CMPE 411
Computer Architecture

Lecture 26

Bus Interconnect

November 30, 2017

www.csee.umbc.edu/~younis/CMPE411/CMPE411.htm

Lecture's Overview

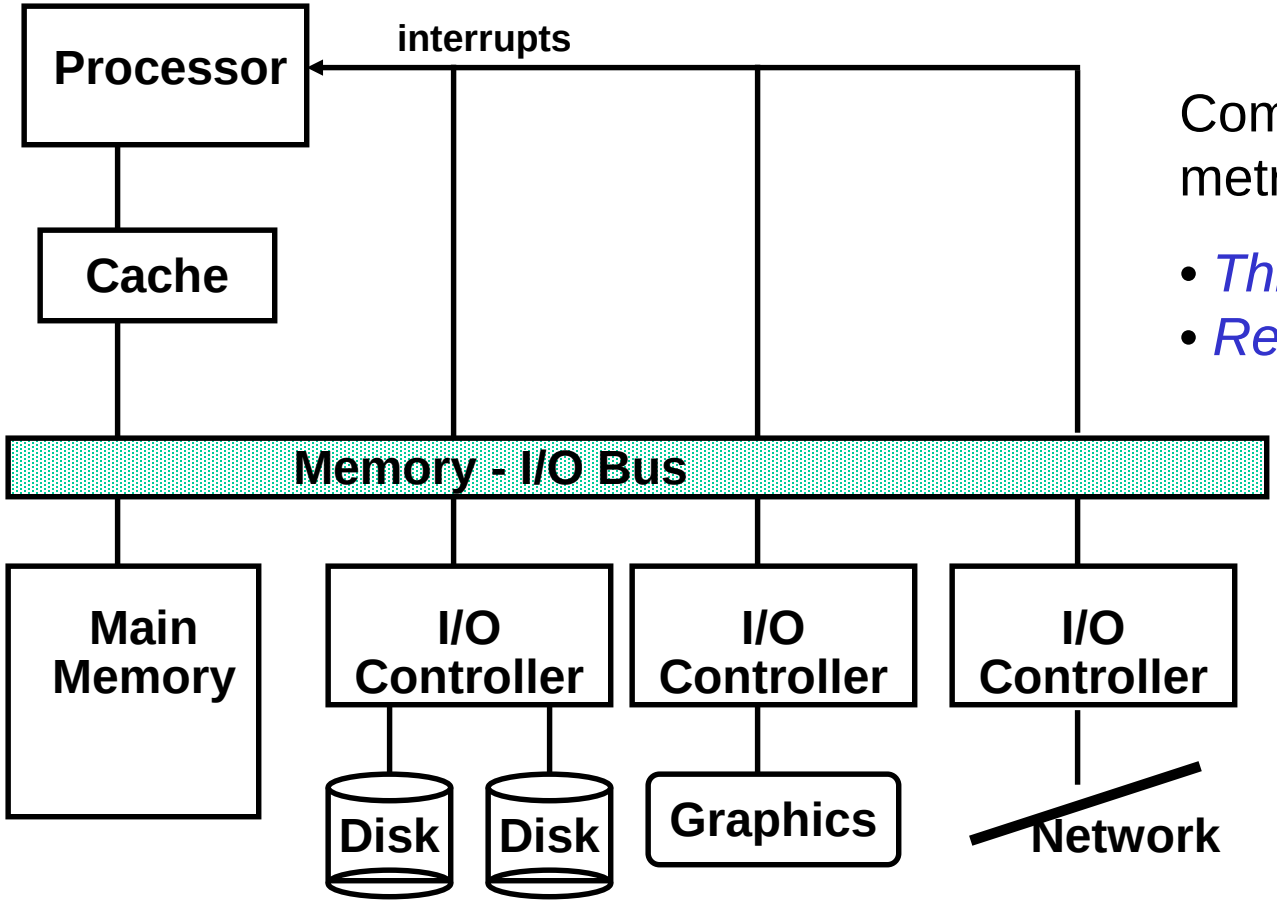
❑ *Previous Lecture:*

- I/O systems architecture
  - ➔ I/O role and interface to the processor
  - ➔ I/O design issues
- I/O devices
  - ➔ Types and characteristics
  - ➔ Performance metrics and optimization factors
  - ➔ I/O benchmarks
- Magnetic Disk
  - ➔ Access time and performance characteristics
  - ➔ Theory of operation and historical trend
  - ➔ Disk non-functional attributes (reliability and availability)

❑ *This Lecture:*

- Memory to processor interconnect

Typical I/O System



Common performance metrics:

- *Throughput*: I/O bandwidth
- *Response time*: Latency

- The connection between the I/O devices, processor, and memory are usually called (local or internal) *buses*
- Communication among the devices and the processor use both bus protocols and interrupts

* Figure is courtesy of Dave Patterson

Connecting I/O Devices

- A bus is a shared communication link, which uses one set of wires to connect multiple subsystems
- The two major advantages of the bus organization are:
  - ➔ *Versatility*: adding new devices and moving current ones among computers
  - ➔ *Low cost*: single set of wires are shared in multiple ways
- The major disadvantage of a bus is that it creates a communication bottleneck possibly limiting throughput
- The maximum bus speed is largely limited by physical factors: the length of the bus and the number of devices
- Increasing bus bandwidth (*throughput*) can be achieved using buffering which slow down bus access (*response time*)
- A bus generally contains:
  - ➔ *control lines*: requests and acknowledge signals
  - ➔ *data lines*: data, addresses, commands
- A bus protocol is enforced to define the semantics of the bus transaction and arbitrate bus usage

Bus Transactions

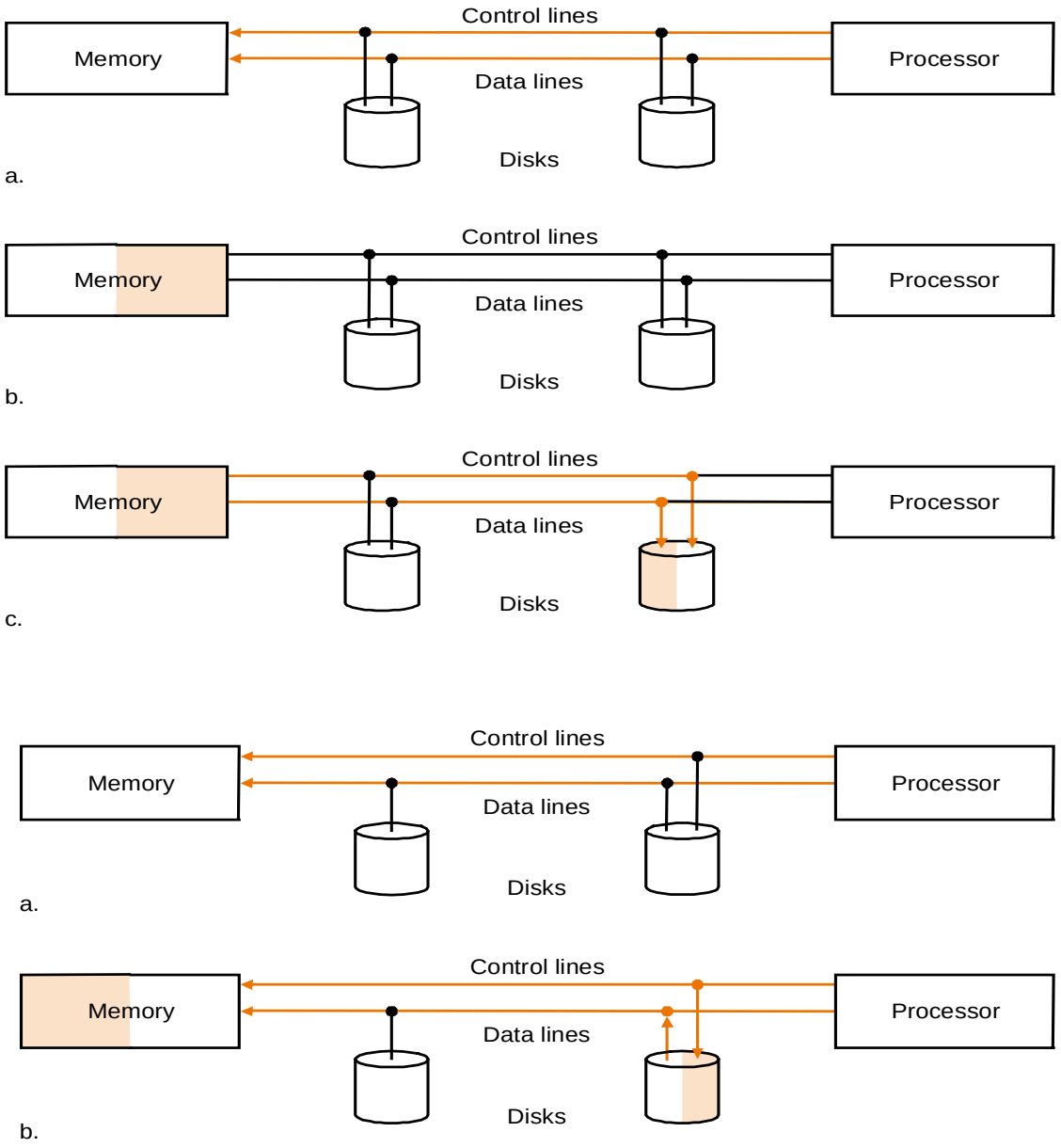
A bus transaction includes two parts:

❶ sending the address

❷ receiving or sending data

Read transaction: transfers data from memory to processor or output device

Write transaction: transfers data from processor or an input device to memory



Types of Buses

❶ ***Processor-memory (local) bus:***

- Generally short and high speed to maximize the bandwidth

❷ ***I/O Bus***

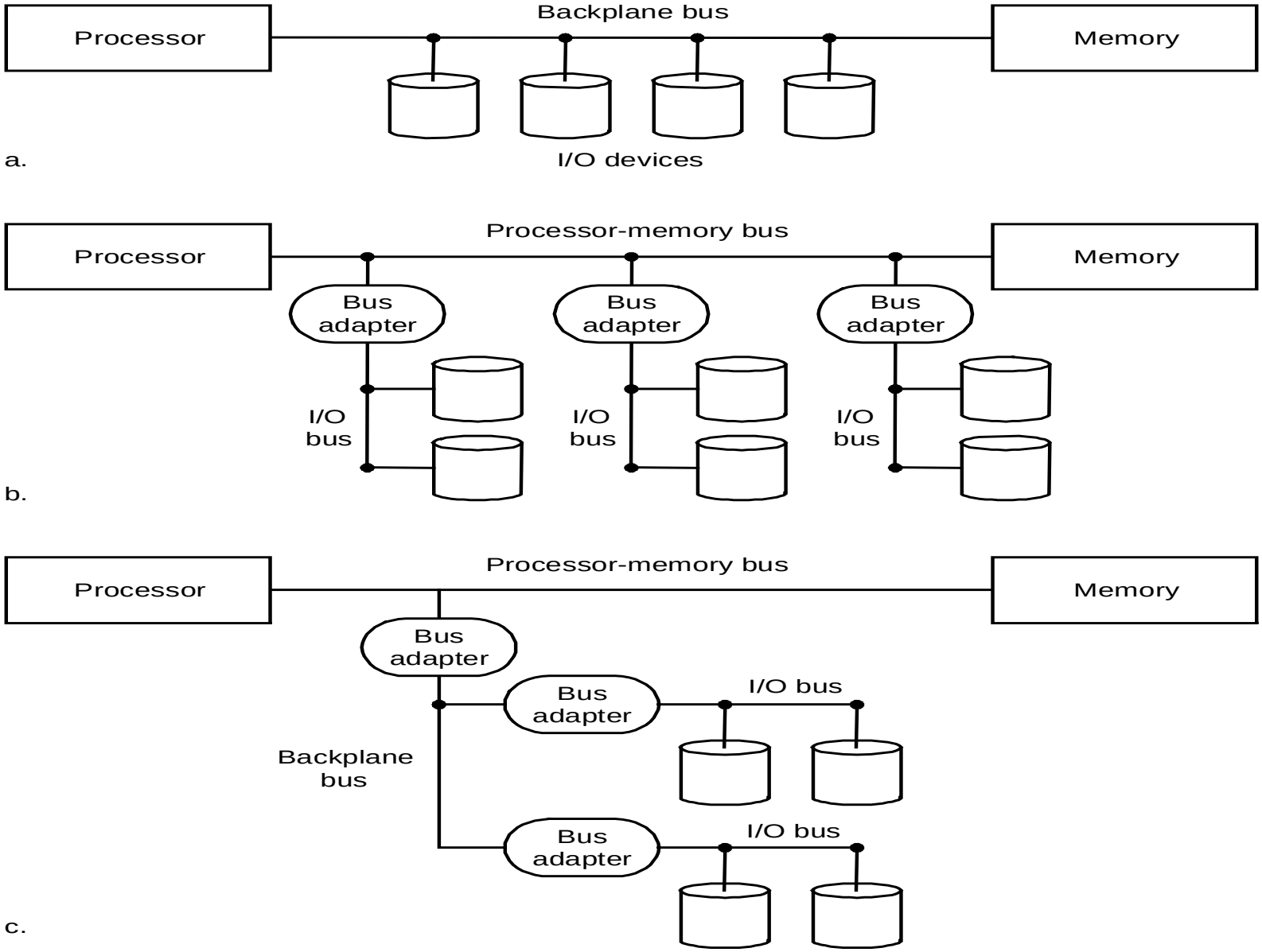
- Lengthy and supportive to multiple data rates and devices
- Do not interface directly to memory but use local or backplane bus
- Must handle wide range of device latency, bandwidth and characteristics

❸ ***Backplane Bus***

- Received that name because they lay in the back of the chassis structure
- Allows memory, processor and I/O devices to be connected
- Requires additional logic to interface to local and I/O buses

- Local buses may be design-specific while I/O and backplane buses are usually portable and often follow industry-recognized standard
- A system can use one backplane or a combination of multiple buses to link memory, processor and the various I/O devices

Bus Configurations



Synchronous vs. Asynchronous Buses

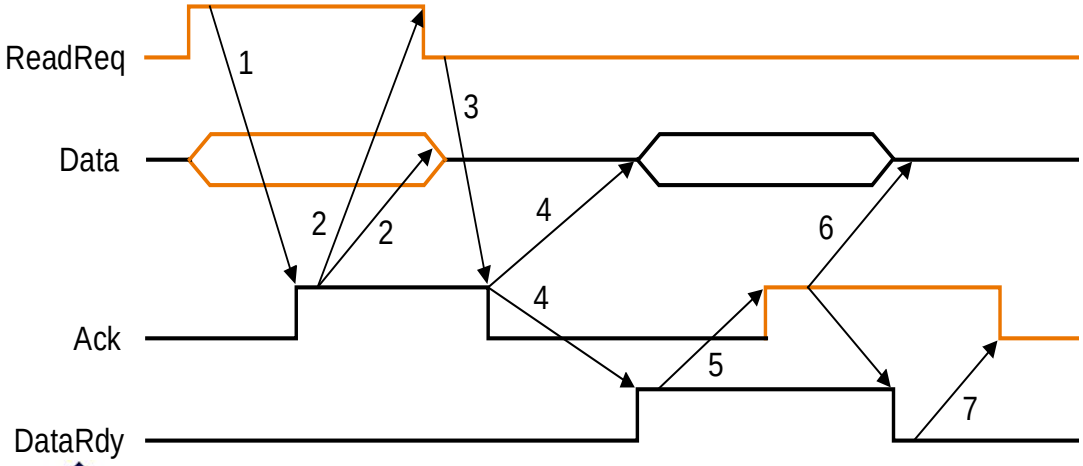
❑ Synchronous Bus:

➔ Clock based protocol and time-based control lines
➔ Simple interface logic and fast bus operations
➔ Every device on the bus must run at the same clock rate
➔ Because clock skew, synchronous buses cannot be long
➔ Local buses are often synchronous

❑ Asynchronous Bus:

➔ Not clock-based → can accommodate a wide variety of devices
➔ Not limited in length because of clock skew
➔ Uses a handshaking protocol to ensure coordination among communicating parties
➔ Requires additional control lines and logic to manage bus transactions

❑ Example:



1. Memory read address when it sees ReadReq
2. I/O device sees Ack and release ReadReq
3. Memory drops Ack when ReadReq is low
4. Memory sets DataRdy when data is available
5. I/O device reads data and set Ack
6. Memory drops DataRdy when sees Ack.
7. I/O device drops Ack When DataRdy is low

Bus Performance (An Example)

One synchronous bus has a clock cycle time of 50 ns with each bus transmission taking 1 clock cycle. Another asynchronous bus requires 40 ns per handshake. The data portion of both is 32-bit wide. Find the bandwidth of each bus for one-word reads from 200-ns memory.

Answer:

The step for the synchronous bus are:

1. Send the address to memory: 50 ns
2. Read the memory: 200 ns
3. Send the data to the device: 50 ns

} 300 ns

The maximum bandwidth is 4 bytes every 300 ns $\Rightarrow$ $\frac{4\text{ bytes}}{300\text{ ns}} = \frac{4\text{ MB}}{0.3\text{ sec}} = 13.3\text{ MB/Sec}$

The step for the asynchronous bus are:

1. Memory read address when seeing ReadReq : 40 ns
2,3,4. Data ready & handshake: max(3×40 ns, 200 ns)= 200 ns
5,6,7. Read & Ack. : 3 × 40 ns = 120 ns

} 360 ns

The maximum bandwidth is 4 bytes every 360 ns $\Rightarrow$ $\frac{4\text{ bytes}}{360\text{ ns}} = \frac{4\text{ MB}}{0.36\text{ sec}} = 11.1\text{MB/Sec}$

Increasing Bus Bandwidth

- Much of bus bandwidth is decided by the protocol and timing characteristics
- The following are other factors for increasing the bandwidth:

Data bus width:

- ➔ Transfers multiple words requires fewer bus cycles
- ➔ Increases the number of bus lines

Multiplexing Address & data line:

- ➔ Uses separate lines for data and address speeds up transactions
- ➔ Simplifies the bus control logic
- ➔ Increases the number of bus lines

Block transfer:

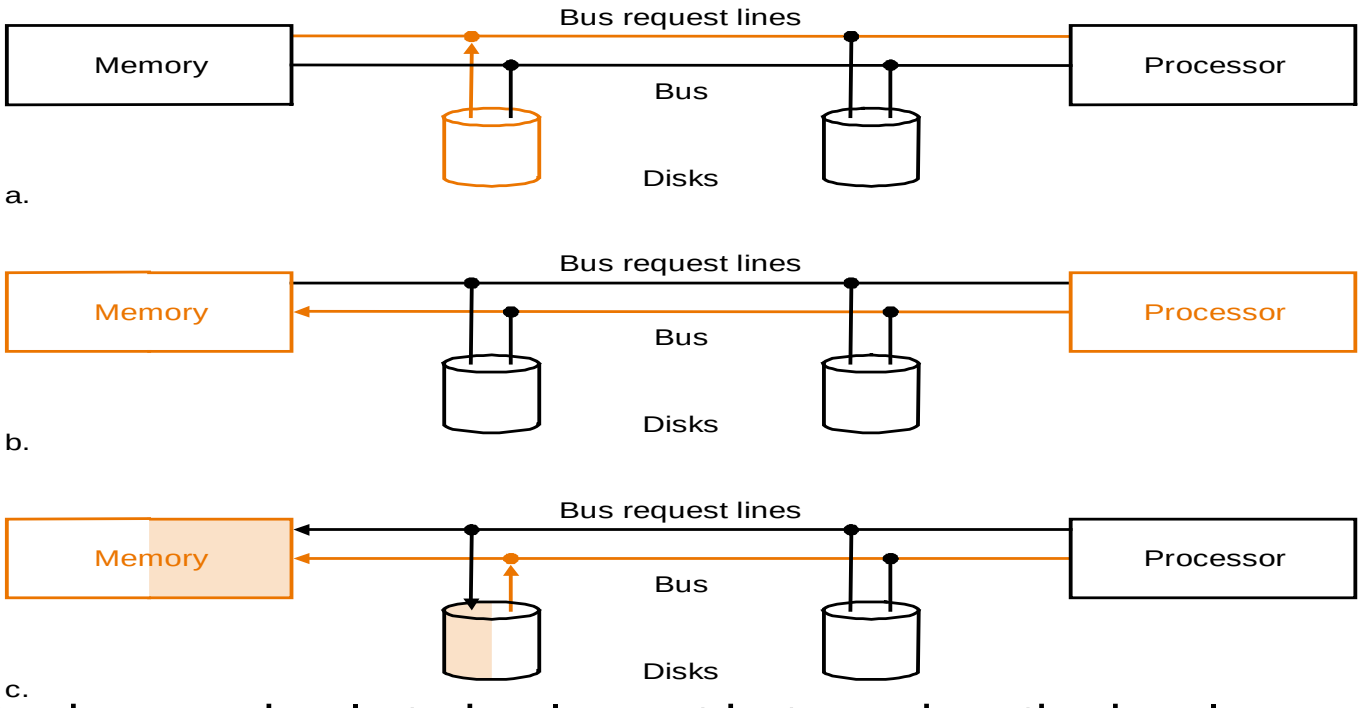
- ➔ Transfers multiple words in back-to-back bus cycles without releasing the bus or sending new address
- ➔ Increases response time since transactions will be longer
- ➔ Increases complexity of the bus control logic

Increasing the number of bus lines has very significant impact on the bus cost

Bus Access

- I/O should occur without processor's continuous and low-level involvement
- A bus master, e.g., processor, controls access to the bus and initiates and manage bus requests
- A bus slave, e.g. memory, only responds to access requests but never initiate its own requests

Single master bus transaction



- A single bus master is very simple to implement but requires the involvement of the processor in every transaction
- Multi-master buses requires arbitration to coordinate bus usage among potential access requesters

Bus Arbitration

- Bus arbitration coordinates bus usage among multiple devices using *request*, *grant*, *release* mechanism
- Arbitration usually tries to balance two factors in picking the granted device:
  - ➔ Devices with high *bus-priority* should be served first
  - ➔ Maintaining *fairness* to ensure that no device will be locked out from the bus
- Arbitration time is an overhead

  $\Rightarrow$ should be minimized to expedite bus access

Arbitration Schemes

Distributed arbitration by self-selection: (e.g. NuBus used on Apple Macintosh)

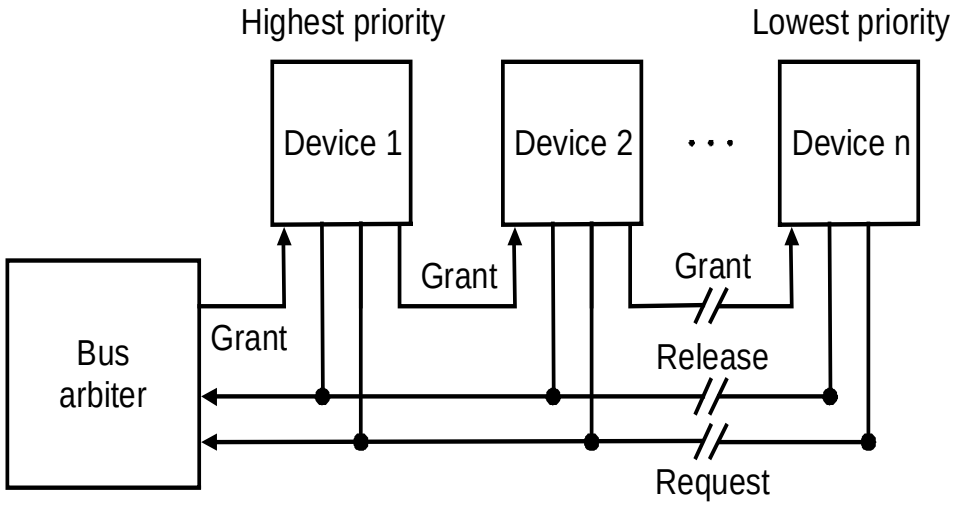
- ➔ Uses multiple request lines for devices
- ➔ Devices requesting the bus determine who will be granted access
- ➔ Devices associate a code with their request indicating their priority
- ➔ Low priority devices leave the bus if they notice a high priority requester

Distributed arbitration by collision detection: (e.g. Ethernet)

- ➔ Devices independently request the bus and assume access
- ➔ Simultaneous requests result in a collision
- ➔ A scheme for selecting among colliding parties is used

Arbitration Schemes (Cont.)

Daisy chain arbitration: (e.g. VME bus)



➔ The bus grant line runs through devices from highest priority to lowest

➔ High priority devices simply intercept the grant signal and prevent the low priority device from seeing the signal

➔ Simple to implement (use edge triggered bus granting)

➔ Low priority devices can starve (some designs prevent bus usage in two successive clock cycles)

➔ Limits the bus speed (grant signals take long to reach the last device in the chain)

➔ Allow fault propagation (failure of a high priority device might lock the bus)

Centralized, parallel arbitration: (e.g. PCI bus)

➔ Uses multiple request-lines and devices independently request bus (one line / device)

➔ A centralized arbiter selects a device and notifies it to be the bus master

➔ The arbiter can be a bottleneck for bus usage

Bus Standards

- I/O bus serves as a way of expanding the machine and connecting new peripherals
- Standardizing the bus specifications ensure compatibility and portability of peripherals among different computers
- Popularity of a machine can make its I/O bus a de facto standard, e.g. IBM PC-AT bus
- Examples of widely known bus standards are Small Computer Systems Interface (SCSI), Ethernet, and Peripheral Computer Interface (PCI)

Characteristics	PCI	SCSI
Bus type	Backplane	I/O
Basic data bus width	32-64	8-32
Address/data multiplexied?	Yes	Yes
Single / Multiple bus masters	Multiple	Multiple
Arbitration	Centralized. Parallel arbitration	Self-select
Clocking	Synchronous 33-66 MHz	Asynchronous or Synchronous (5-10 MHz)
Theoretical peak bandwidth	133-512 MB/sec	5-40 MB/sec
Estimating typical achievable bandwidth of basic bus	80 MB/sec	2.5-4.0 MB/sec (synchronous) or 1.5 MB/sec (asychronous)
Maximum number of devices	1024 (for multiple bus segments, with 32 device / bus segment)	7-31 (bus width -1)
Maximum bus length	0.5 meter	25 meters
Standard name	PCI (Intel)	ANSI X3.131

Conclusion

- ***Summary***
  - ➔ Memory to processor interconnect
    - Definition of bus structure
    - Bus transactions
    - Types of buses
    - Bus Standards
  - ➔ Bus Performance and Protocol
    - Synchronous versus Asynchronous buses
    - Bandwidth optimization factors
  - ➔ Bus Access
    - Single versus multiple master bus
    - Bus arbitration approaches
- ***Next Lecture***
  - ➔ Interfacing I/O devices to memory, processor and OS