I/O System

Computer Organization

Sunday, 16 October 2022

Many slides adapted from:
Computer Organization and Design,
Patterson & Hennessy
5th Edition, © 2014, MK
and from Prof. Mary Jane Irwin, PSU

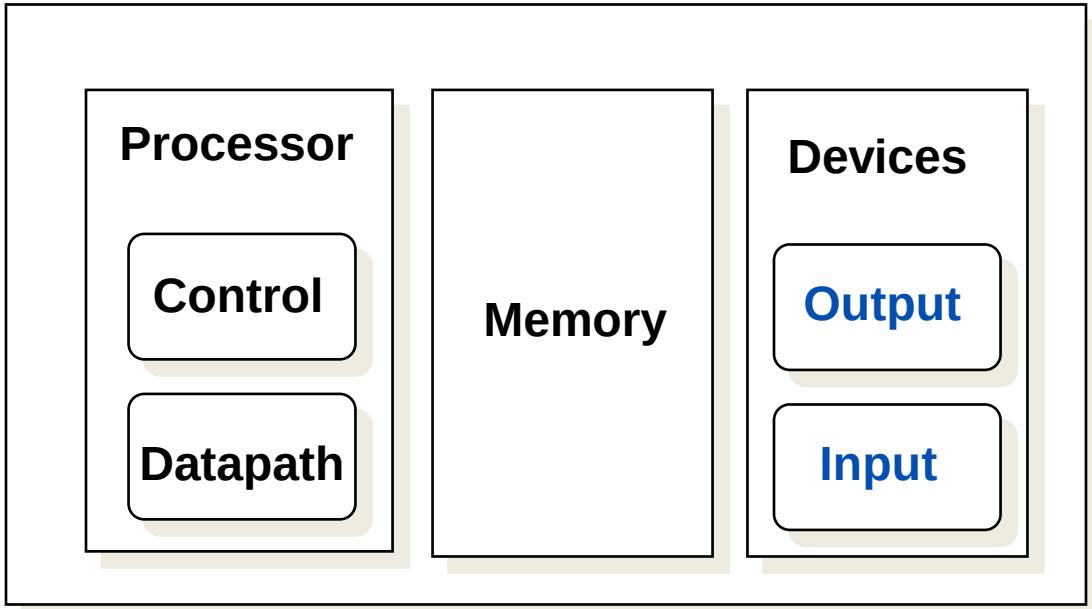
TÉCNICO LISBOA

Summary

- Previous Class
  - The Processor: Instruction-Level Parallelism

- Today:
  - IO System

Review: Major Components of a Computer

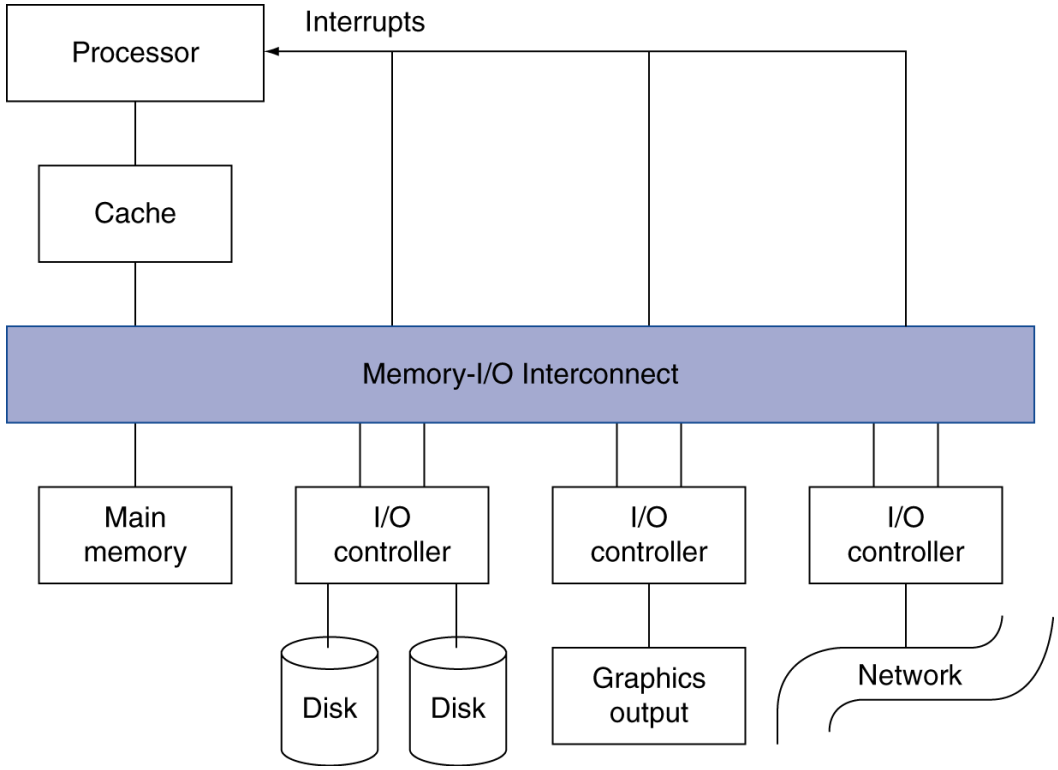


- Important metrics for an I/O system
  - Performance
  - Expandability
  - Dependability
  - Cost, size, weight
  - Security

Introduction

- I/O devices can be characterized by
  - Behavior: input, output, storage
  - Partner: human or machine
  - Data rate: bytes/sec, transfers/sec

- I/O bus connections



I/O System Characteristics

- Dependability is important
  - Particularly for storage devices
- Performance measures
  - Latency (response time)
  - Throughput (bandwidth)
  - Desktops & embedded systems
    - Mainly interested in response time & diversity of devices
  - Servers
    - Mainly interested in throughput & expandability of devices

I/O System Interconnect Issues

- A bus is a shared communication link (a single set of wires used to connect multiple subsystems) that needs to support a range of devices with widely varying latencies and data transfer rates
  - Advantages
    - Versatile – new devices can be added easily and can be moved between computer systems that use the same bus standard
    - Low cost – a single set of wires is shared in multiple ways
  - Disadvantages
    - Creates a communication bottleneck – bus bandwidth limits 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

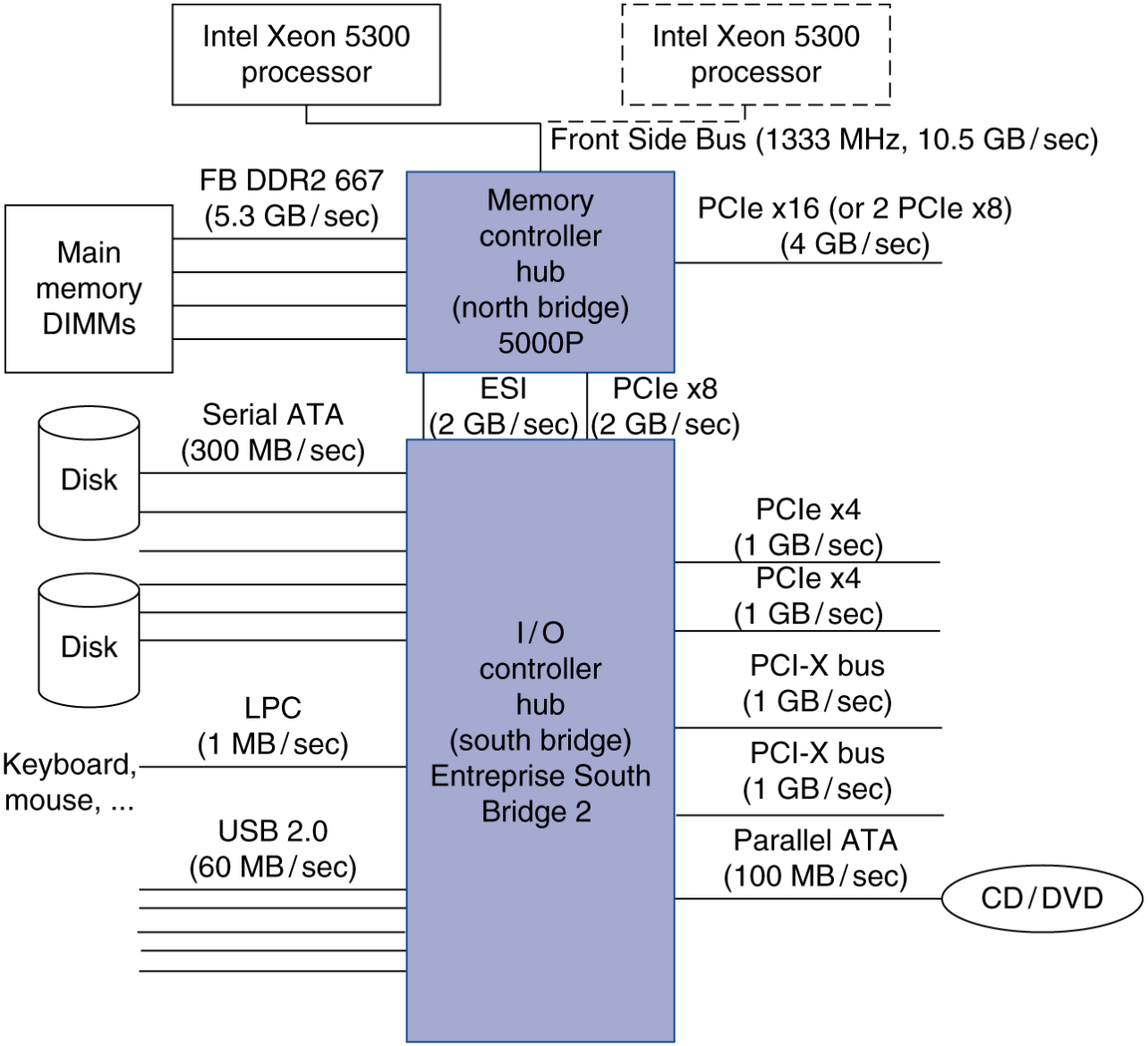
Bus Signals and Synchronization

- Data lines
  - Carry address and data
  - Multiplexed or separate
- Control lines
  - Indicate data type, synchronize transactions
- Synchronous
  - Uses a bus clock
- Asynchronous
  - Uses request/acknowledge control lines for handshaking

Types of Buses

- Processor-memory bus (“Front Side Bus”, proprietary)
  - Short and high speed
  - Design matched to memory organization to maximize bandwidth
  - Optimized for cache block transfers
- I/O bus (industry standard, e.g., ATA, SCSI, USB, Firewire)
  - Usually is lengthy and slower, allowing multiple connections
  - Needs to accommodate a wide range of I/O devices
  - Connect to processor-memory bus through a bridge or a backplane bus
- Backplane bus (industry standard, e.g., PCIexpress)
  - Allow processor, memory and I/O devices to coexist on a bus
  - Used as an intermediary bus connecting I/O busses to the processor-memory bus

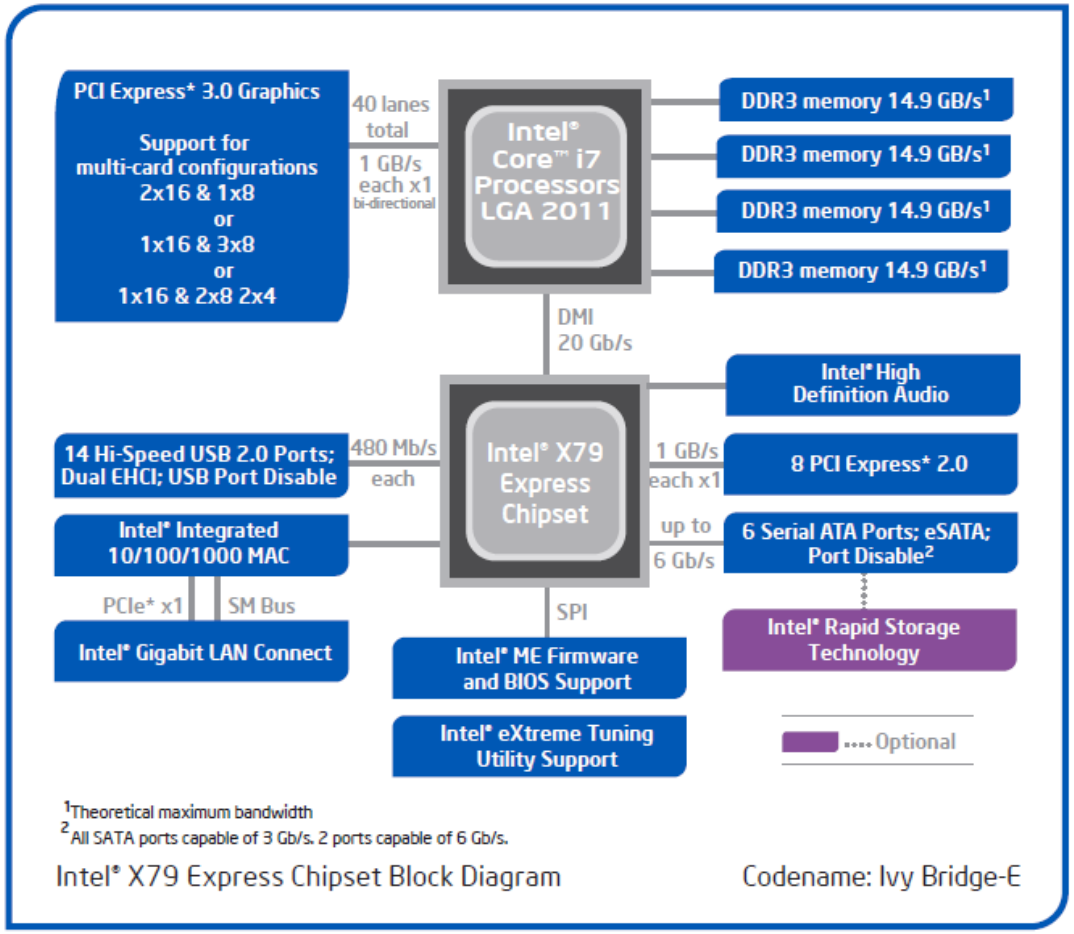
Typical x86 PC I/O System

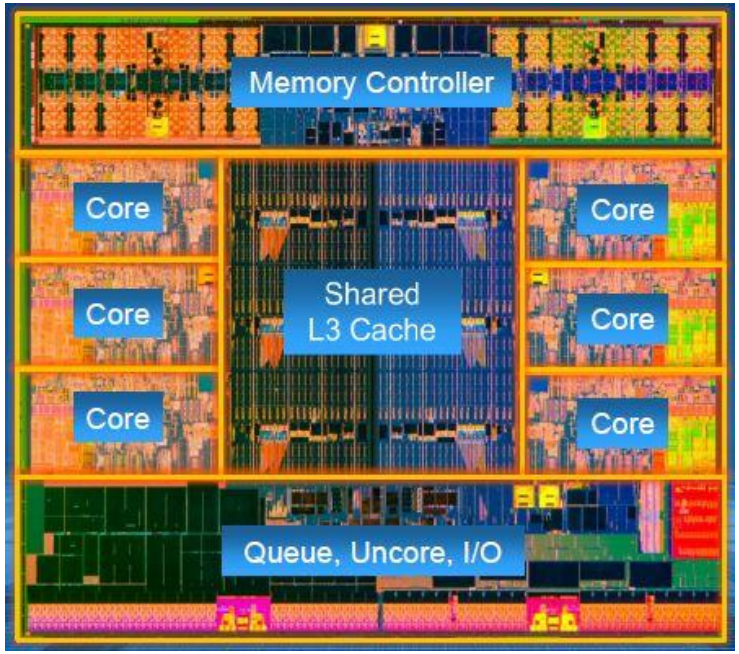


I/O Bus Examples

	Firewire	USB	PCI Express	Serial ATA	Serial Attached SCSI
Intended use	External	External	Internal	Internal	External
Devices per channel	63	127	1	1	4
Data width	4	1 - 3	2/lane	4	4
Peak bandwidth (MB/s)	375	0.2, 1.5, 60, 625, 1250, 2500, 5000	1969/lane 1×, 2×, 4×, 8×, 16×, 32×	150, 300, 600, 2000, 6000	300
Hot pluggable	Yes	Yes	Depends	Yes	Yes
Max length	4.5m	5m	0.5m	1m	8m
Standard	IEEE 1394	USB Implementers Forum	PCI-SIG	SATA-IO	INCITS TC T10

Intel i7-4960X Ivy Bridge-E





I/O Management

- I/O is mediated by the OS
  - Multiple programs share I/O resources
    - Need protection and scheduling
  - I/O causes asynchronous interrupts
    - Same mechanism as exceptions
  - I/O programming is fiddly
    - OS provides abstractions to programs

I/O Commands

- I/O devices are managed by I/O controller hardware
  - Transfers data to/from device
  - Synchronizes operations with software
  - Command registers
    - Cause device to do something
  - Status registers
    - Indicate what the device is doing and occurrence of errors
  - Data registers
    - Write: transfer data to a device
    - Read: transfer data from a device

I/O Register Mapping

- Memory mapped I/O
  - Registers are addressed in same space as memory
  - Address decoder distinguishes between them
  - OS uses address translation mechanism to make them only accessible to kernel

- I/O instructions
  - Separate instructions to access I/O registers
  - Can only be executed in kernel mode
  - Example: x86

Polling

- Periodically check I/O status register
  - If device ready, do operation
  - If error, take action
- Common in small or low-performance real-time embedded systems
  - Predictable timing
  - Low hardware cost
- In other systems, wastes CPU time

Interrupts

- When a device is ready or error occurs
  - Controller interrupts CPU
- Interrupt is like an exception
  - But not synchronized to instruction execution
  - Can invoke handler between instructions
  - Cause information often identifies the interrupting device
- Priority interrupts
  - Devices needing more urgent attention get higher priority
  - Can interrupt handler for a lower priority interrupt

I/O Data Transfer

- Polling and interrupt-driven I/O
  - CPU transfers data between memory and I/O data registers
  - Time consuming for high-speed devices
- Direct memory access (DMA)
  - OS provides starting address in memory
  - I/O controller transfers to/from memory autonomously
  - Controller interrupts on completion or error

DMA/Cache Interaction

- If DMA writes to a memory block that is cached
  - Cached copy becomes invalid
- If write-back cache has dirty block, and DMA reads memory block
  - Reads invalid data
- Need to ensure cache coherence
  - Flush blocks from cache if they will be used for DMA
  - Or use non-cacheable memory locations for I/O

DMA / VM Interaction

- OS uses virtual addresses for memory
  - DMA blocks may not be contiguous in physical memory
- Should DMA use virtual addresses?
  - Would require controller to do translation
- If DMA uses physical addresses
  - May need to break transfers into page-sized chunks
  - Or chain multiple transfers
  - Or allocate contiguous physical pages for DMA

I/O System Design

- Satisfying latency requirements
  - For time-critical operations
  - If system is unloaded
    - Add up latency of components
- Maximizing throughput
  - **Find "weakest link"** (lowest-bandwidth component)
  - Configure to operate at its maximum bandwidth
  - Balance remaining components in the system
- If system is loaded, simple analysis is insufficient
  - Need to use queuing models or simulation

Pitfall: Offloading to I/O Processors

- Overhead of managing I/O processor request may dominate
  - Quicker to do small operation on the CPU
  - But I/O architecture may prevent that
- I/O processor may be slower
  - Since it's supposed to be simpler
- Making it faster makes it into a major system component
  - Might need its own coprocessors!

Pitfall: Peak Performance

- Peak I/O rates are nearly impossible to achieve
  - Usually, some other system component limits performance
  - E.g., transfers to memory over a bus
    - Collision with DRAM refresh
    - Arbitration contention with other bus masters
  - E.g., PCI bus: peak bandwidth ~133 MB/sec
    - In practice, max 80MB/sec sustainable

Conclusion

- I/O performance measures
  - Throughput, response time
  - Dependability and cost also important
- Buses used to connect CPU, memory, I/O controllers
  - Polling, interrupts, DMA

Next Class

- Disk Storage
- Dependability

I/O System

Computer Organization

Sunday, 16 October 2022

Many slides adapted from:
Computer Organization and Design,
Patterson & Hennessy
5th Edition, © 2014, MK
and from Prof. Mary Jane Irwin, PSU

TÉCNICO LISBOA