CMP 1203

LECTURE 7

I/O and Performance

- Slow computer: processing time exceeds user's "think" time
- Most focus is on CPU speed but I/O processing can slow down a computer
- Amdahl's Law
- The overall speed-up of a computer depends on both the speed-up in a particular component and how much that component is used by the system

$$S = \frac{1}{(1 - f) + f / k}$$

- ***S*** is the overall system speedup, ***f*** is the fraction of work done by sped-up component, ***k*** is the speed-up of the new component

Interrupt Structures

- **How does processor know which device issued an interrupt?**
- **If multiple interrupts have occurred, how does processor know which one to process?**
- **4 possible ways**

1. Provide ***multiple interrupt lines*** between processor and I/O modules. **BUT** impractical to dedicate too many lines to interrupts. Also high probability of sharing an interrupt line.
2. ***Software polling***: poll each I/O module to determine which caused interrupt. Once device responds, activate service routine for that device

Interrupt Structures

3. ***Hardware poll/ vectored interrupt***: Each device is assigned an interrupt vector
- Vector identifies a particular interrupt handler
- When the device interrupts, system asks device to identify itself
- Device responds with interrupt vector
- Vector used to identify the handler

4. ***Bus arbitration***: uses vectored interrupts
- I/O module must first gain control of the bus before it raises an interrupt
- Only one module can raise an interrupt at a time

Interrupt Structures

- Priority is used when more than one device issues interrupt
- Multiple lines ➔ respond to line with highest priority
- Software polling ➔ order of polling determines priority
- Order of modules on a daisy-chain (hardware polling) determines priority
- Bus arbitration ➔ uses a priority scheme

Character Vs. Block I/O

- I/O devices have different data processing rates (avg. number of characters that can be processed by a device per second)
- Influences choice of I/O control processing

1. Keyboard ➔ strike sends character to computer. Time between next x-ter depends on user thinking speed

- Slow and burst-like input
- Computer cant waste time waiting for input
- Which mechanism is appropriate?

2. Disk ➔ typically high transfer rates. Useless to transfer data byte per byte or word per word.

- Transfer in blocks. Choose huge data transfer mechanism without CPU intervention
- Which mechanism would we choose?

1.Interrupt driven I/O 2. DMA!

I/O Buffering

- Buffering: Accumulating data before I/O transfer
- Buffer: area in memory where the data is placed
- External devices which handle large blocks of data e.g. printers, disks have buffer memory
- Buffers allow system to send large amounts of data to peripherals quickly without waiting for slow mechanical devices to actually write the data
- Device control circuits take data to/from buffers and ensure it gets where its going e.g. moving print head, ejecting paper in a printer, aligning disk locations (sectors) to write head

Buses

- Physical connection used to carry signals form one point to another
- Bus ➔ several connections together, each called a bus line
- ***Synchronous bus***: data transfer over the bus is controlled by a bus clock
- ***Asynchronous bus***: data transfer is based on availability of data and not on clock cycles
- Asynchronous buses use ***handshaking***
- **Event Sequence**

1. Master sends request to use bus
2. Request is granted and bus allocated to master
3. Master places address/data on bus
4. Slave is selected
5. Master signals data transfer
6. Slave accepts data
7. Master frees bus

Synchronous Bus

- Data transfer steps occur at fixed clock cycles
- Clock signals available to both master and slave
- Bus clock is square wave
- Transfer may take one or more clock cycles ➔ depends on bus speed and the two devices
- **Example**: on first clock cycle:
- Master puts address on address bus and data on data bus
- Master activates control lines
- Slave recognizes its address on address bus
- **On second cycle:**
- Slave reads data on bus etc
- Synchronous bus is simple and easy to implement BUT if several devices using it, slowest device determines speed of bus

Asynchronous Bus

- No fixed cycles
- Example of handshaking protocol

1. Master activates data ready line until it receives data accept signal
2. Slave sees data ready and asserts data accept signal.
3. Action 2 triggers fall of data ready and removes data from bus
4. Action 3 triggers fall of data-accept line
5. Repeat until all data transferred

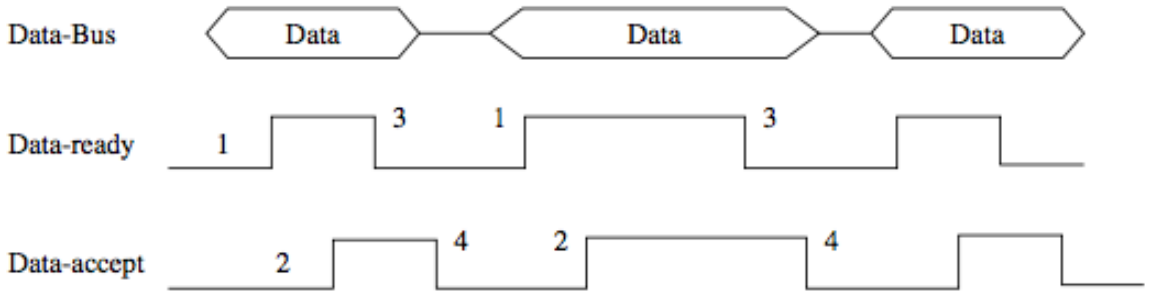


Figure 8.11 Asynchronous bus timing using handshaking protocol

Sources: fundamentals of computer architecture and organization

Bus Arbitration

- Selecting bus master among multiple candidates
- Resolves conflict when multiple devices want to be bus master at the same time

Centralized Arbitration

- Uses a single arbiter to select next bus master
- Bus request line, bus grant line and bus busy line
- Each line shared by potential masters in cascade

Centralized Arbitration: Example 1

- Each potential master can submit request at any time
- Fix priority from left to right
- When arbiter receives bus request, it sets bus grant line
- Potential master closest to arbiter (master 1) sees bus grant, checks to see if it had requested to use bus
- If yes, takes over bus and stops propagation of bus grant signal
- If no, it allows bus grant signal to continue to next master
- And so on
- When transaction is complete, bus busy line is deactivated

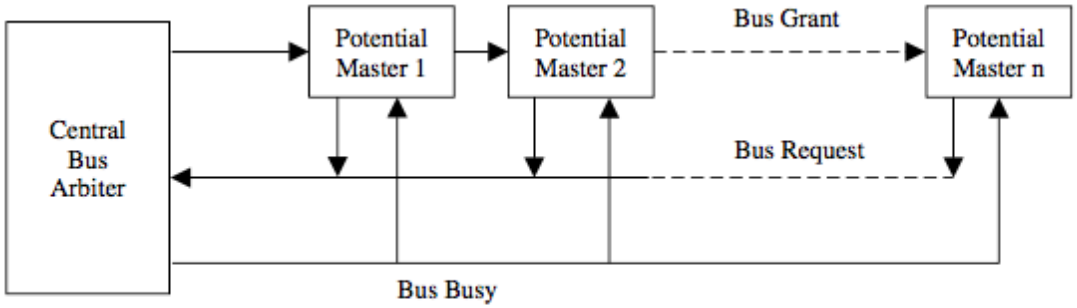


Figure 8.12 Centralized arbiter in a daisy-chain scheme

Centralized Arbitration: Example 2

- Use multiple bus request and bus grant lines
- Each master has its own lines
- Any tie-breaker method – fairness or priority can be used by arbiter

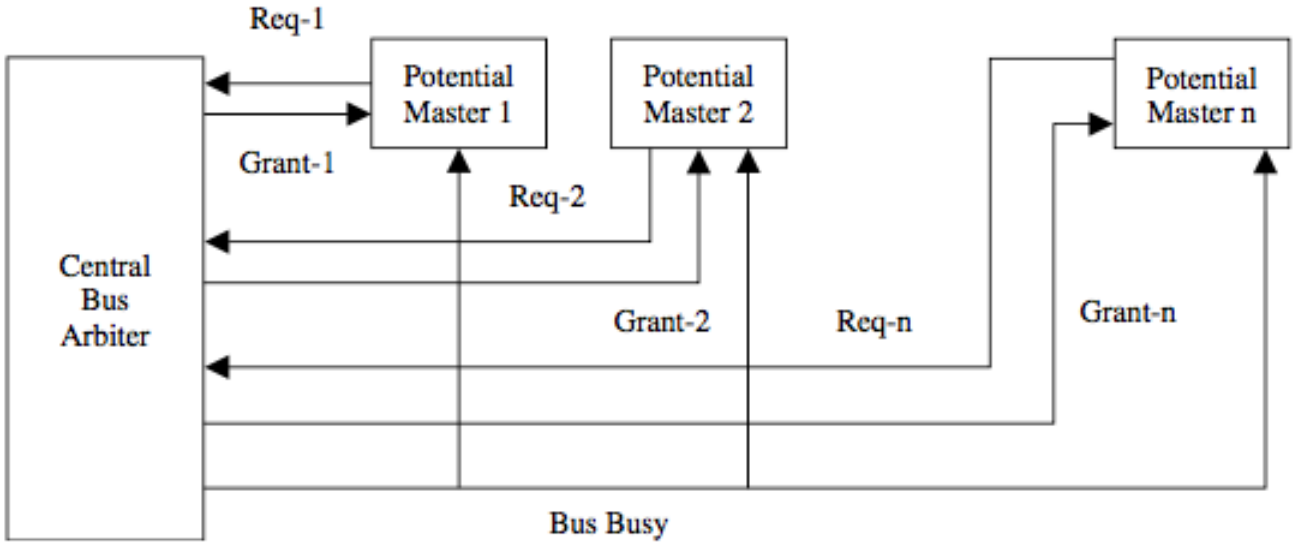


Figure 8.13 Centralized arbiter with independent request and grant lines

Centralized Arbitration: Example 3

- Masters have multiple priority levels
- Within each priority level, use daisy chain (example 1)
- Each device attached to one priority level
- If multiple requests in different levels, assign bus based on priority
- If within same level – same as example 1

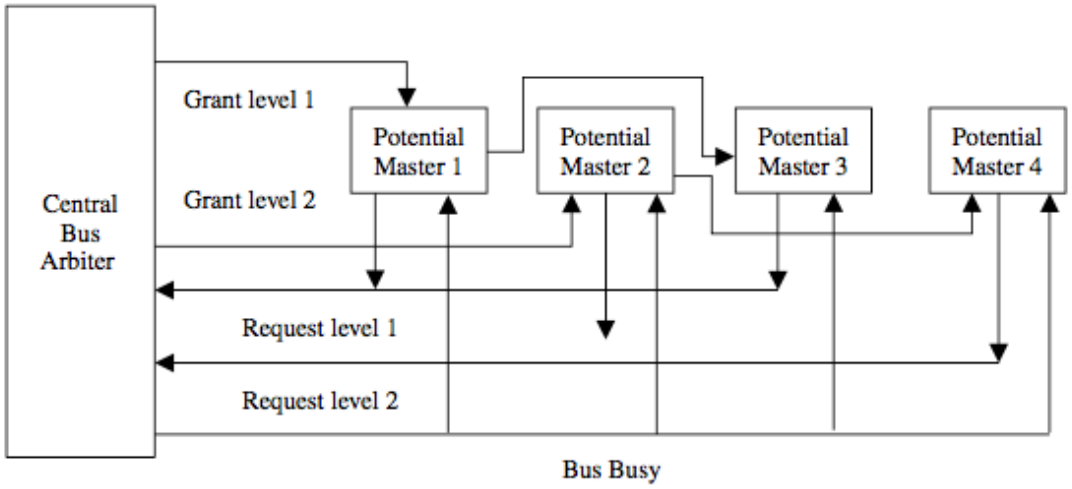


Figure 8.14 Centralized arbiter with two priority levels (four devices)

Decentralized Arbitration

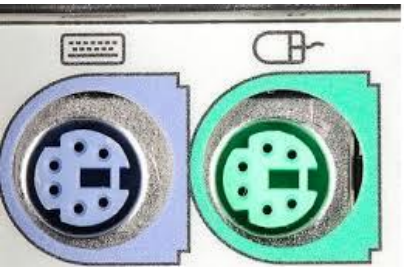
- Uses priority-based arbitration
- Each potential master has a unique arbitration number
- Number is used to resolve conflicts e.g. grant device with highest arbitration number. How?
- E.g. device which wishes to use bus avails its number to all devices
- Each device compares with its own
- Smaller number is always dismissed
- In the end, device with highest number remains and can use bus

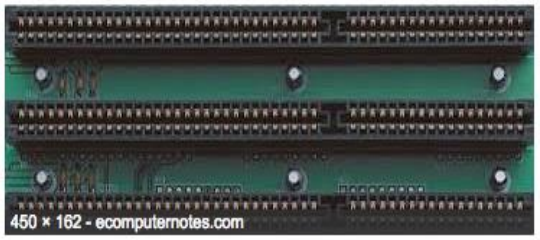
I/O Interfaces

- Interface: data path between 2 separate devices in a computer system
- Classified based on number of bits transmitted at a time: serial vs. parallel ports
- Parallel: has multiple wires, each carrying one bit
- Serial: fewer wires, less interference form signals travelling at same time BUT increased latency – system must wait for one bit to be sent before sending another

Bus/Interface Examples

Bus/Interface	Description
PS/2	Port for mice/ keyboards
ISA (Industry standard Architecture)	Originally 8-bit bus, now 16-bit
EISA (extended ISA)	32-bit data but also supports 8 and 16-bit.
Micro Channel Architecture (MCA)	32-bit bus
PCI (Peripheral component Interconnect)	32- bit bus. Also 64-bit variant
Advanced Graphic Port	32-bit bus designed for 3D graphics



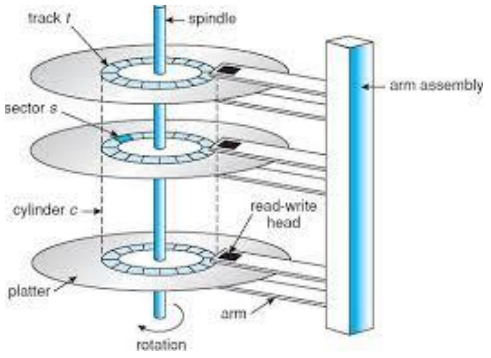


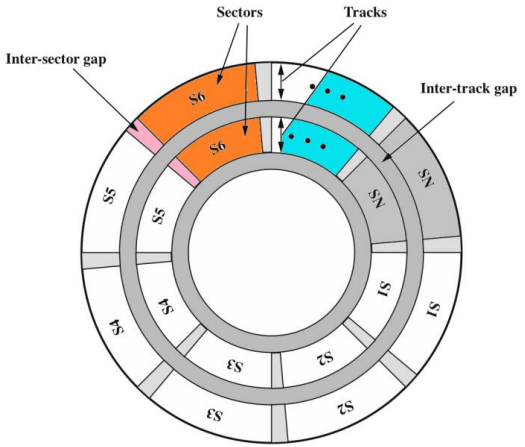
Bus/Interface Examples

Bus/Interface	Description
USB	External bus. Supports 127 devices. 12 Mbps , plug and play. Devices connected to USB ports
Firewire	External bus. High data transfer rates. Suitable for video devices
IDE (integrated drive electronics)	Interface commonly for hard drives, CD-ROM
SCSI (Small Computer Systems Interface)	Parallel interface. Commonly used for mass storage devices

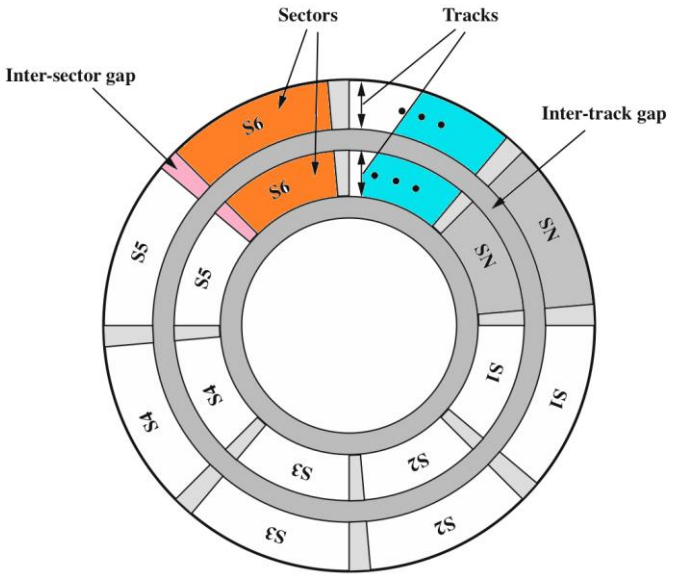
External Storage: Magnetic Disks

- Consists of concentric circles called tracks, divided into sectors.
- Each sector has a unique address
- Each track has same number of sectors and each sector holds same number of bytes.
- One or more metal or glass disks (platters) coated with magnetic material
- Platters are stacked on a spindle which is rotated by motor
- Read/write heads are magnetized with current to write data to disk and magnetized spots can induce current in head to allow data to be read





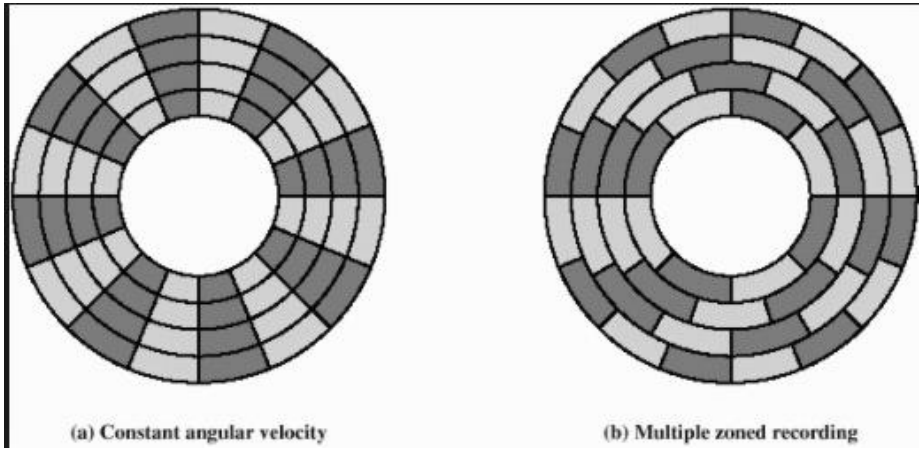
External Storage: Magnetic Disks



- Inter track gap minimizes errors due to misalignment of head/ interference of magnetic fields
- Data transferred to/from disk in sectors
- Several sectors per track either fixed or variable length
- A bit near center travels faster than bit on the outside
- Need to compensate for this so that head can read all bits at same rate e.g. increasing spacing between bits in different segments

Magnetic Disks

- Then scan bits at same speed- constant angular velocity (CAV)
- CAV: individual data blocks can be addressed by track and sector BUT stores same amount of data on long outer track and short inner track
- Density increases towards inner tracks hence limits disk storage capacity
- Increase storage capacity by multiple zone recording
- Divide surface into zones
- In each zone, number of bits per track is constant
- Zones further from center store more bits (more sectors)
- Improves storage capacity BUT more complicated



External Storage

- Magnetic Disks
- circular plate of non-magnetic material (substrate), coated with magnetizable material
- Substrate: aluminum, glass, aluminum alloy
- Read/Write Mechanism: How do magnetic disks work?
- https://www.youtube.com/watch?v=wteUW2sL7bc

Disk Performance Parameters

- ***Seek time***: time taken to position the head at the track
- ***Rotational delay***: After selecting track, required sector needs to line-up to head. Time it takes for the beginning of the sector to reach the head
- ***Access time*** = sum of seek time + rotational delay: time it takes to get into position to read or write
- ***Transfer time***: time required for data transfer (read/write)

Handout

- Mostafa Abd-El-Barr, Hesham El-Rewini, Fundamentals of Computer Organization and Architecture, Chapter 8
- William Stallings, Computer Organization and Architecture, Chapter 3, 7, 6
- Lobur and Null, Essentials of Computer Organization and Architecture, Chapter 7