ECE437: Introduction to Digital Computer Design

Chapter I/O

Fall 2016

Motivation

- I/O needed for
  - To/from users
  - To/from computers
  - To/from non-volatile storage media
- I/O Performance matters
- Total time = CPU + I/O – overlap
  - 10 + 4 – 4 = 10; 1x
  - 5 + 4 – [0,4] = [9,5]; [1.1x, 2x]
  - 1 + 4 – [0,1] = [5,4]; [2x, 2.5x]

I/O Performance

- What is performance?
  - With CPU : Iron Law
  - With I/O
    - Applications have different I/O needs
- Supercomputers read and write 1G of data
  - High data throughput
- Transactions processing does many independent, small I/O ops.
  - Fast I/O throughput (# of I/Os per sec)
- File systems
  - Fast response time, locality

I/O Device Characteristics

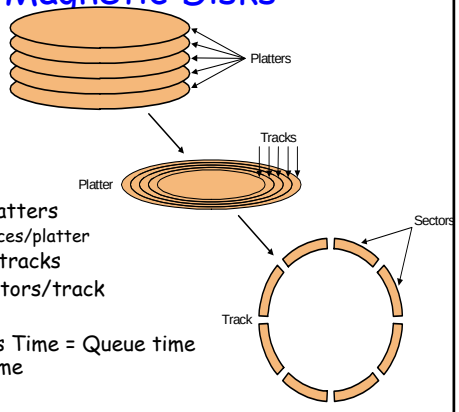
See Figure 6.2

Device	I or O?	Partner	Data Rate Mb/s
Mouse	I	Human	0.0038
Graphics Display	O	Human	800-8000
Modem	I/O	Machine	0.016-0.064
LAN	I/O	Machine	100-1000
Tape	Storage	Machine	32
Disk	Storage	Machine	240-2560

Disk Trends

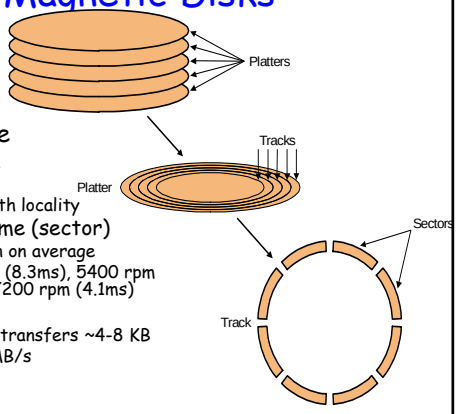
- Disk Trends
  - $/MB decreasing
    - 2010: $125/2TB = 6.3c/GB
    - 2008: $149/1.5TB = 10c/GB
    - 2006: $92/250 GB = 36c/GB
    - 2004 : 50c/GB (was 10000c/GB in 1997)
    - OEM prices lower
  - Disk diameter 14" -> 1.8" -> 1" **
  - Seek time down ~10ms
  - Rotation speed unchanged
    - ~7200 rpm for consumer
    - ~15K rpm for high end disks
  - Xfer rates up
  - Move from parallel buses to Serial bus (Serial ATA)

Magnetic Disks



- Stack of platters
  - Two surfaces/platter
- Concentric tracks
- Several sectors/track
- Disk Access Time = Queue time + service time

Magnetic Disks



- Service time
  - Seek track
    - ~10-20ms
    - Better with locality
  - Rotation time (sector)
    - ½ rotation on average
    - 3600 rpm (8.3ms), 5400 rpm (5.6ms), 7200 rpm (4.1ms)
  - Xfer
    - Page size transfers ~4-8 KB
    - Several MB/s

Disk Access Time

- Disks R' us™ has a GenX disk with the following parameters
  - Average seek time = 10ms
  - Rotation speed = 5400 rpm
  - Xfer rate = 160MB/s
  - Disk-block size = 4KB
  - Controller overhead = 3ms
- They introduce a GenY disk technology with rotation speed of 7200 rpm which they advertise as being 50% faster
- If average queue time is 0 ms, what is the true speedup of GenY over GenX when reading a randomly chosen disk-block.

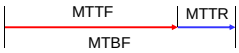
Disk Access Time

- GenX access time =
  - Queuetime + controller overhead + seektime+ rotational latency for ½ rotation + Xfer time
  - 0 + 3 + 10 + (0.5*60/5400 x 10^3)ms+ (4K/160M x 10^3) ms
  - 0+ 3 + 10 + 5.6 + 0.025
  - 18.625ms
- GenY access time
  - 0 + 3 + 10 + (0.5*60/7200 x 10^3)ms+ (4K/160M x 10^3) ms
  - 0 + 3 + 10 + 4.1 + 0.025
  - 17.125 ms
- Speedup = 18.625/17.125 = 1.087 = 8.7%

Exercise

- A disk has the following parameters
  - Access size : 8KB
  - Bandwidth: 80MB/s
  - Controller overhead : 0.1 ms
  - Seek time : 6ms
  - Rotation speed: 7200 rpm
- What is the average access time per transfer?
- If the disk controller has a 8MB cache that results in a 20% hit rate, what is the average access time? (Hits eliminate seek time and rotational latency)

I/O {Depend/Reli/Avail}-ability



- MTTF Mean time to Failure
- MTTR Mean time to repair
- MTBF Mean time between Failure
  - MTTR + MTTF
- Reliability metric: MTTF
- Availability metric: MTTF/MTBF

Redundant Array of Inexpensive Disks

- What if we want to store 100 disks
- MTTF : 5 yr/100 = 18 days
  - RAID 0 = No redundancy (Striping)
  - RAID 1 = mirror = full data redundancy = 100% overhead
  - RAID 3 = bit interleaved parity = small overhead
    - Dedicated parity disk
  - RAID 4 = block interleaved parity = small overhead + small writes
  - RAID 5 = distributed block wise parity = small overhead + small writes
- Other raid levels in book

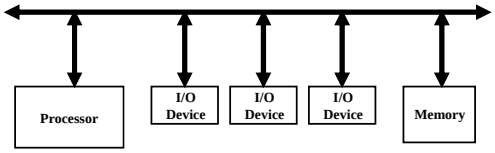
Recap

- Disks
  - Components of latency
  - Organization
  - RAID
  - Availability/Dependability/Reliability
- Buses
  - Types
    - One-bus/two-bus/three-bus architectures
  - Transaction timing
    - Synchronous vs. asynchronous

Connecting I/O to the CPU

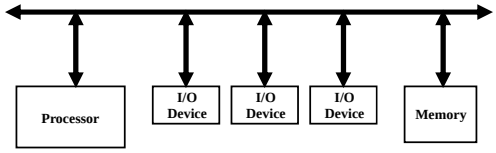
- Many I/O devices with vastly different datarates
- I/O device economics
  - Need standards
    - Large number of peripheral device producers
    - Multiple business entities involved favors longer lasting standards
- Let's examine several connection strategies

Advantages of Buses



- Versatility:
  - New devices can be added easily
  - Peripherals can be moved between computer systems that use the same bus standard
- Low Cost:
  - A single set of wires is shared in multiple ways
- Manage complexity by partitioning the design

Disadvantages of Buses

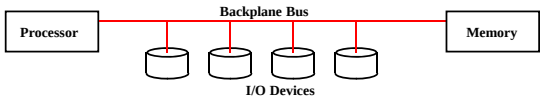


- It creates a communication bottleneck
  - The bandwidth of that bus can limit the maximum I/O throughput
- The maximum bus speed is largely limited by:
  - The length of the bus
  - The number of devices on the bus
  - The need to support a range of devices with:
    - Widely varying latencies
    - Widely varying data transfer rates

Types of buses

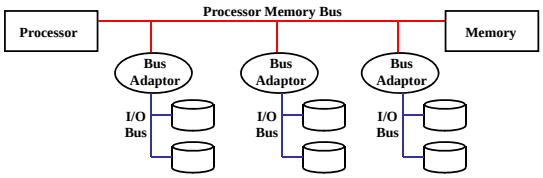
- Processor-Memory Bus (design specific/proprietary)
  - Short and high speed
  - Only need to match the memory system
    - Maximize memory-to-processor bandwidth
  - Connects directly to the processor
  - Optimized for cache block transfers
- I/O Bus (industry standard)
  - Usually is lengthy and slower
  - Need to match a wide range of I/O devices
  - Connects to the processor-memory bus or backplane bus
- Backplane Bus (standard or proprietary)
  - Backplane: an interconnection structure within the chassis
  - Allow processors, memory, and I/O devices to coexist
  - Cost advantage: one bus for all components

One Bus (Backplane) Architecture



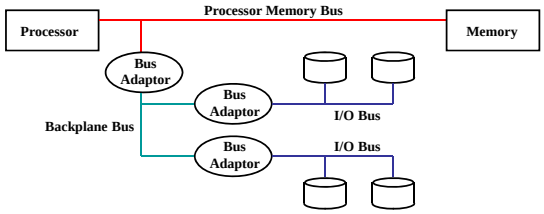
- A single bus (the backplane bus) is used for:
  - Processor to memory communication
  - Communication between I/O devices and memory
- Advantages: Simple and low cost
- Disadvantages: slow and the bus can become a major bottleneck
- Example: IBM PC – AT

Two Bus System



- I/O buses tap into the processor-memory bus via bus adaptors:
  - Processor-memory bus: mainly for processor-memory traffic
  - I/O buses: provide expansion slots for I/O devices
- Apple Macintosh-II
  - NuBus: Processor, memory, and a few selected I/O devices
  - SCCI Bus: the rest of the I/O devices

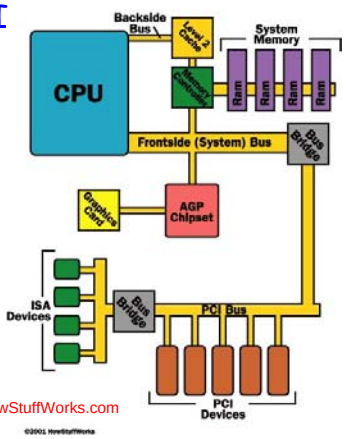
Three Bus System



- A small number of backplane buses tap into the processor-memory bus
  - Processor-memory bus is used for processor memory traffic
  - I/O buses are connected to the backplane bus
- Advantage: loading on the processor bus is greatly reduced

Real Stuff: PCI

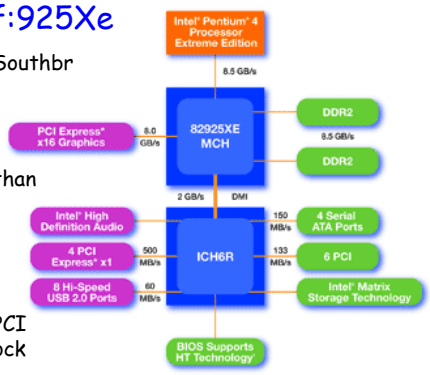
- Older
  - Dual Independent Bus (DIB) Architecture between processor/memory/L2 cache
- Peripheral Component Interconnect
  - 33 (66) MHz, synchronous, 32 (64) bit data, multiple masters
- Claim: PCI too slow for hi-end graphics.
  - AGP taps directly into FSB



Graphic from HowStuffWorks.com

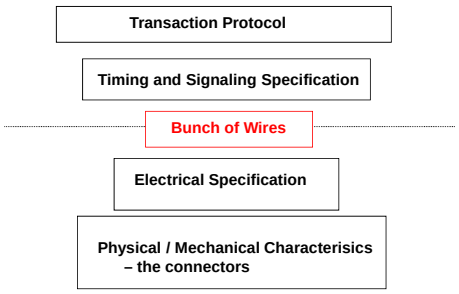
Real Stuff:925Xe

- Northbridge/Southbridge
- Serial faster than parallel
- PCI-e
- PCI-X is just PCI with faster clock



Graphic from intel.com

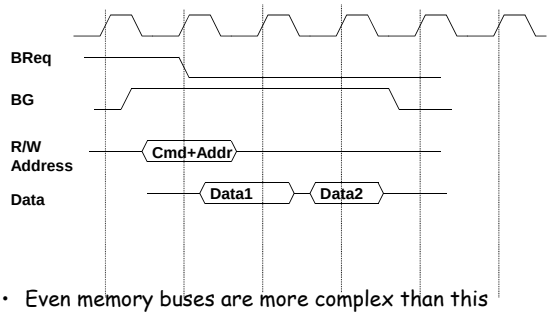
Bus Definition



Synchronous vs. Asynchronous

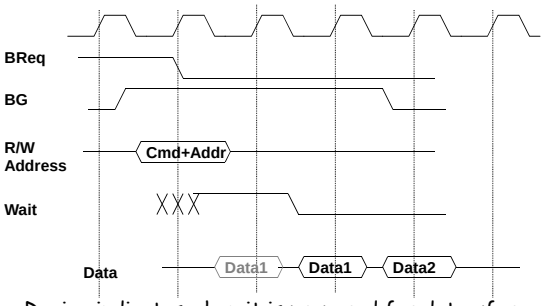
- Synchronous Bus:
  - Includes a clock in the control lines
  - A fixed protocol for communication that is relative to the clock
  - Advantage: involves very little logic and can run very fast
  - Disadvantages:
    - Every device on the bus must run at the same clock rate
    - To avoid clock skew, they cannot be long if they are fast
- Asynchronous Bus:
  - It is not clocked
  - It can accommodate a wide range of devices
  - It can be lengthened without worrying about clock skew
  - It requires a handshaking protocol

Simple Synchronous Protocol



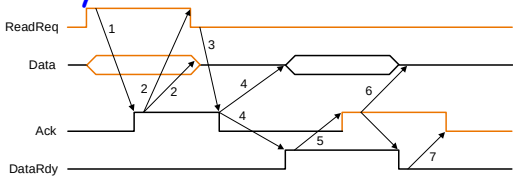
- Even memory buses are more complex than this
  - memory may take time to respond
  - need to control data rate

Typical Synchronous Protocol



- Device indicates when it is prepared for data xfer
- Actual transfer goes at bus rate

Asynchronous Bus Protocol



Proc asserts READREQ, <address>	
	Mem asserts ACK after reading <address>
Proc deasserts READREQ, <address> and waits	
	Mem deasserts ACK
	Mem asserts DATA-RDY, <data>
Proc asserts ACK after reading <data>	
	Mem deasserts DATARDY, <data>
Proc deasserts ACK	

I/O Interfacing: Outline

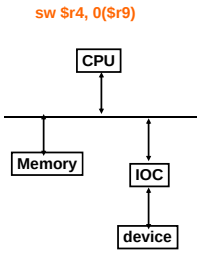
- How does the CPU initiate I/O?
- How does I/O device notify CPU when I/O op is complete?
- Who (on the CPU) performs I/O ops?

I/O Interfacing

- I/O operation needs to be initiated
  - Special opcodes
  - Memory-mapped I/O
- I/O completion must be known
  - Polling
  - Interrupt-driven

I/O commands

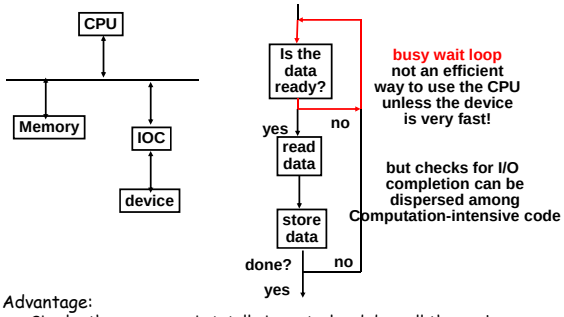
- Memory-mapped I/O
  - Special addresses not for memory ($r9 contains special address for IOC)
  - Commands written (sw) as data ($r4)
- I/O commands
  - Special opcodes
  - Send over I/O Bus



I/O Device Notifying the OS

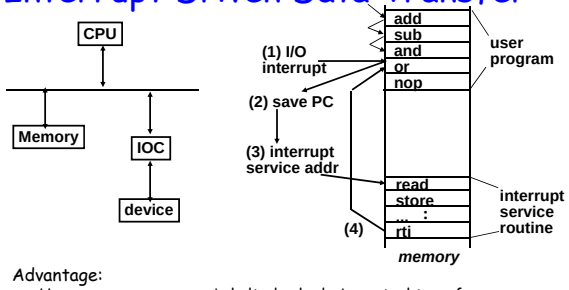
- The OS needs to know when:
  - The I/O device has completed an operation
  - The I/O operation has encountered an error
- This can be accomplished in two different ways:
  - Polling:
    - The I/O device put information in a status register
    - The OS periodically check the status register
  - I/O Interrupt:
    - Whenever an I/O device needs attention from the processor,
      it interrupts the processor from what it is currently doing.

Polling: Programmed I/O



- Advantage:
  - Simple: the processor is totally in control and does all the work
- Disadvantage:
  - Polling overhead can consume a lot of CPU time

Interrupt Driven Data Transfer



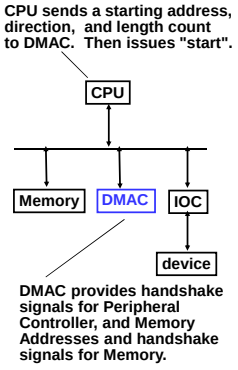
- Advantage:
  - User program progress is halted only during actual transfer
- Disadvantage, special hardware is needed to:
  - Cause an interrupt (I/O device)
  - Detect an interrupt (processor)
  - Save the proper states to resume after the interrupt (processor)

I/O Interrupts

- An I/O interrupt is just like the exceptions except:
  - An I/O interrupt is asynchronous
  - Further information needs to be conveyed
- An I/O interrupt is asynchronous with respect to instruction execution:
  - I/O interrupt is not associated with any instruction
  - I/O interrupt does not prevent any instruction from completion
    - You can pick your own convenient point to take an interrupt
- I/O interrupt is more complicated than exception:
  - Needs to convey the identity of the device generating the interrupt
  - Interrupt requests can have different urgencies:
    - Interrupt request needs to be prioritized

Direct Memory Access (DMA)

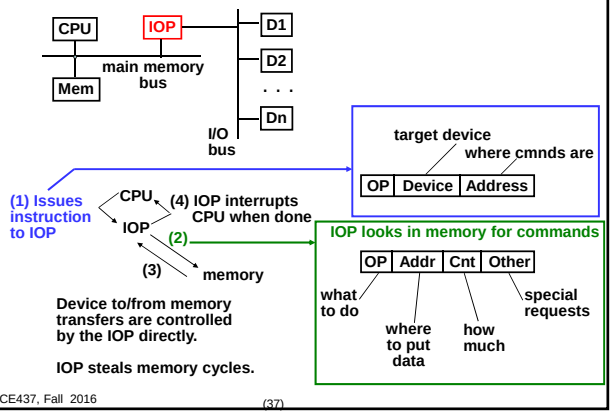
- Direct Memory Access (DMA):
  - External to the CPU
  - Act as a master on the bus
  - Transfer blocks of data to/from memory without CPU intervention



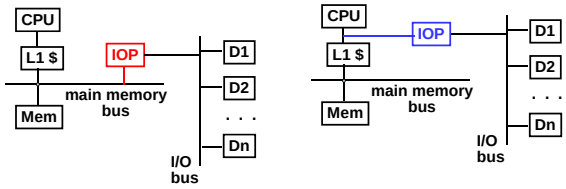
DMA interfacing

- DMA : virtual addresses or physical addresses?
- Challenge: crossing page boundaries
  - Virtual addresses
    - Translation provided by OS
  - Physical addresses
    - One page per transfer
    - OS chains the physical addresses
- No page faults in between
  - "Pin" pages

Delegating I/O Responsibility



I/O Coherence Problem



- Where is the cache w.r.t. I/O device (and/or I/O processor)?
  - In front of cache : Slows down CPU
  - Behind Cache : Cache coherence
- Applies to any I/O device that writes autonomously to memory
  - DMA, other
  - Multiprocessors also
- Coherence is a problem when both the following are involved:
  - Multiple writers AND
  - Replication

CPU involvement

- Low offloading to high offloading
  - Polling
    - CPU does everything including detecting events and subsequent I/O copying to/from memory
  - Interrupts
    - Someone else notifies CPU when processing needed, CPU does the needful
  - DMA
    - CPU describes what needs to be done, DMAC does the rest
  - IOP
    - CPU tells IOP where to find instructions on what needs to be done

OS Responsibilities

- The operating system acts as the interface between:
  - The I/O hardware and the program that requests I/O
  - Why?
    - Why not let user programs directly read/write to I/O devices?
- Three characteristics of the I/O systems:
  - The I/O system is shared by multiple program using the processor
  - I/O systems often use interrupts (external generated exceptions) to communicate information about I/O operations.
    - Interrupts must be handled by the OS because they cause a transfer to supervisor mode
  - The low-level control of an I/O device is complex:
    - Managing a set of concurrent events
    - The requirements for correct device control are very detailed

OS Responsibilities

- Provide protection to shared I/O resources
  - Guarantees that a user's program can only access the portions of an I/O device to which the user has rights
- Provides abstraction for accessing devices:
  - Supply routines that handle low-level device operation
- Handles the interrupts generated by I/O devices
- Provide equitable access to the shared I/O resources
  - All user programs must have equal access to the I/O resources
- Schedule accesses in order to enhance system throughput

Interfacing: Summary

- Multiprogramming
  - Program invokes syscall (i.e., invokes OS)
  - OS checks protection
  - OS runs device drivers
  - Suspends current process and switches process
  - I/O interrupt fielded by OS
  - OS completes I/O and wakes up suspended process (i.e. make it runnable)
  - Run next ready process
- Ch I/O done!

The end!

- ECE 437 done!
- Remember that you learnt the 4 key pillars of modern computer systems
  - Pipelining
  - Caches
  - Coherence
  - Virtual memory
  - And a few other cool things