Memory Architectures for NoC-Based Real-Time Mixed Criticality Systems

Neil Audsley

Real-Time Systems Group
Computer Science Department
University of York
York
United Kingdom

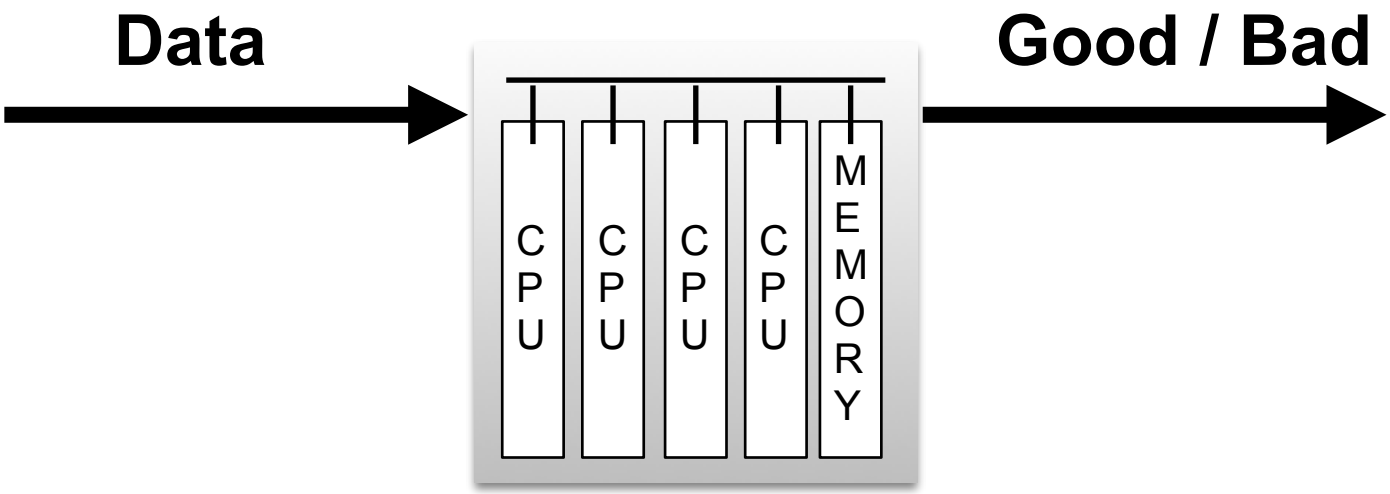
Overview

- Motivation:

  Examine provision of memory hierarchy for MC

- Structure:
  1. A Story
  2. Memory Hierarchy
  3. Colourful Digression
  4. Issues / Requirements
  5. Memory Tree for NoC

A Story ….



Access to memory strictly TDMA

Not allowed to compute application if dont have TDMA slot

Fundamental Trade-off

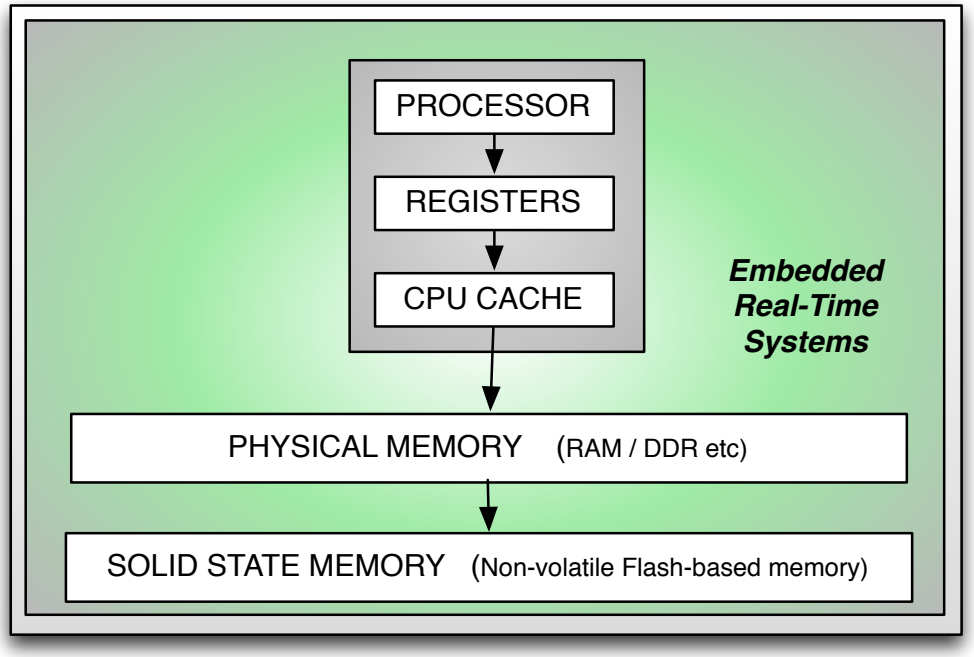
Efficient Resource Usage

v

Physical Resource Separation

- Must maintain Safety Properties - *stop bad things happening*
  - Hard for memory
    - Issues of single fault (bit flip) causing system failure, correct MMUs …
  - Need performance
    - Especially as we move to many / multi-core architectures

Memory Hierarchy



Increasing Latency

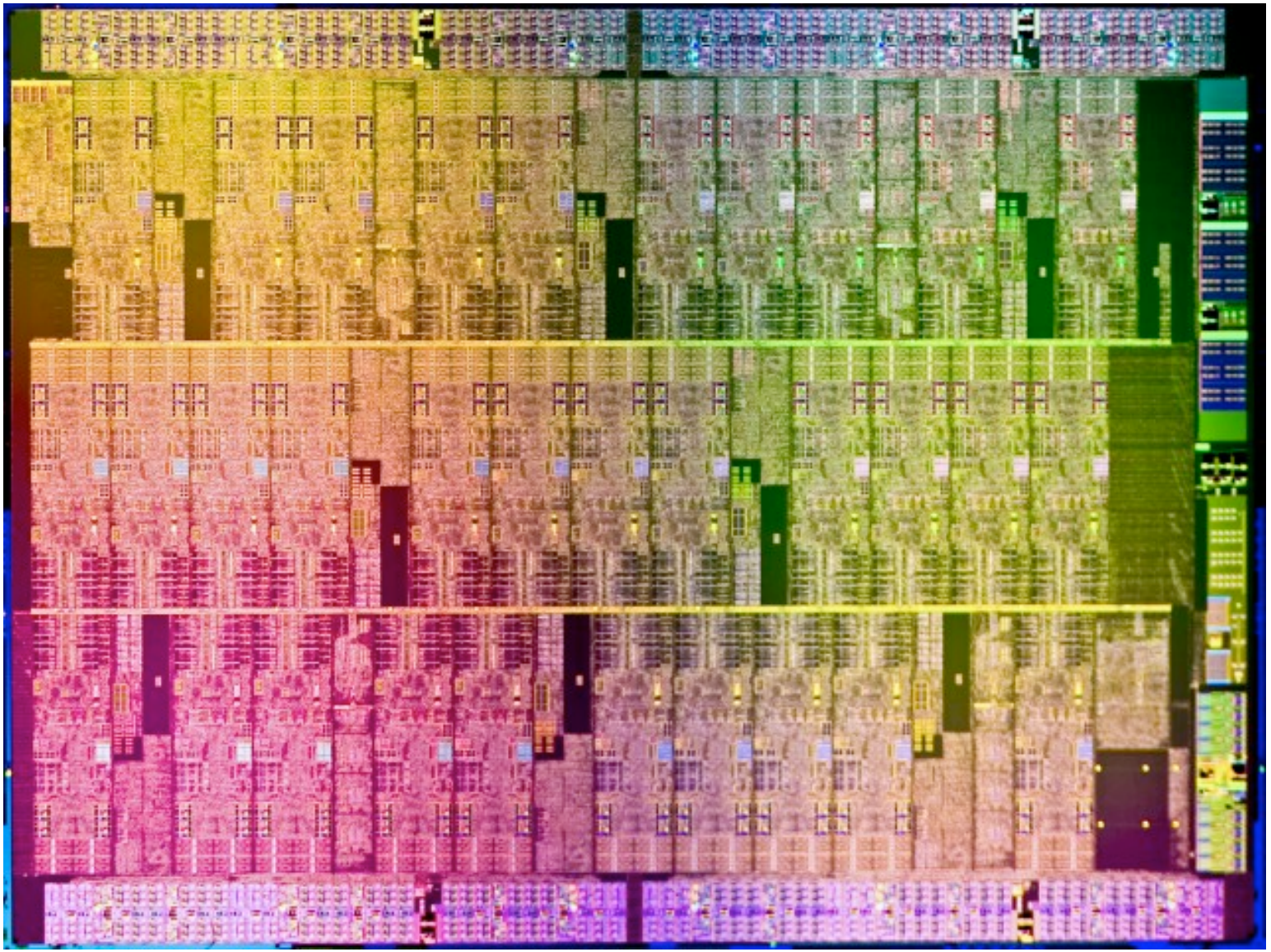
Worst-Case Execution Time increasingly hard to calculate:

- *Caches*
- *Shared memory*
- *File system access*

Memory and Mixed Criticality

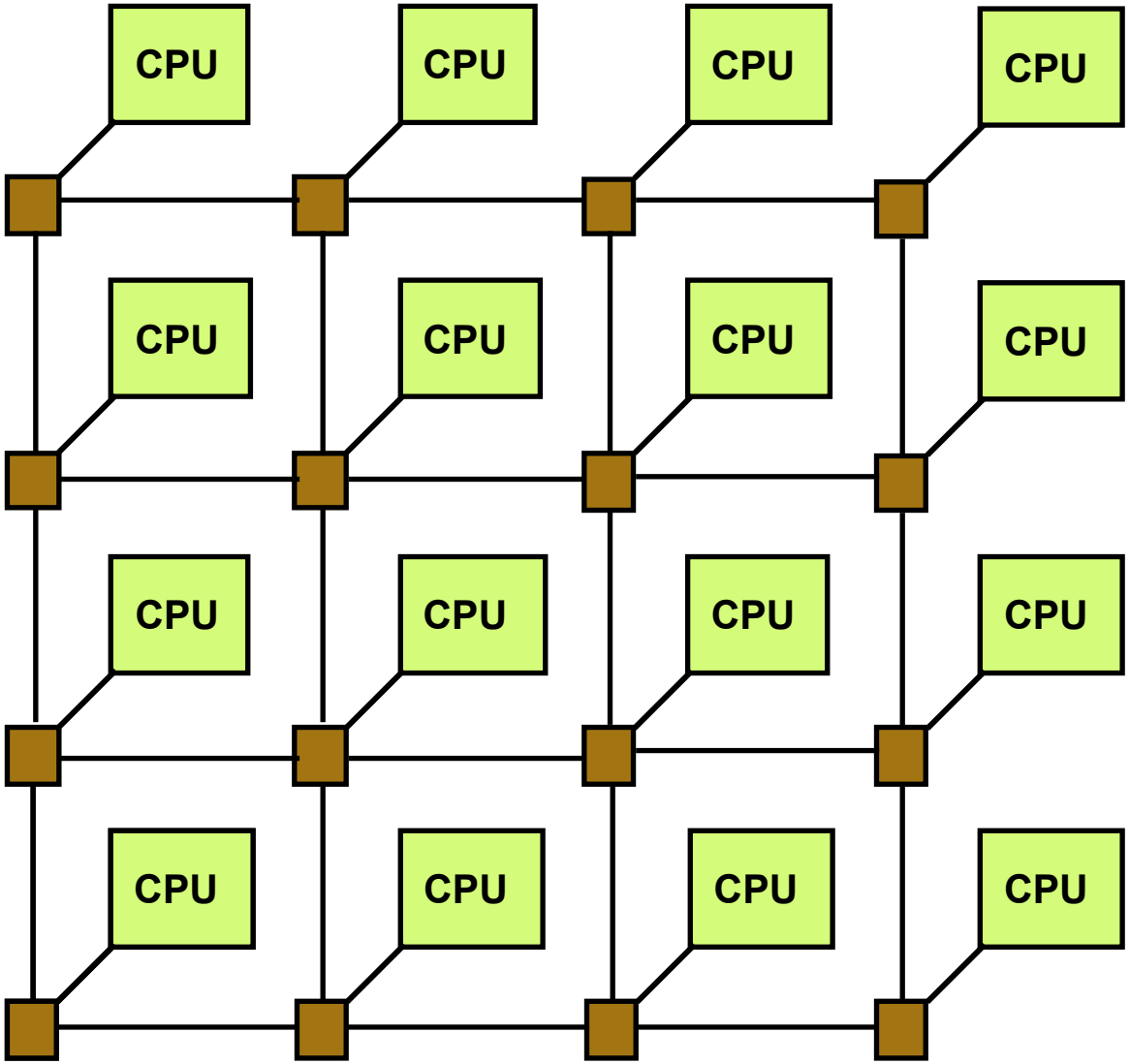
- Sufficient physical partitioning to meet safety requirements
  - Impact upon system architecture
- MC analysis - *WCET up as criticality increases*
  - Exploit "*pessimism*" in modelling / analysis for WCET
    - System architecture unchanged
  - And / Or allow architecture to provide increased physical separation as criticality increases
    - Increase conservatism - increase latencies / end-to-end memory access times

Intel Xeon Phi *(Knights Landing)*



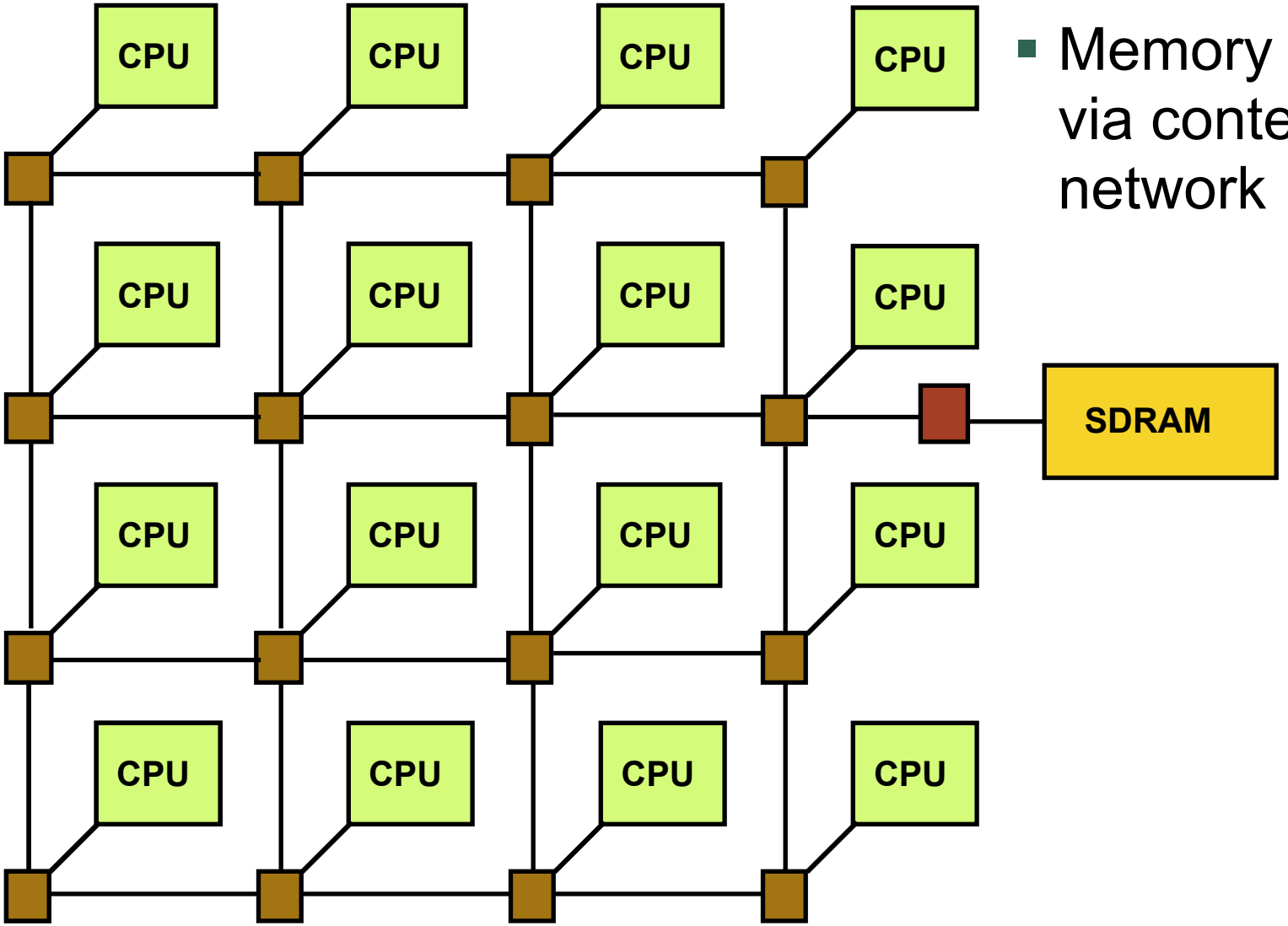
- Regular structure
- Replicated CPU unit
- Not suited to *shared bus*
- Many other examples

Network-on-Chip Architecture



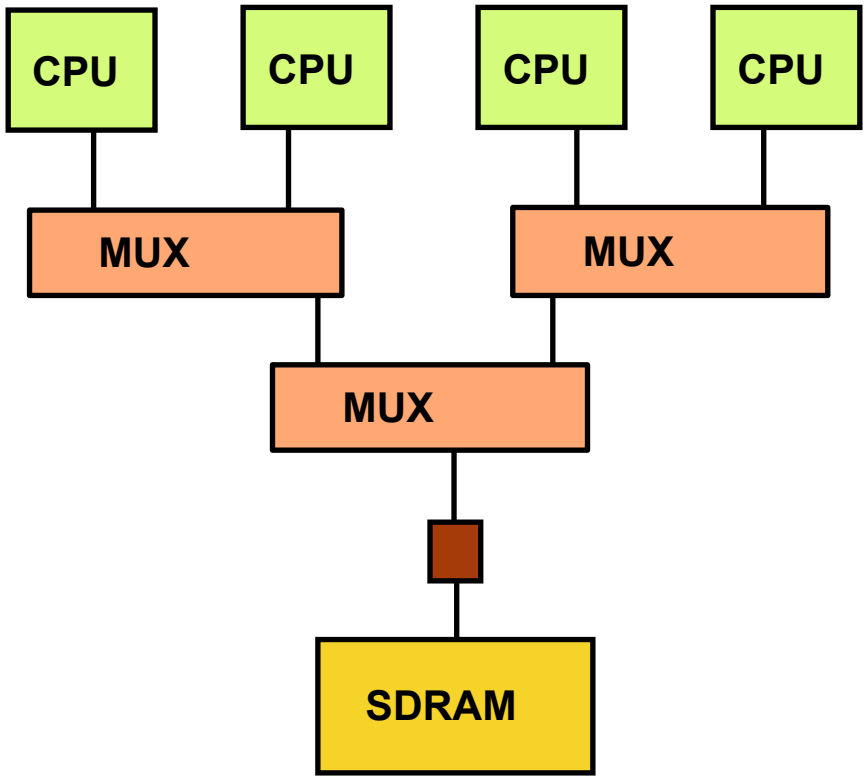
- Network is mesh
  - Arbiters / Routers
  - Local connections between CPU and arbiter
- Packet Switched
  - Worm-hole routing prevalent
- CPU local memory
  - Often assume local memory big enough for all code / data
    - no cache misses

Network-on-Chip Architecture - *External Memory*



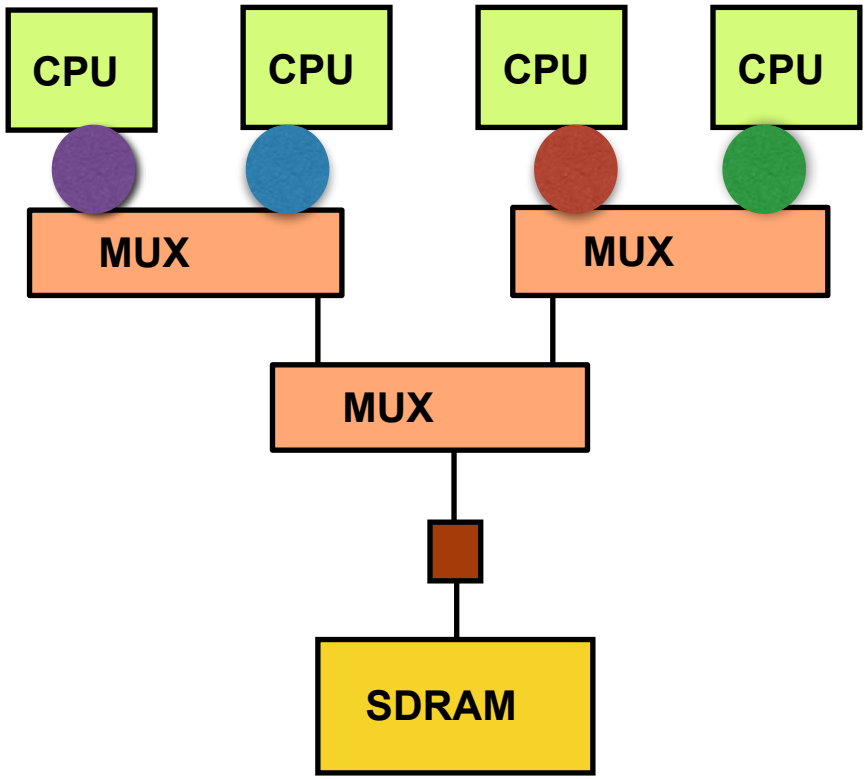
- Memory transactions via contended network

Network-on-Chip Architecture - *Memory Tree*



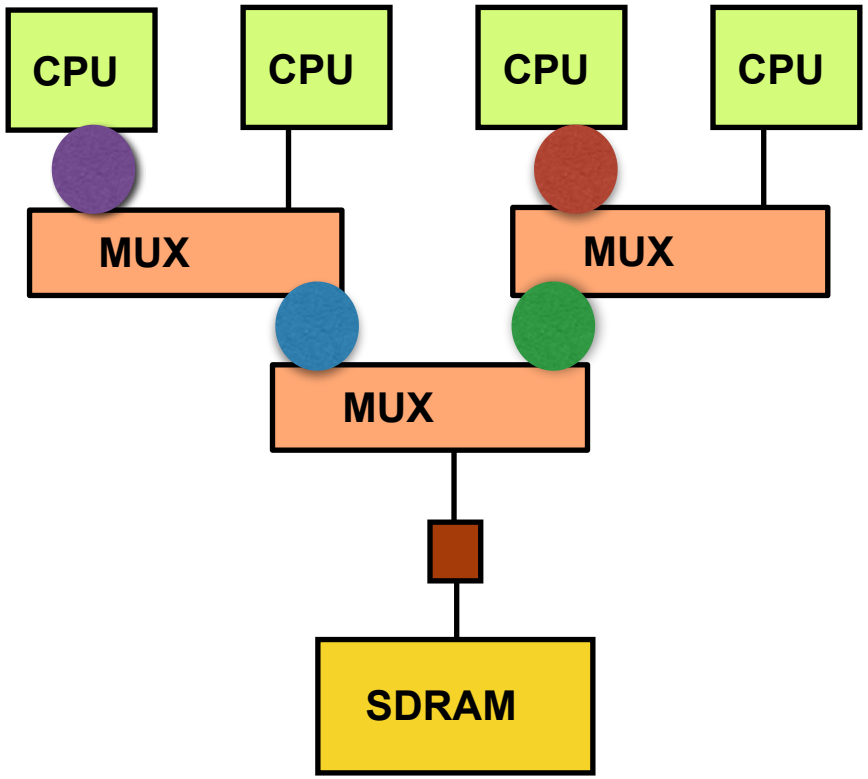
- CPU has connections to
  - inter-CPU mesh
  - memory tree
- Memory requests are multiplexed through the tree
  - End-to-end depends on arbitration policy at MUX
- Full-duplex
- Supports CPU cache / SPM operations
  - reduces local memory sizes

Memory Tree Arbitration



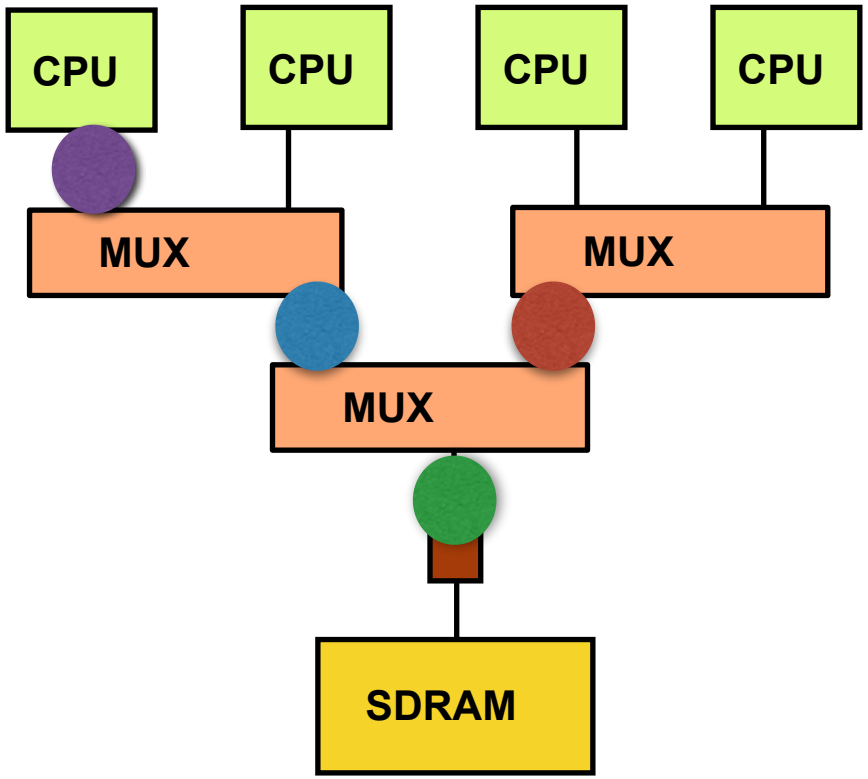
- ALTERNATE policy
  - Requests arriving one same clock cycle take turns
  - Fixed arbitration pattern

Memory Tree Arbitration



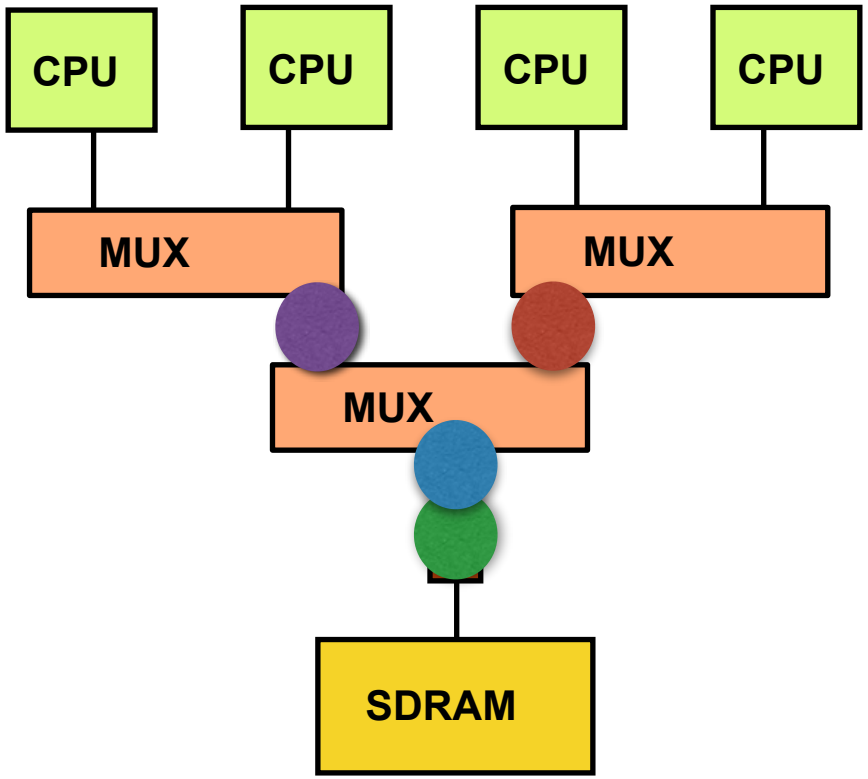
- ALTERNATE policy
  - Requests arriving one same clock cycle take turns
  - Fixed arbitration pattern

Memory Tree Arbitration



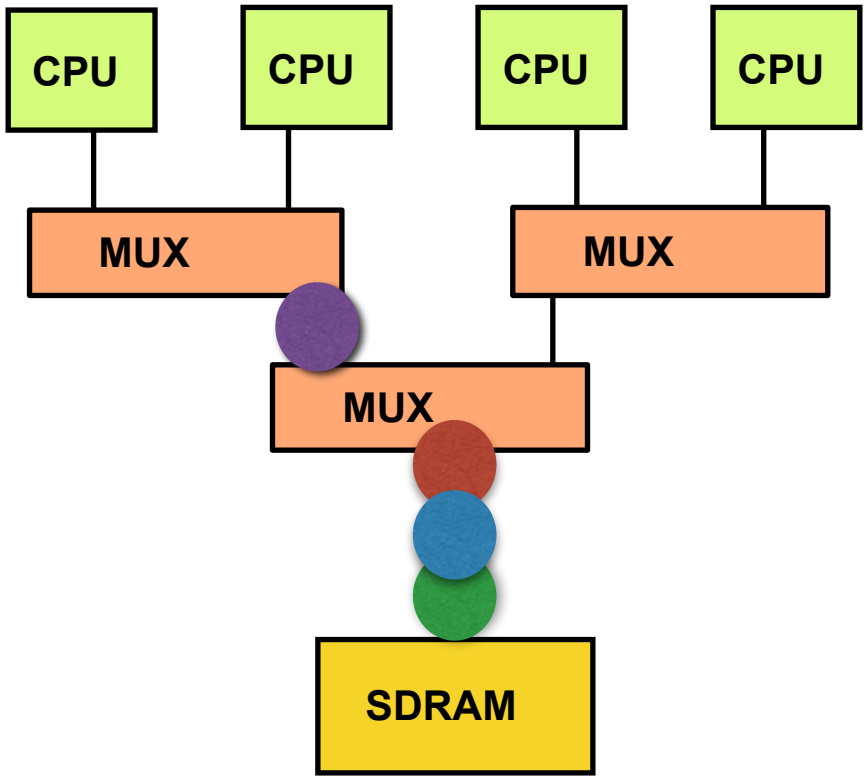
- ALTERNATE policy
  - Requests arriving one same clock cycle take turns
  - Fixed arbitration pattern

Memory Tree Arbitration



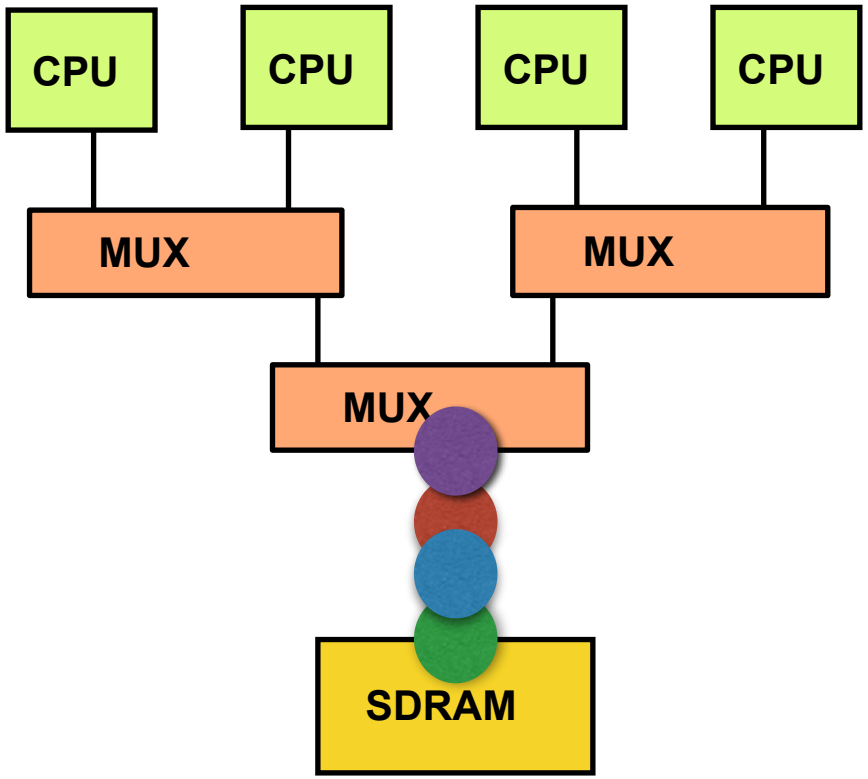
- ALTERNATE policy
  - Requests arriving one same clock cycle take turns
  - Fixed arbitration pattern

Memory Tree Arbitration



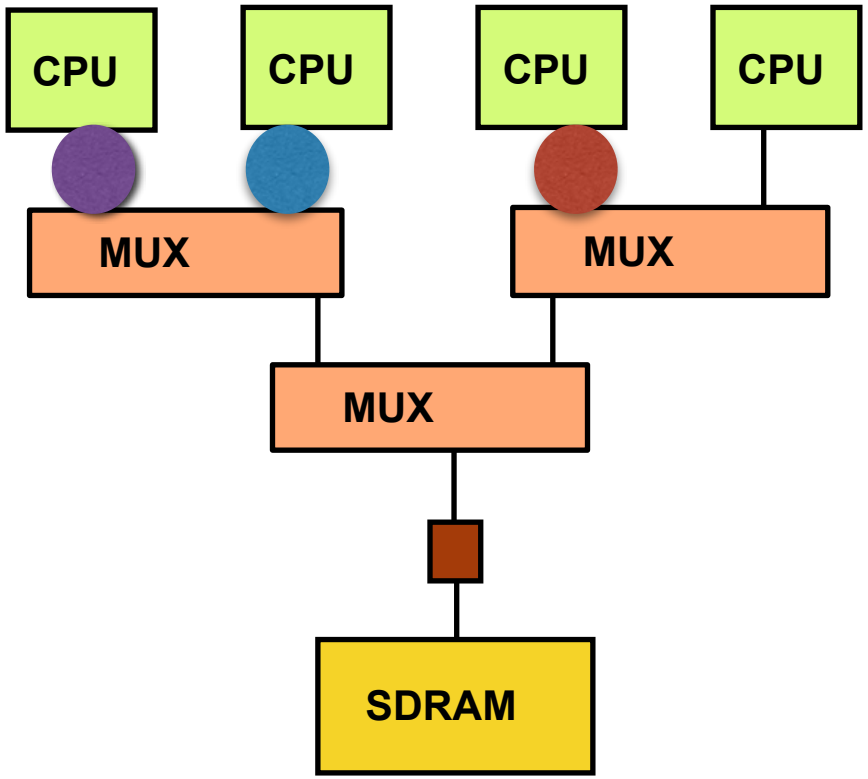
- ALTERNATE policy
  - Requests arriving one same clock cycle take turns
  - Fixed arbitration pattern

Memory Tree Arbitration



- ALTERNATE policy
  - Requests arriving one same clock cycle take turns
  - Fixed arbitration pattern
- Latched in memory controller during:
  - cycle 2
  - cycle 3
  - cycle 4
  - cycle 5 - exhibits WC

Memory Tree Arbitration

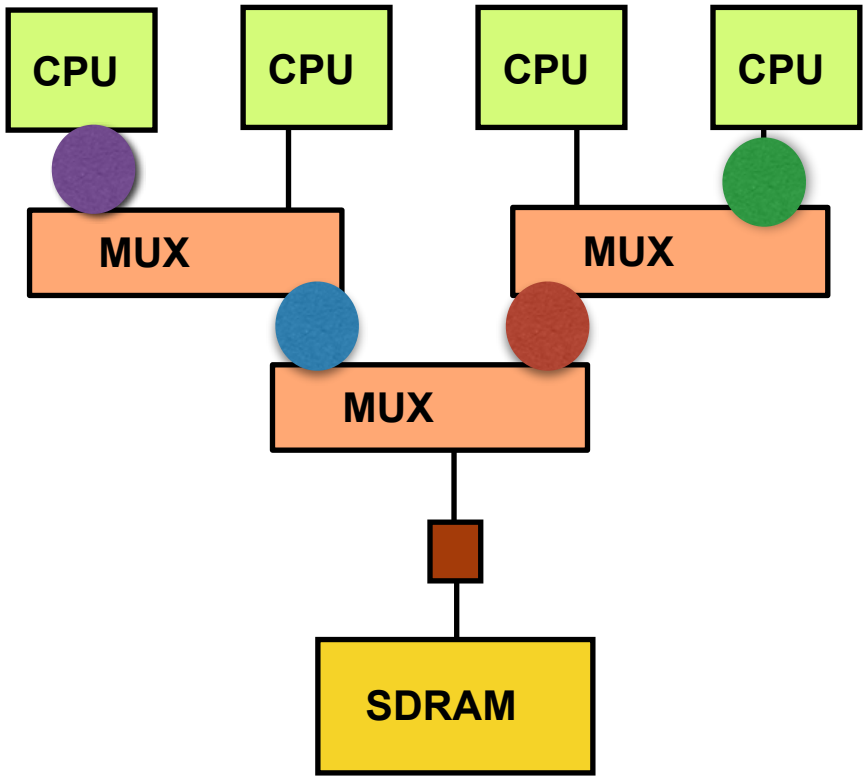


- PRIORITISE policy
  - Set arbitration (dynamically) according to system requirement
  - Eg. *criticality*

High
Medium
Medium
Low

  - *Secondary arbitration for equal criticality, eg. ALTERNATE*
  - *Can be set dynamically*
    - *eg. at context switch*
    - *eg. per memory request*

Memory Tree Arbitration

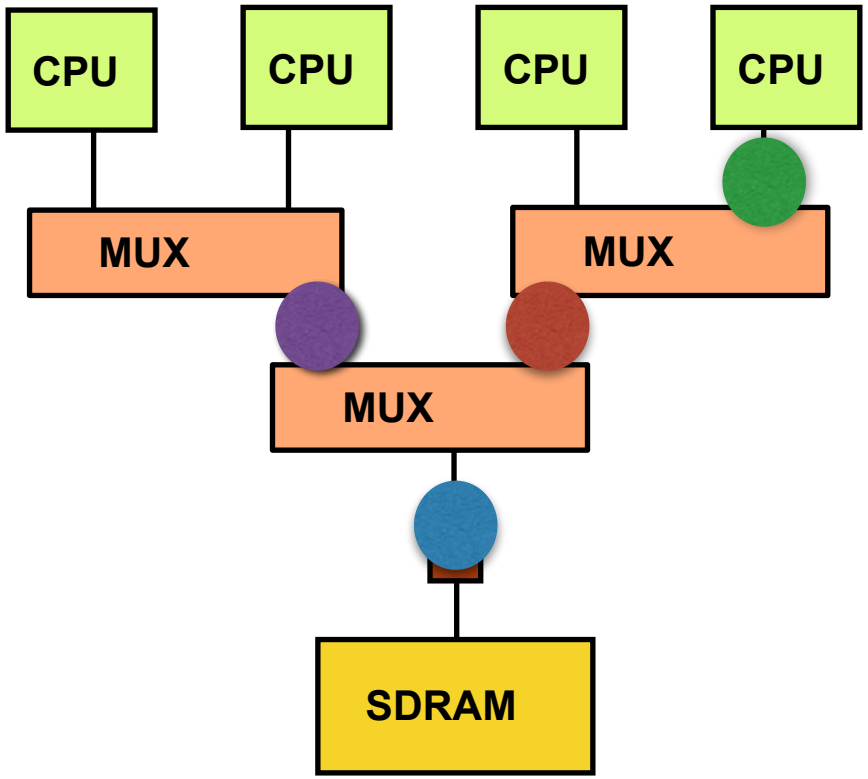


- PRIORITISE policy
  - Set arbitration (dynamically) according to system requirement
  - Eg. *criticality*

High
Medium
Medium
Low

- ***Danger of Priority Inversion***

Memory Tree Arbitration

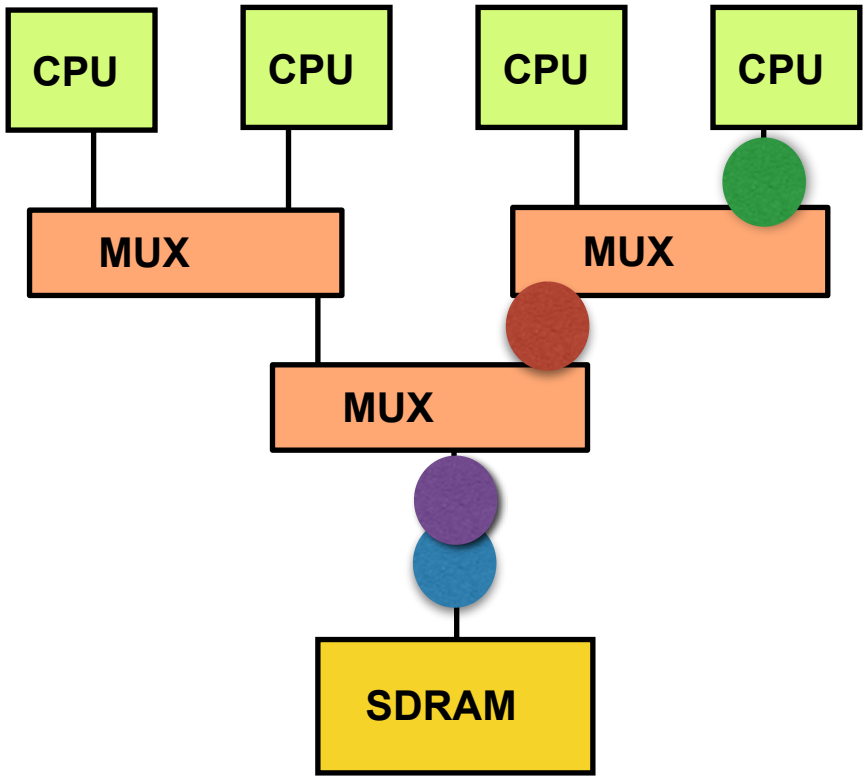


- PRIORITISE policy
  - Set arbitration (dynamically) according to system requirement
  - Eg. *criticality*

High
Medium
Medium
Low

- ***Danger of Priority Inversion***

Memory Tree Arbitration

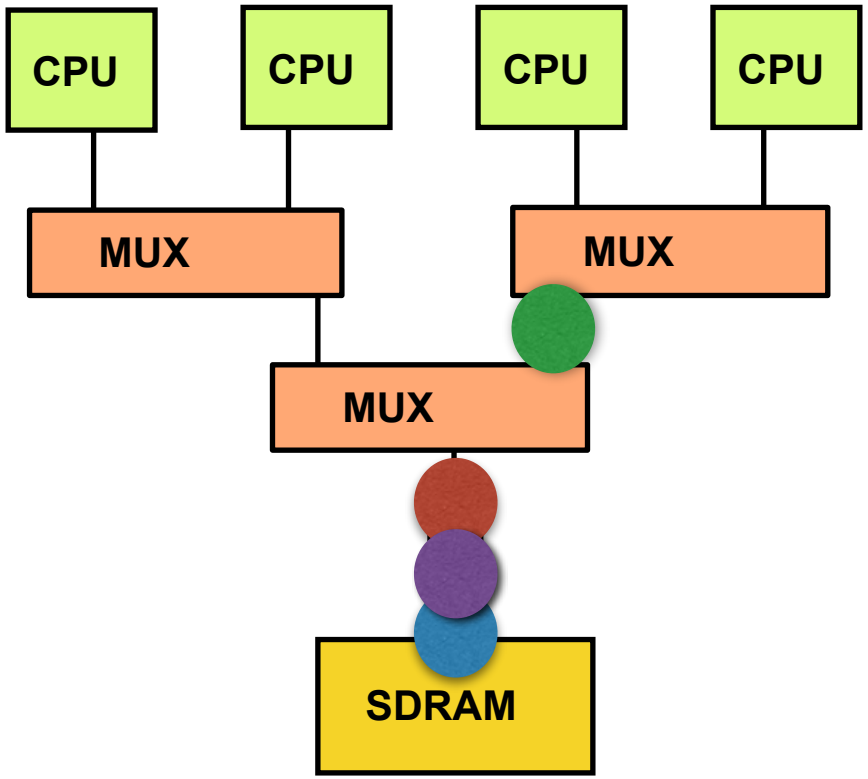


- PRIORITISE policy
  - Set arbitration (dynamically) according to system requirement
  - Eg. *criticality*

High
Medium
Medium
Low

- ***Danger of Priority Inversion***

Memory Tree Arbitration

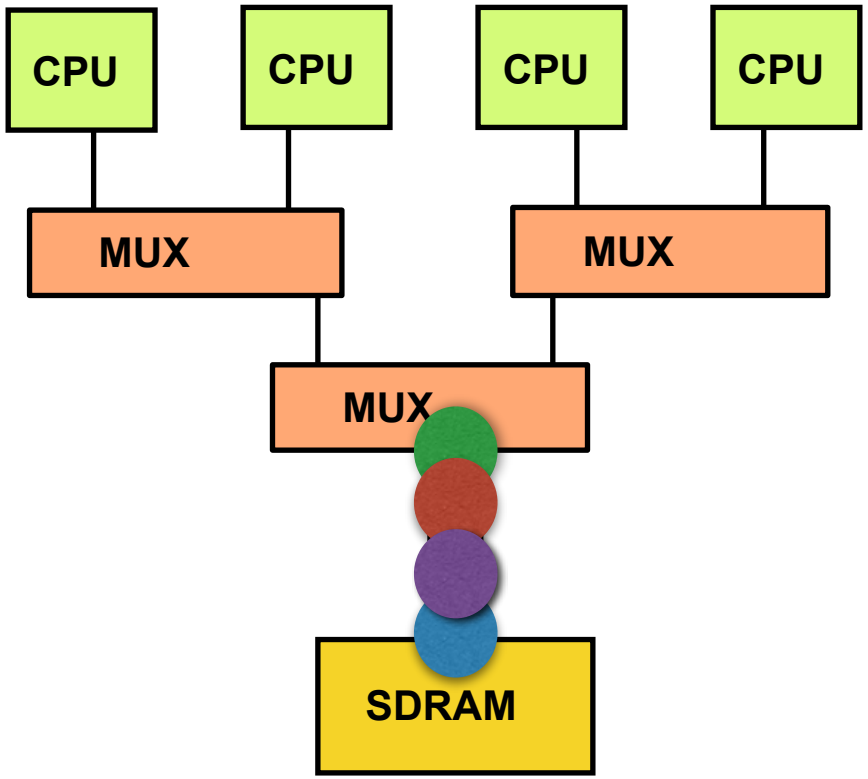


- PRIORITISE policy
  - Set arbitration (dynamically) according to system requirement
  - Eg. *criticality*

High
Medium
Medium
Low

- ***Danger of Priority Inversion***

Memory Tree Arbitration

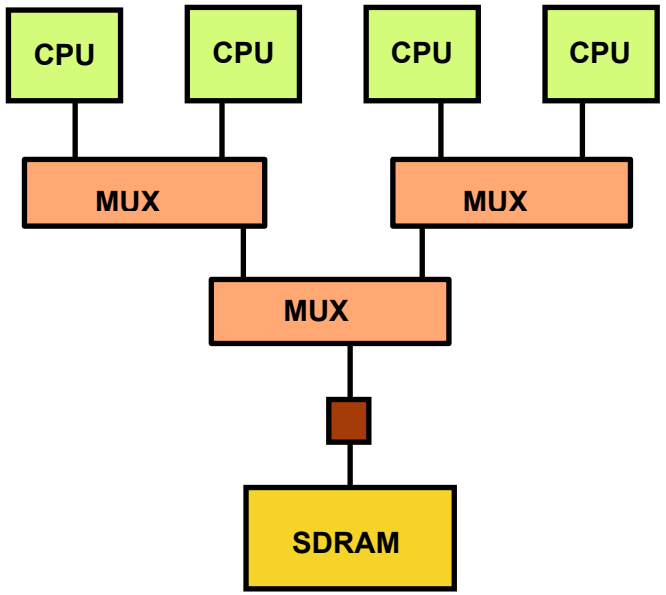


- PRIORITISE policy
  - Set arbitration (dynamically) according to system requirement
  - Eg. *criticality*

High
Medium
Medium
Low

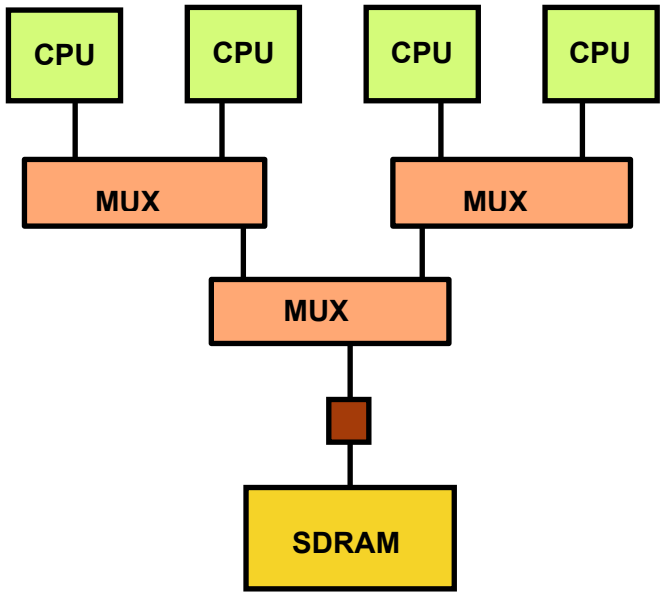
- ***Danger of Priority Inversion***
  - ***Investigating priority inheritance across a MUX***

Memory Tree Timing - Worst-Case



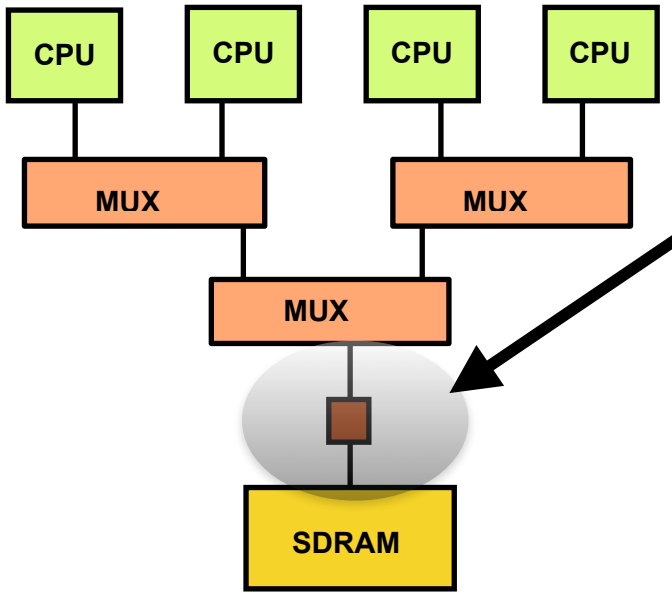
- Worst-case
  - 2 cycles to cross each MUX from CPU to SDRAM
  - Delay at external memory controller & physical SDRAM
  - 1 cycle to cross each MUX from SDRAM to CPU
- End-to-end depends on arbitration policy at MUX & SDRAM scheduling
  - defines “blocking time” at each MUX
  - ALTERNATE
    - max 1 cycle per MUX on request

Memory Tree Timing - Worst-Case



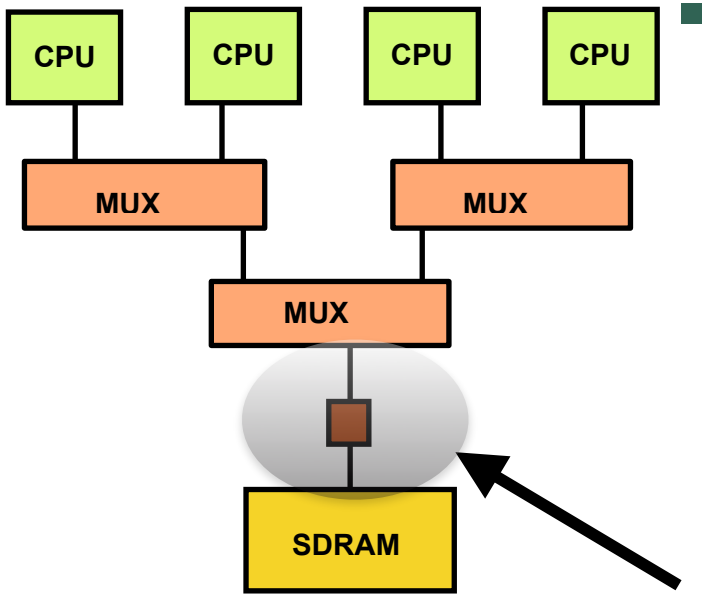
- CPU blocked on a cache miss
  - typical characteristic of CPUs
  - limits contending memory transactions
- SPM loads must complete before next can be issued
  - ie. blocking

Memory Tree Timing - Worst-Case



- Burst requests supported
  - One memory request from CPU converted into a number of sequential accesses at memory controller
  - Worst-case time in SDRAM for sequential requests much better than random
  - Currently 1/2/4/8 supported

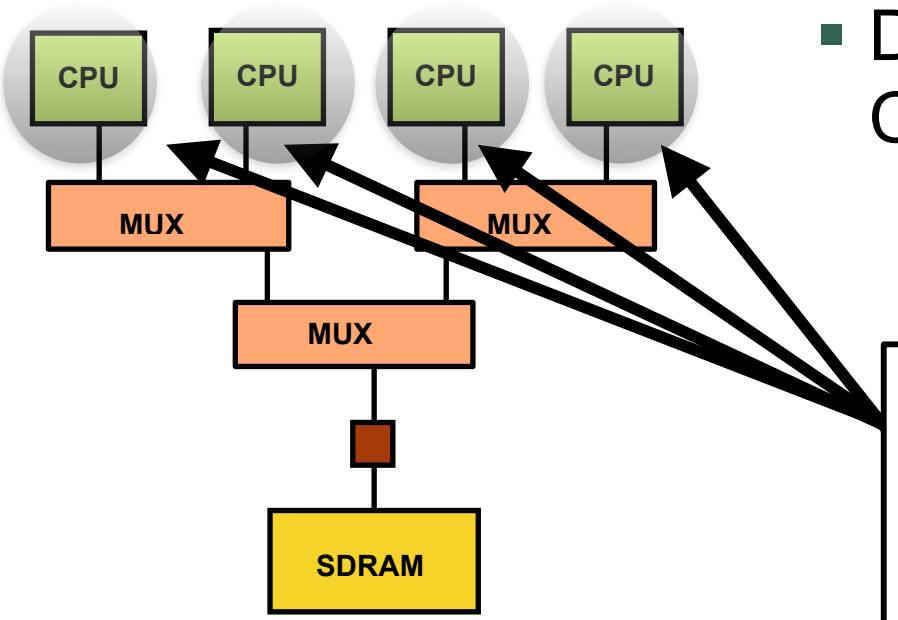
Memory Tree Timing - Worst-Case



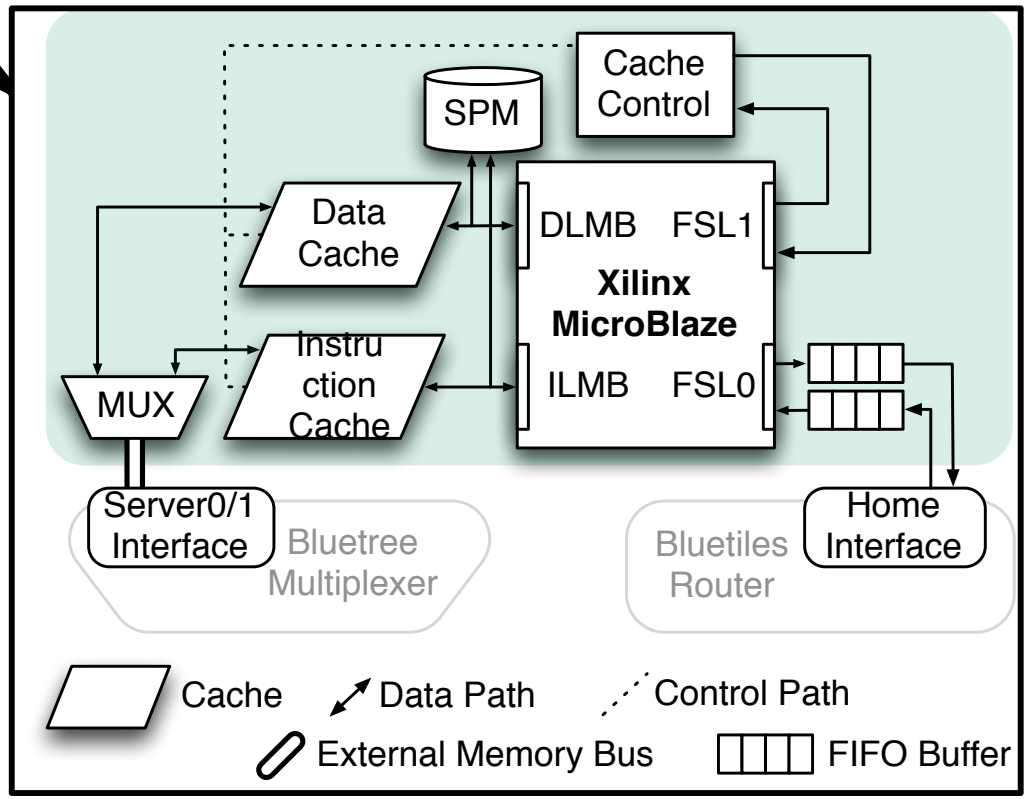
- Burst requests supported
  - One memory request from CPU converted into a number of sequential accesses at memory controller
  - Worst-case time in SDRAM for sequential requests much better than random
  - Currently 1/2/4/8 supported

- Bandwidth control within memory memory controller (TUE)
  - Effectively multiple channels / bandwidths can be set up
    - Currently max 4
  - Note still one channel between memory controller and SDRAM

Memory Tree Timing - Worst-Case



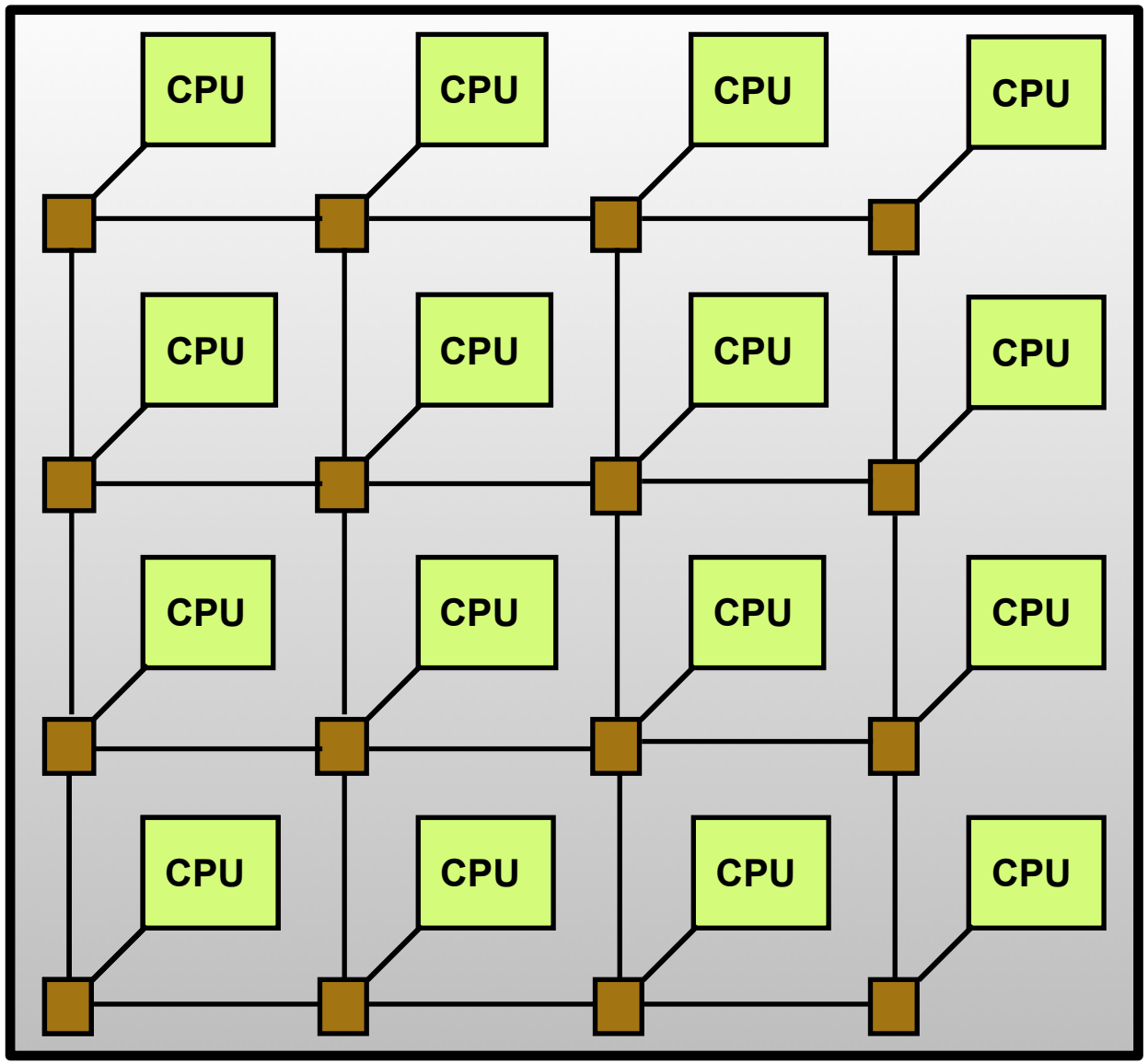
- Dual port "cache" provided between CPU and MUX
  - Allow CPU and memory tree to access local memory at same time



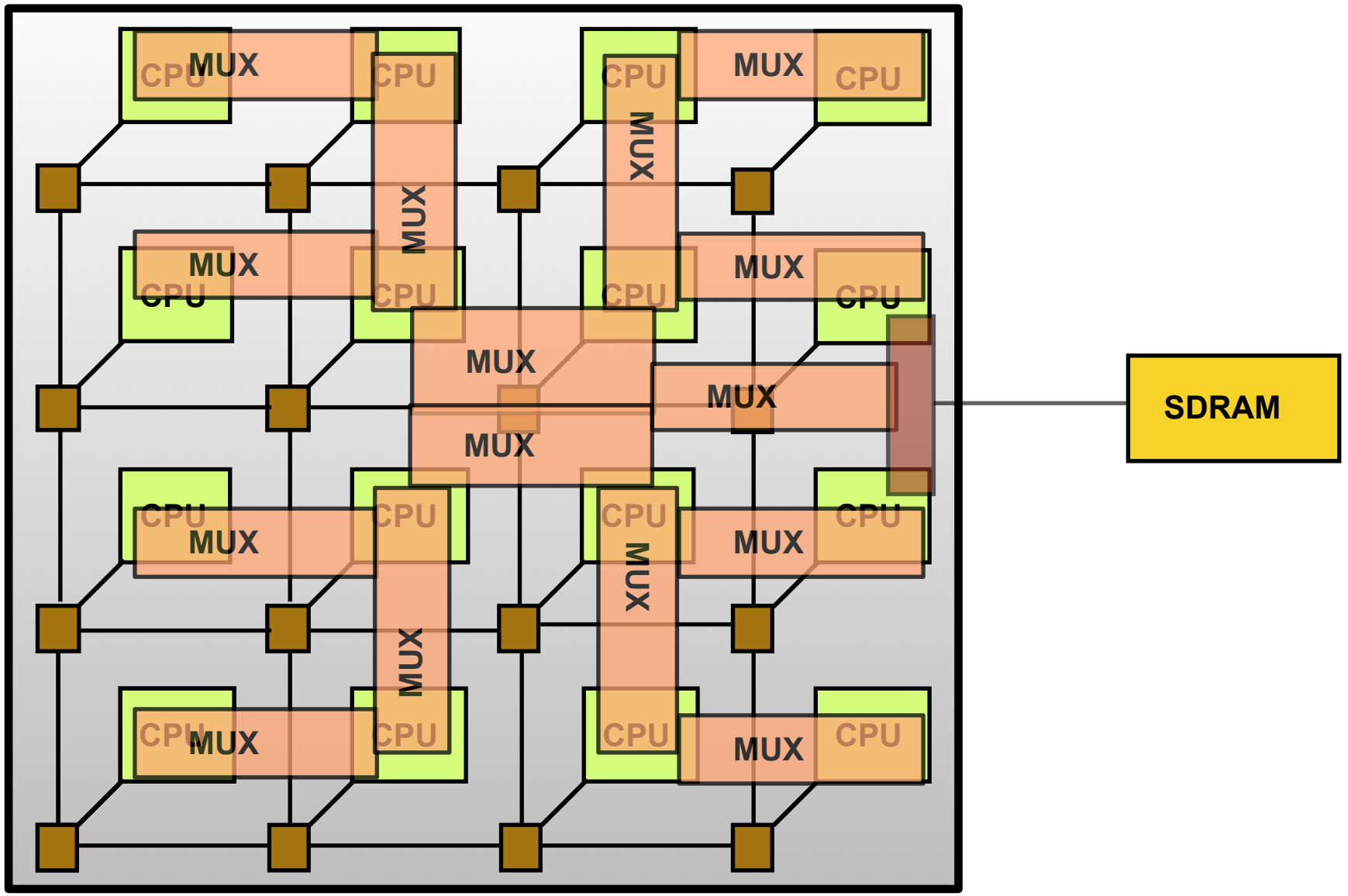
- *Being extended to limit bandwidth of request issued by CPU*
  - *Mechanism can also be used to limit effect of "babbling idiot"*

- 16 CPU
- Mesh, tree in middle

NoC & Overlayed Memory Mesh



NoC & Overlayed Memory Mesh



Summary

- Mixed criticality memory systems based on predictable memory systems (physical separation)
- Can support:
  - Per-latency analysis
  - Bandwidth analysis - (current work)
    more amenable to improved average case

Summary

- Mixed criticality memory systems based on predictable memory systems (physical separation)
- Can support:
  - Per-latency analysis
  - Bandwidth analysis - (current work)
    more amenable to improved average case
- Mixed Criticality Systems:
  - a safety-critical system with a high performance average case system trying to get out?