target needs some time to prepare the second block of data for transmission. Therefore, it deasserts TRDY to signal the initiator that there will not be new data during the coming cycle. Accordingly, the initiator does not read the data lines at the beginning of cycle 5 and does not change the byte enable on that cycle. The block of data is read at the beginning of cycle 6.

g) During clock 6, the target places the third data item on the bus. However, in this example the initiator is not yet ready to read the data item (i.e. temporarily buffers are full). It therefore deasserts IRDY. This will cause the target to hold the data for an extra cycle.

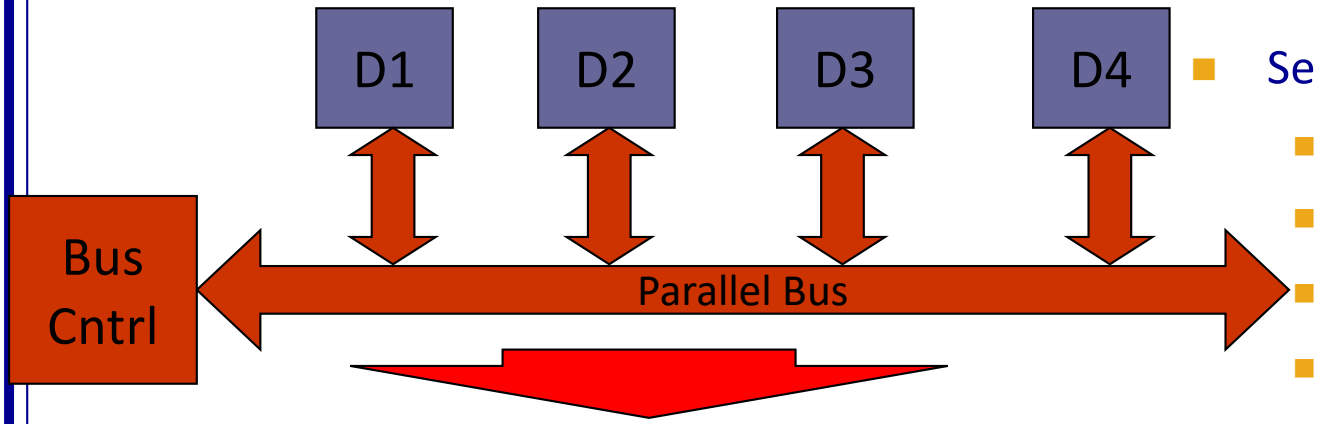
h) The initiator deasserts FRAME to signal the target that the third data transfer is the last, and asserts IRDY to signal that it is ready.

i) Return to the idle state. The initiator deasserts IRDY, and the target deasserts TRDY & DEVSEL.

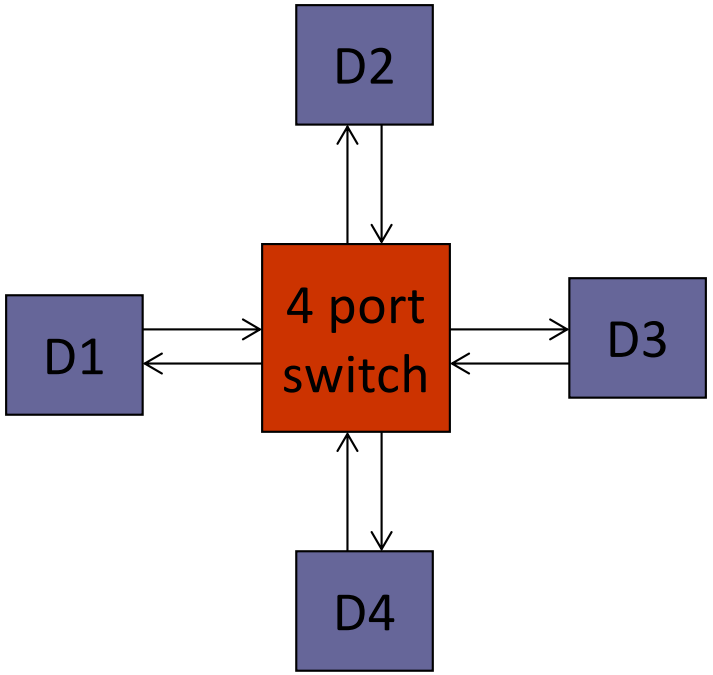
Buses Revisited

- Advantages:
  - Simple
  - Broadcast
  - Serialization
    - Where is this useful? Tip: think multi-core
- Drawbacks: many ways of saying it's slow
  - Only one transaction at a time
  - Electrically ugly
    - Stubs and discontinuities limit operating speed
  - Global closed-loop handshakes

Alternative: Point-to-Point Links & Switches



- Serial point-to-point advantages
  - Faster links
  - Fewer chip package pins
  - Higher performance
  - Switch keeps arbitration on chip



- Many bus standards are moving to serial, point to point
- 3GIO, PCI-Express(PCI)
- Serial ATA (IDE hard disk)
- AMD Hypertransport, Intel QPI versus Intel Front Side Bus (FSB)

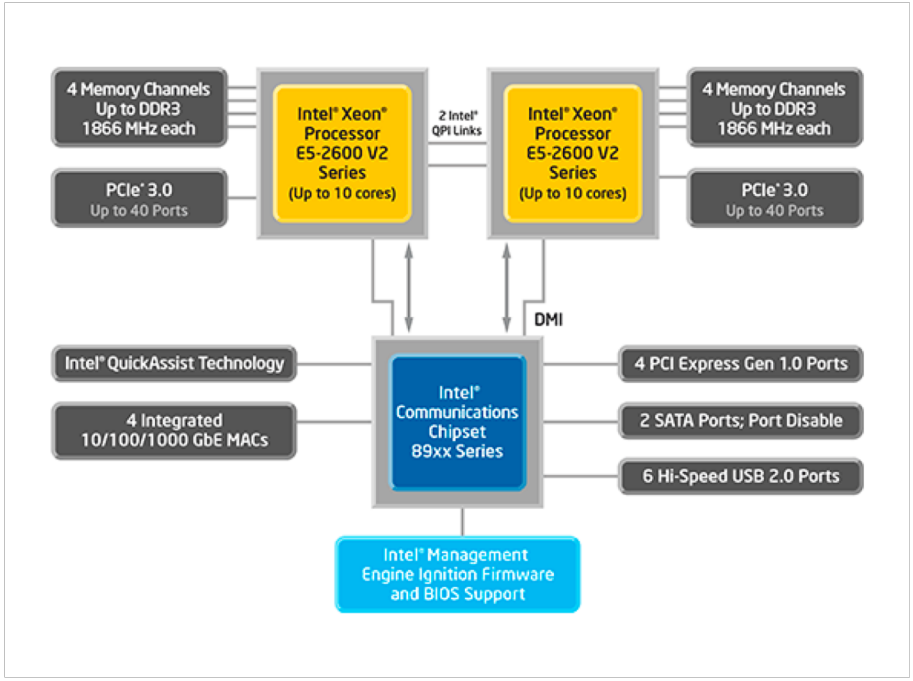
PCI vs. PCI Express

- Same bus protocol
  - Same driver software
- PCI
  - 32–64 shared wires
  - Frequency: 33MHz – 133 MHz
  - Bandwidth: 132 MB/s – 1 GB/s
- PCI Express
  - 1-16 wires per direction
  - Frequency: 2.5GT/s (PCIe 2.0: 5GT/s, PCIe 3.0: 5GT/s, PCIe 4.0: 16GT/s)
  - Bandwidth: 250 MB/s per direction per wire (500MB/984MB/1669MB)
- PCI Express advantage
  - 20 x pin bandwidth
  - Multiple links (lanes) for more bandwidth

Buses Vs Switched Interconnects

- Both useful, both used
- Busses used mostly on-chip
  - Mostly when connecting very small number of components
  - But also off-chip sometimes, e.g. DDR-x for main memory
- Switch-based interconnects used on and off-chip
  - Particularly when connecting many components
  - But topology can also be simple (e.g., rings)

Hierarchies of Interconnects



- Modern systems have hierarchies of buses/interconnects
  - Each chip tracks address maps
  - Memory-mapped I/O requests forwarded down the hierarchy until they reach the proper device
    - Potentially with translation along the way