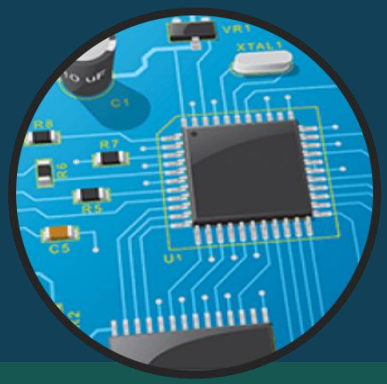
Computer Buses

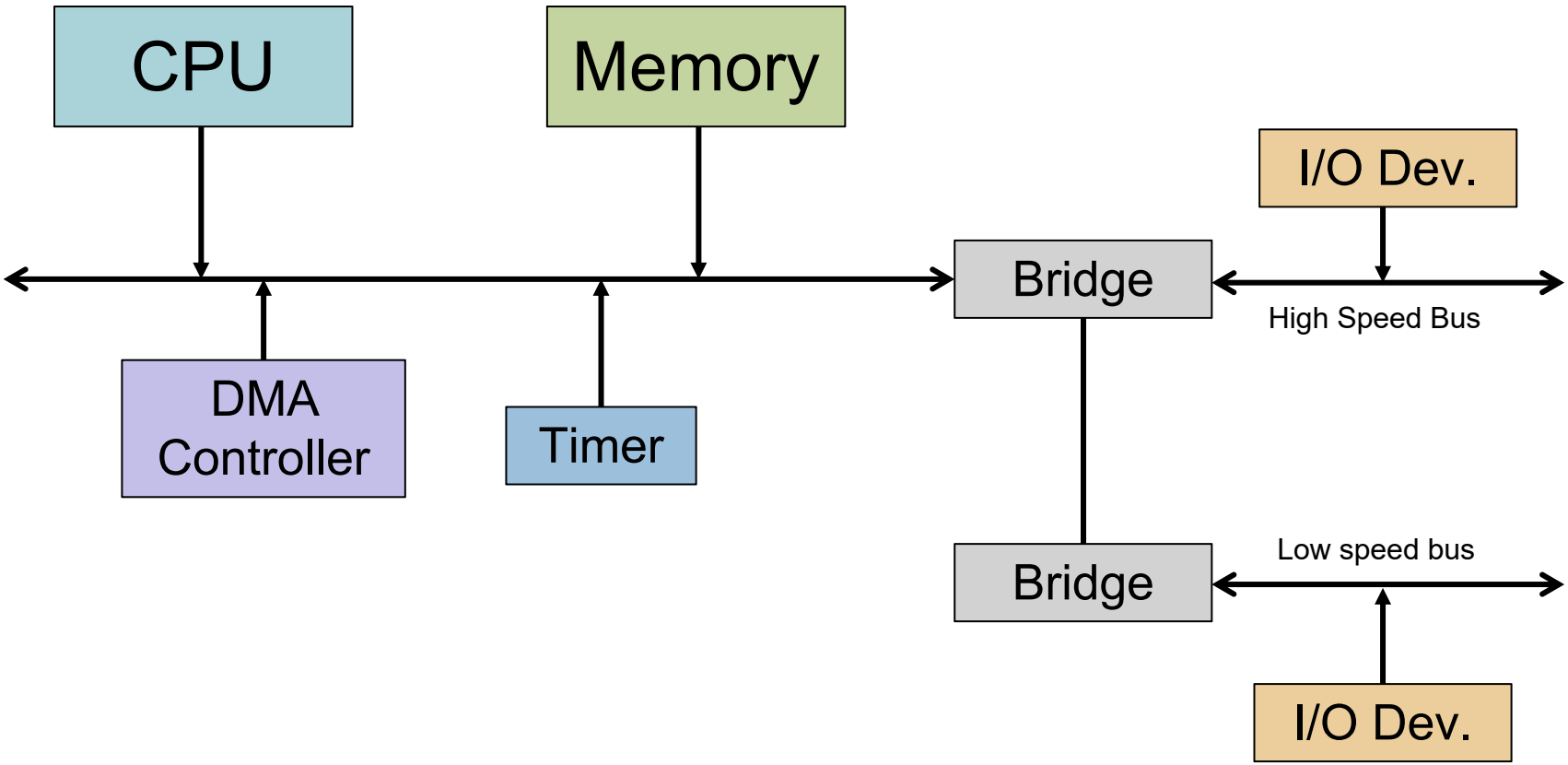


Topic # 9

Fall 2019

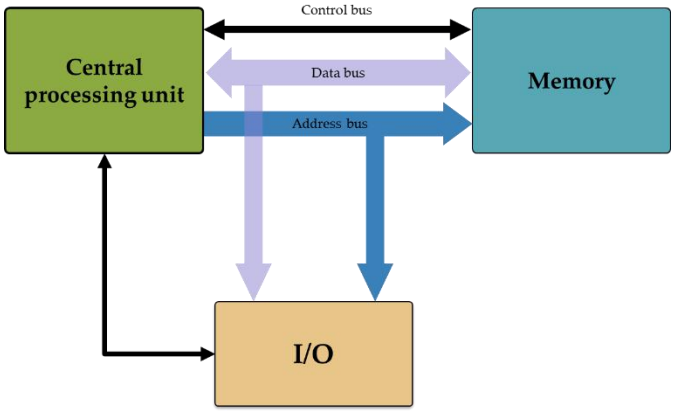
Engr. Taufique-ur-Rehman <taufique.rehman@seecs.edu.pk>
Muhammad Imran Abeel <imran.abeel@seecs.edu.pk>

Interconnections



BUS

- **Shared Communication Link**
- **Single set of wires used to connect Multiple Subsystems**



➢ **Single Function Wires (Non-multiplexed)**

- Address Bus, Data Bus

➢ **Multiplexed Function Lines**

- Control + Address/Data
- Associated Protocol: Rules for Communication

What Defines a Bus?

- Transaction protocol
- Timing and signaling specification
- Bunch of wires
- Electrical Specification
- Physical and mechanical characteristics

What Defines a Bus?

- **Communication bottleneck**
  - Bus bandwidth limits the maximum I/O throughput.
- **Maximum speed limit by**
  - Length of the bus
  - Number of devices
  - Complexity : support devices with
    - Widely varying latencies
    - Widely varying data transfer rates

Types of Buses

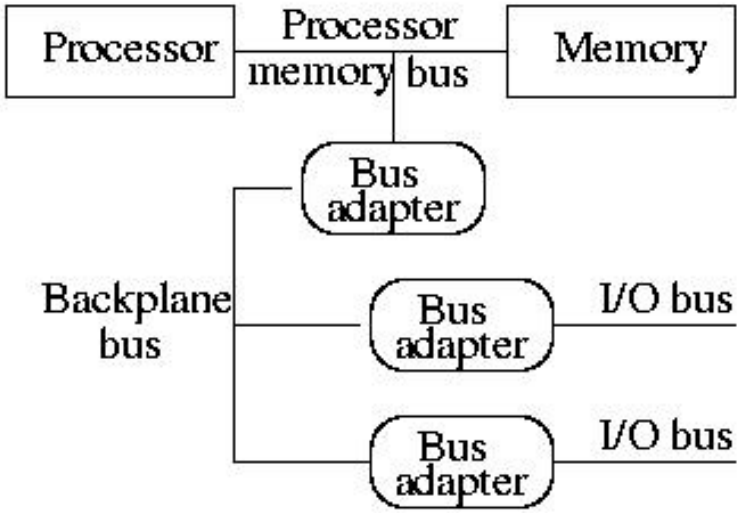
- **Processor-Memory Bus (Front Side Bus)**
  - Short and high speed
  - Only need to match the memory system
    - Maximize memory to processor bandwidth
  - Connects directly to the processor
  - Optimized for cache block transfers

Types of Buses

- **I/O Bus**
  - Usually is length and slower
  - Need to match a wide range of I/O devices
  - Connects to the processor-memory bus or backplane bus
  - ISA, PCI, USB, AGP

Types of Buses

- **Backplane Bus**
  - Allow processors, memory and I/O devices to coexist.



  - Cost advantage: one bus for all components

Sync. vs Async. Bus

Synchronous Bus

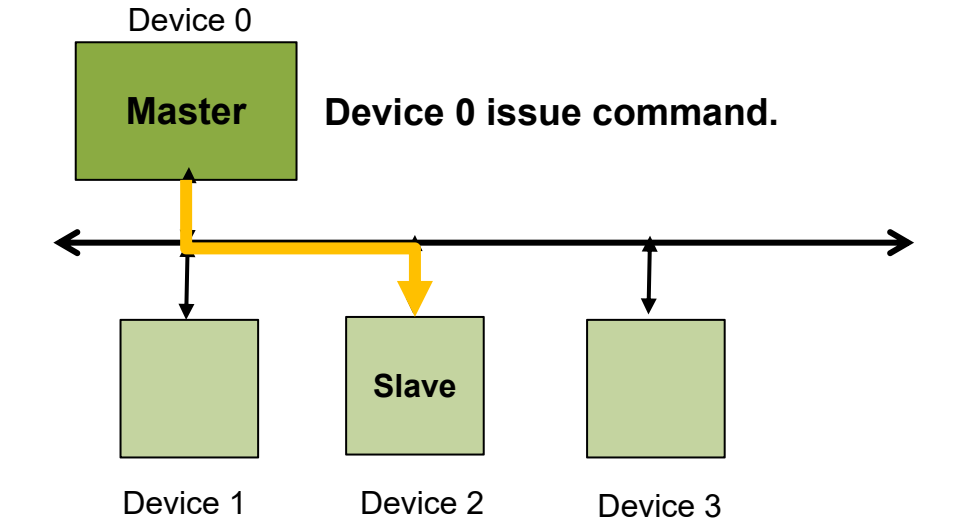
- Clock for all devices in control lines
- Fixed protocol: relative to clock signal
- Simple protocol: very fast.
- **Drawbacks**
  - All devices must run at the same speed
  - The bus must be short due to clock skew

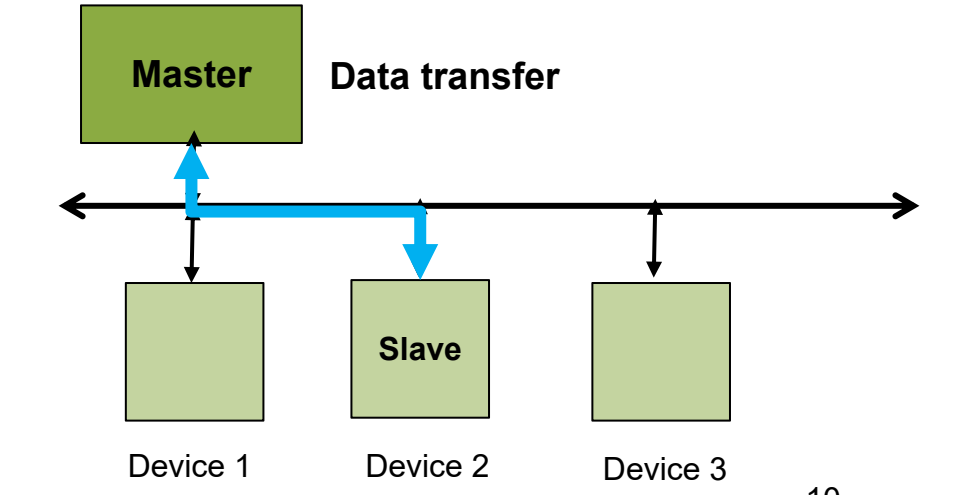
Asynchronous Bus

- Not clocked.
- Varying speed devices can be on the same bus
- no problem with clock skew
- bus must now contain control lines and handshaking protocol.
- **Drawbacks**
  - Slow
  - Complex

Master & Slave

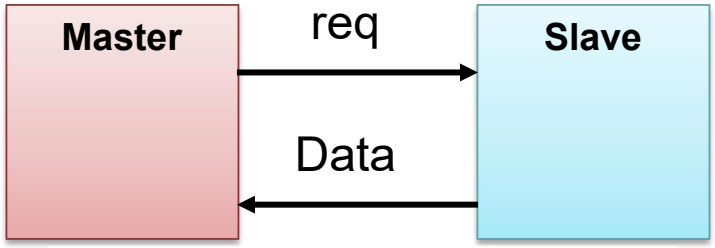
- Master device: Initiates process by issuing;
  - command and address.
- Salve receives command and respond to master request;
  - Sending/Receiving data
- Address:
  - Location in memory
  - Peripheral
  - Register within peripherals.

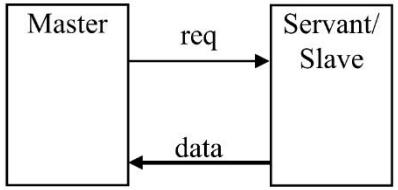


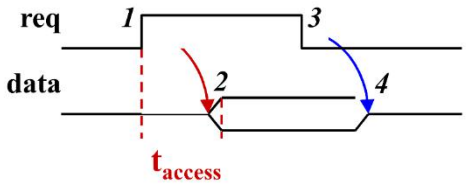


Master & Slave

Strobe protocol

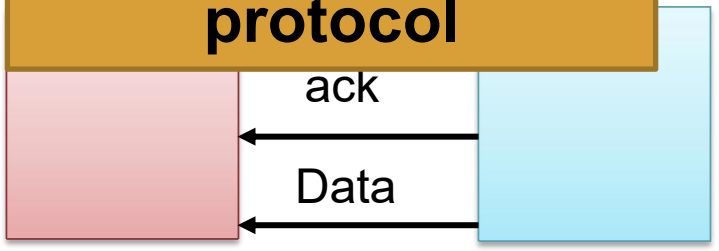


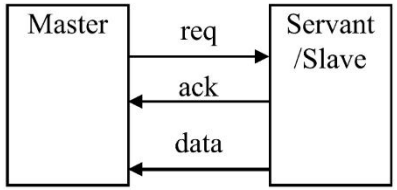


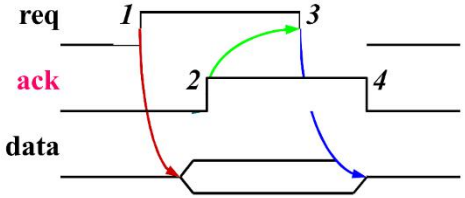


1. Master asserts req to receive data
2. Servant put data on bus within time taccess
3. Master receives data and de-asserts req
4. Slave ready for next request

Handshake protocol

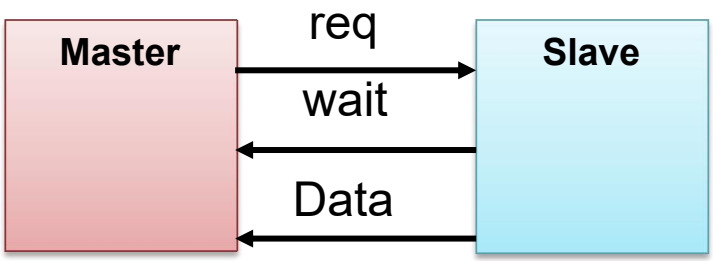




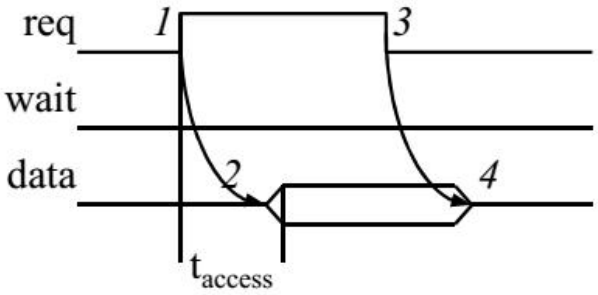


1. Master asserts req to receive data
2. Servant put data on bus and assert ack
3. Master receives data and de-asserts req
4. Slave ready for next request

Strobe/ Handshake Protocol

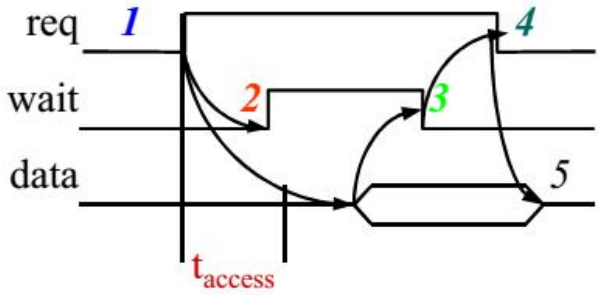


Fast response case



- Master asserts req to receive data
- Servant put data on bus within time taccess
- Master receives data and de-asserts req
- Slave ready for next request

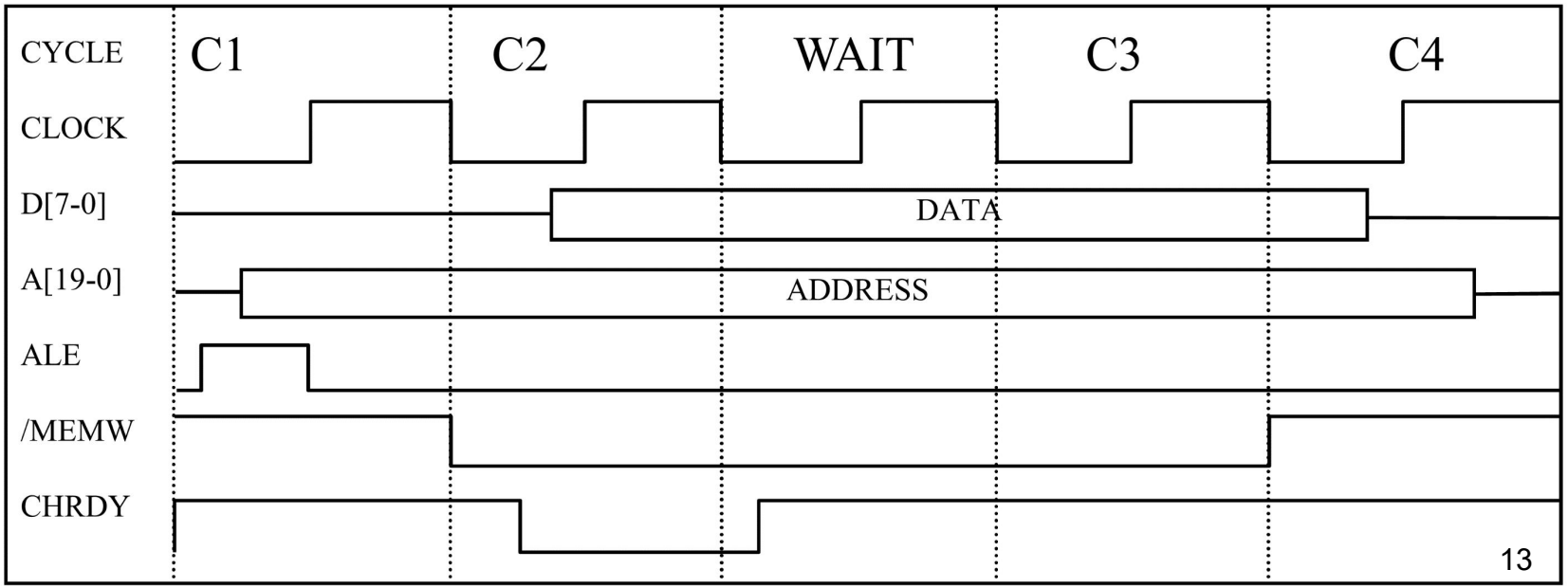
slow response case



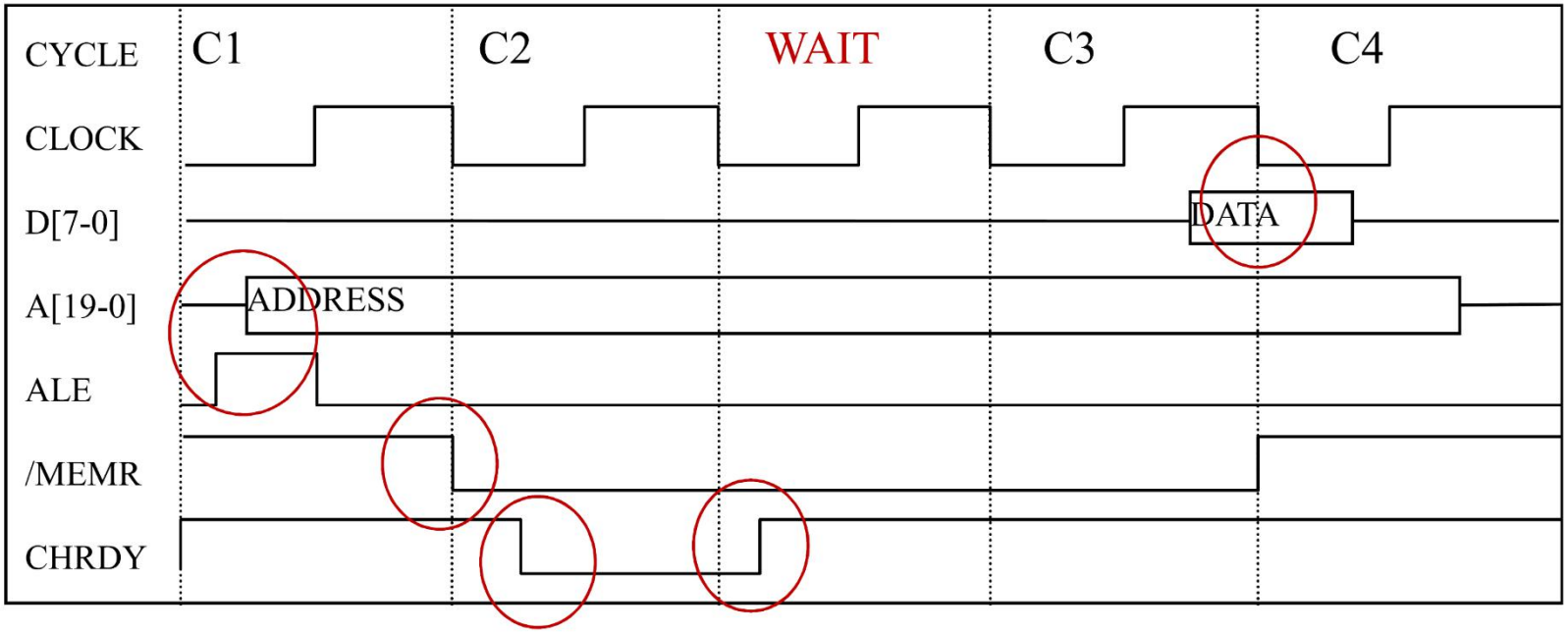
- Master asserts req to receive data
- cannot put data within taccess, assert wait
- Servant put data on bus and assert ack
- Master receives data and de-asserts req
- Slave ready for next request

Example: ISA Bus Protocol

- ISA: Industry standard architecture
  - Common in 80x86's
- Features
  - 20-bit address
  - Compromise strobe/handshake control

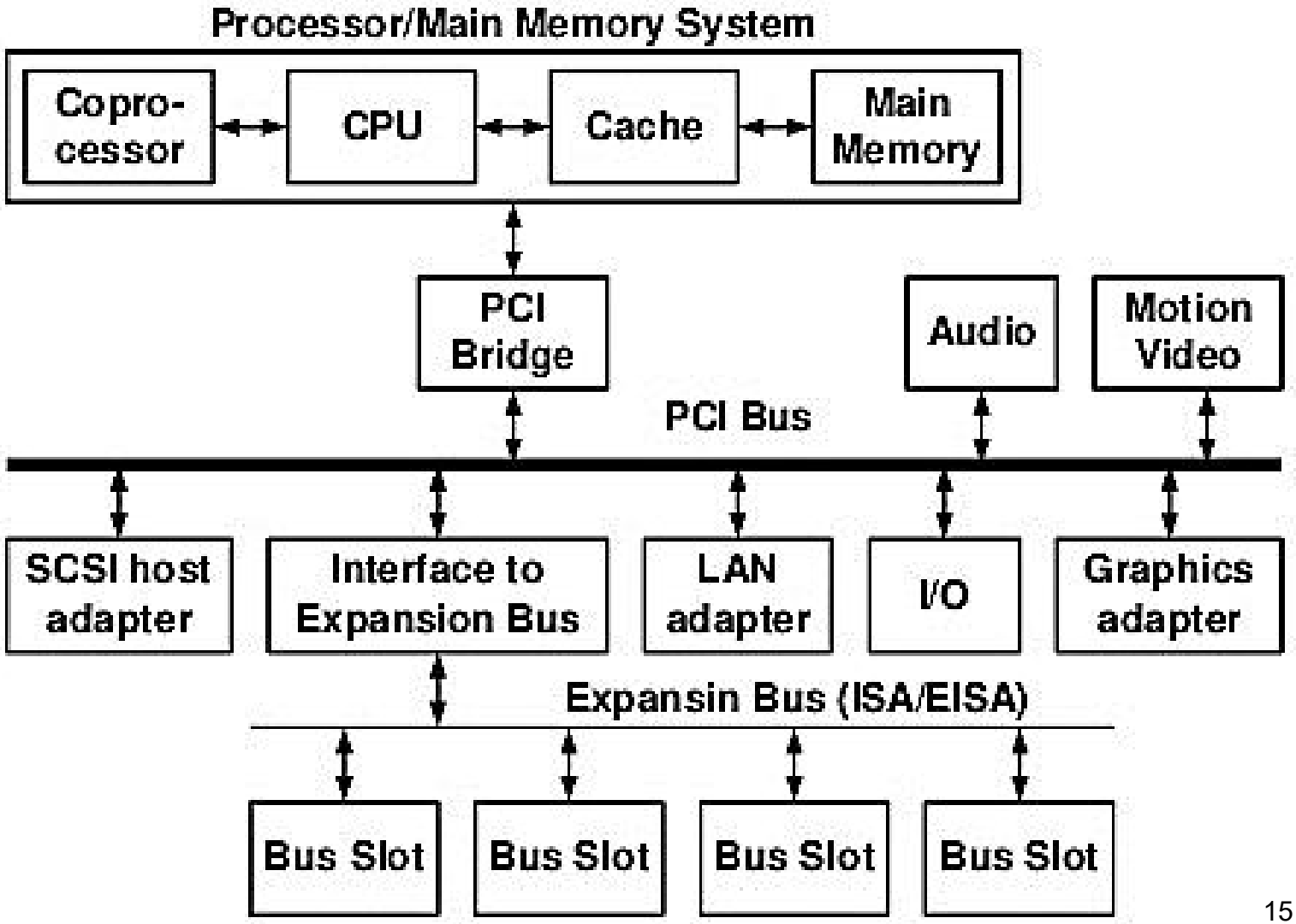


ISA Bus Protocol: Memory Read with Wait

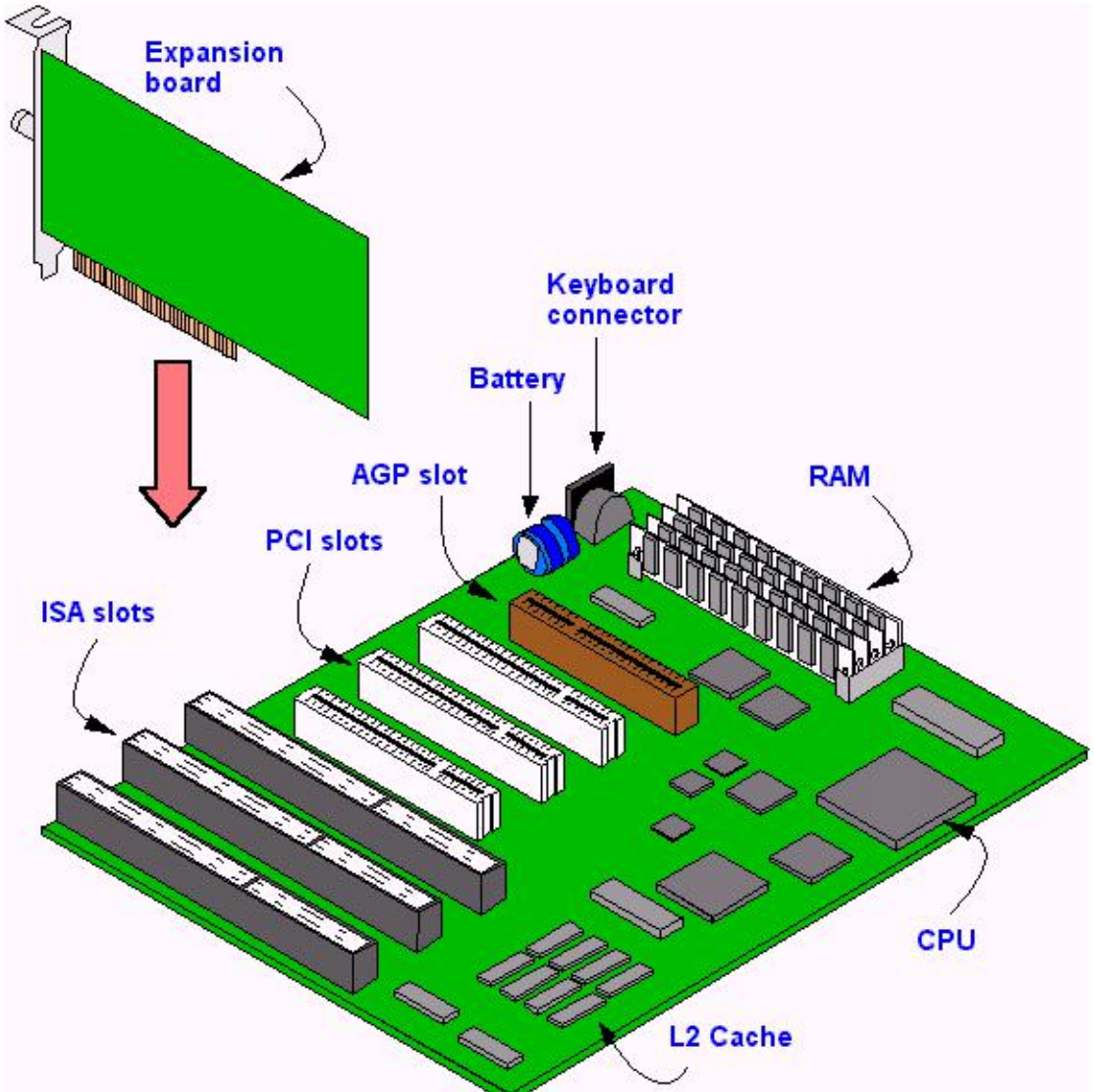


- 4 cycles default
- Unless CHRDY deasserted – resulting in additional wait cycles (up to 6)

PCI-e Bus System Block Diagram



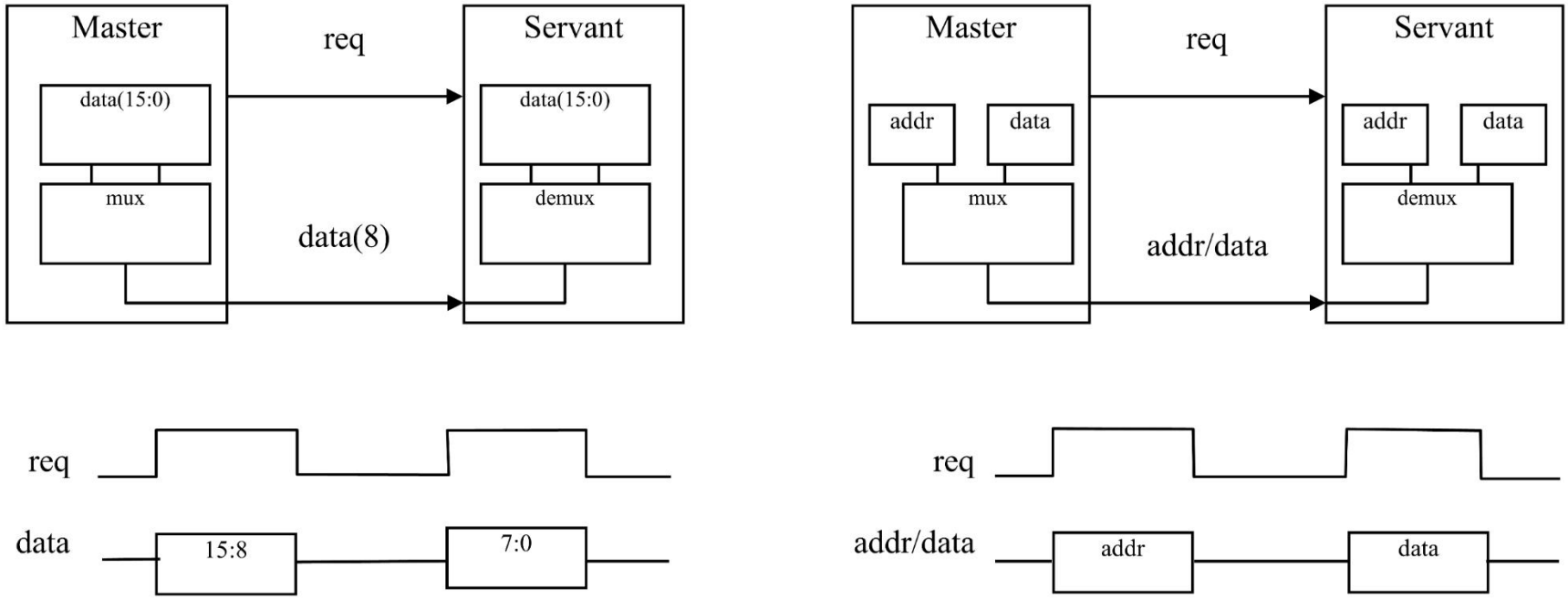
Bus Locations on Mother Board



ISA Bus Protocol: Memory Read with Wait

- Time multiplexing
  - Share a single set of wires for multiple pieces of data
  - Saves wires at expense of time

Time-multiplexed data transfer



data serializing

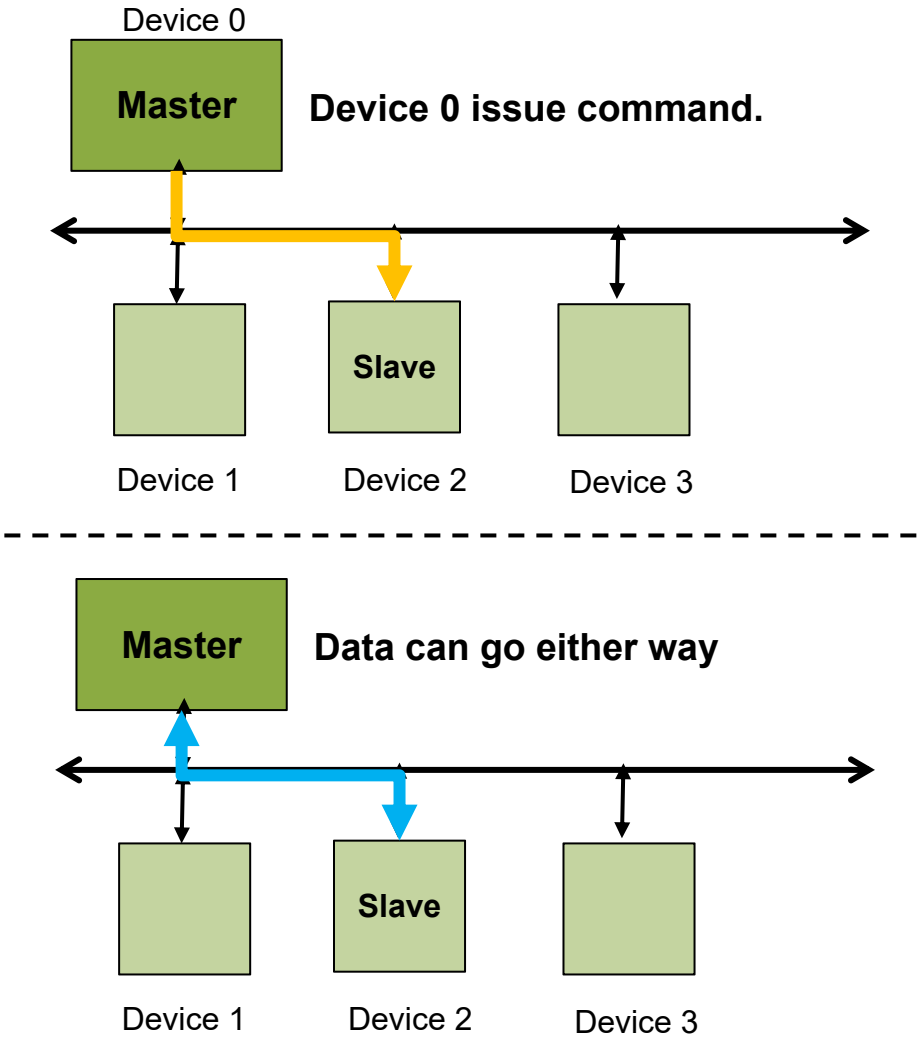
address/data muxing

Bus Arbitration

- One of the most important issues in bus design:
- How is a bus reserved by a device that wants to use it?

- Chaos is avoided by master slave management:

- Only the bus master can control access to the bus: it initiates and control all bus request
- A slave responds to read and write requests

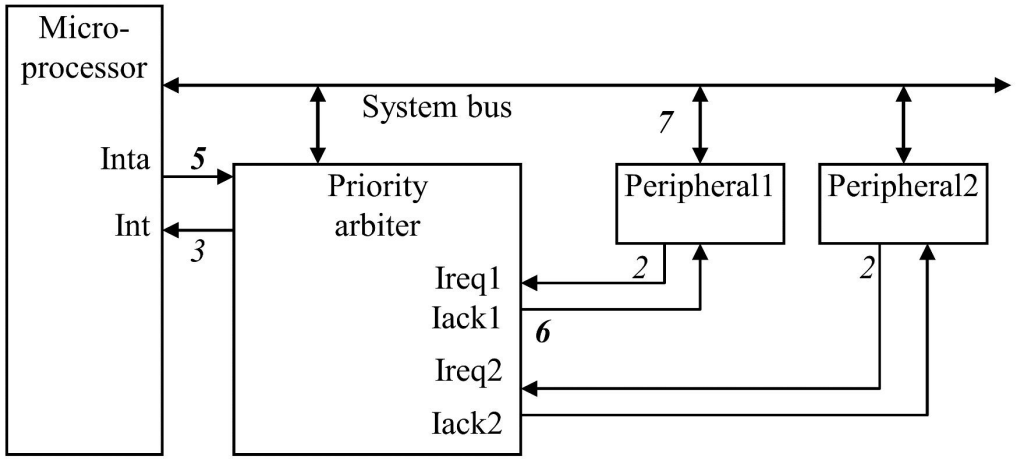


Multiple Bus Master Arbitration

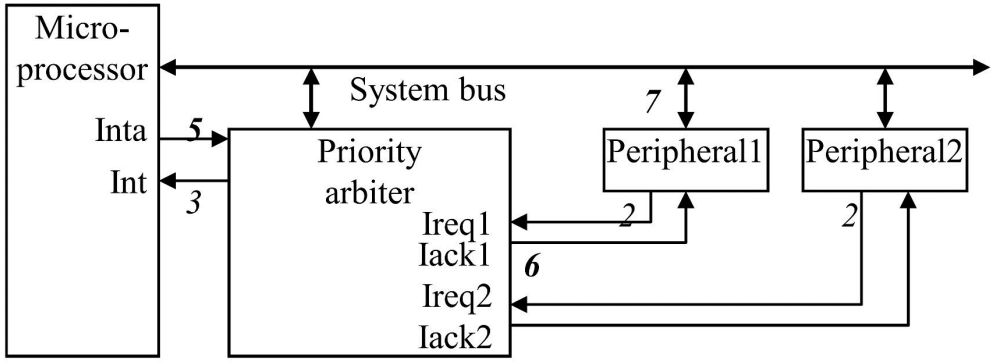
- Bus arbitration scheme:
  - A bus master wanting to use the bus asserts the bus request
  - A bus master cannot use the bus until its request is granted
  - A bus master must signal to the arbiter after finish using the bus
- Bus arbitration schemes usually try to balance two factors:
  - Bus priority: the highest priority device should be serviced first
  - Fairness: Even the lowest priority device should never be completely locked out from the bus

Priority Arbitration

- Priority arbiter
  - Single-purpose processor
  - Peripherals make requests to arbiter, arbiter makes requests to resource
  - Arbiter connected to system bus for configuration only



Priority Arbitrator



- 1. Microprocessor is executing its program.
- 2. Peripheral1 needs servicing so asserts *Ireq1*. Peripheral2 also needs servicing so asserts *Ireq2*.
- 3. Priority arbiter sees at least one *Ireq* input asserted, so asserts *Int*.
- 4. Microprocessor stops executing its program and stores its state.
- 5. Microprocessor asserts *Inta*.
- 6. Priority arbiter asserts *Iack1* to acknowledge Peripheral1.
- 7. Peripheral1 puts its interrupt address vector on the system bus
- 8. Microprocessor jumps to the address of ISR read from data bus, ISR executes and returns
- (and completes handshake with arbiter).
- 9. Microprocessor resumes executing its program.

Priority Arbitrator: Types of Priority

- Fixed priority
  - each peripheral has unique rank
  - highest rank chosen first with simultaneous requests
  - preferred when clear difference in rank between peripherals
- Rotating priority (round-robin)
  - priority changed based on history of servicing
  - better distribution of servicing especially among peripherals with similar priority demands

Priority Arbitrator: Types of Priority

- When multiple microprocessors share a bus
  - Arbitration typically built into bus protocol
  - Separate processors may try to write simultaneously causing collisions
- Example:
  - I2C (Inter-Integrated Circuit); multi-master serial computer bus))
  - Ethernet
  - Multiple processors may write to bus simultaneously. If collision occurs ; processors wait for some random time & then resend the data

Acknowledgement

Muhammad Jameel Nawaz, SEECS, NUST

Questions?

Thank You!