TECHNISCHE
UNIVERSITÄT
WIEN
Vienna | Austria

Resource Management for Mixed-Criticality Systems on Multi-Core Platforms with Focus on Communication

Embedded Tutorial

Robin Arbaud, Dávid Juhász, Axel Jantsch

DSD 2018, Prague, Czech Republic

August 31, 2018

Outline

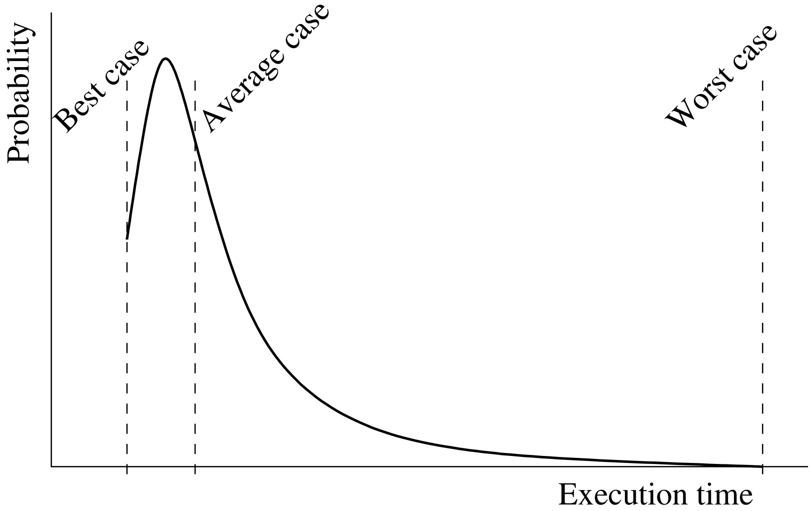
Resource Management for Mixed-Criticality Systems

Mixed Criticality Systems

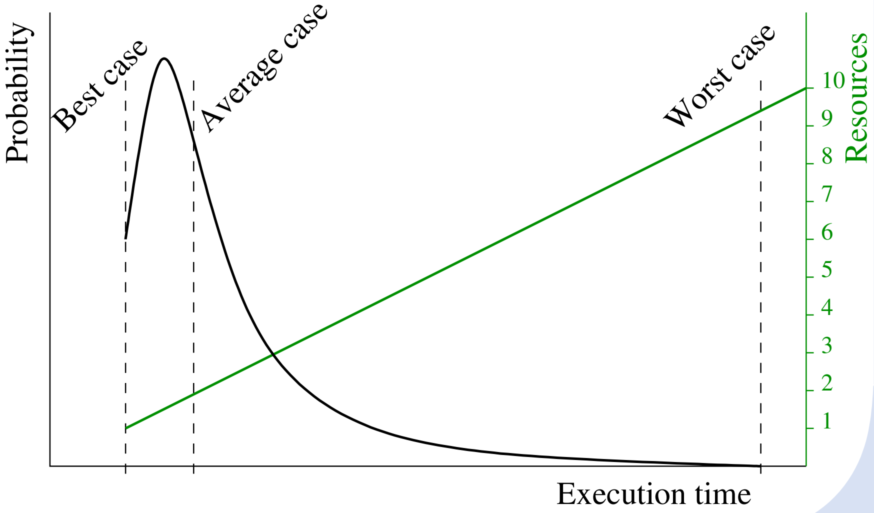
Types of tasks:

Best Effort	May take as long as needed;
Soft Real-Time	Have deadlines, but may miss some deadlines;
Hard Real-time	No deadline must be missed, ever.

Average Case and Worst Case



Average Case and Worst Case



Sharing and Isolation

- **Full Isolation** to accommodate the worst case:
    - Dedicated resources for a given task;
    - Addresses real-time and safety requirements;
    - Costly due to over-provisioning;
    - Complete isolation = no interference.

Sharing and Isolation

- **Full Isolation** to accommodate the worst case:
  - Dedicated resources for a given task;
  - Addresses real-time and safety requirements;
  - Costly due to over-provisioning;
  - Complete isolation = no interference.
- **Full sharing** to accommodate the average case:
  - Best average performance for given cost;
  - Lowest cost for target average performance;
  - Highest efficiency;
  - Unbounded worst case performance.

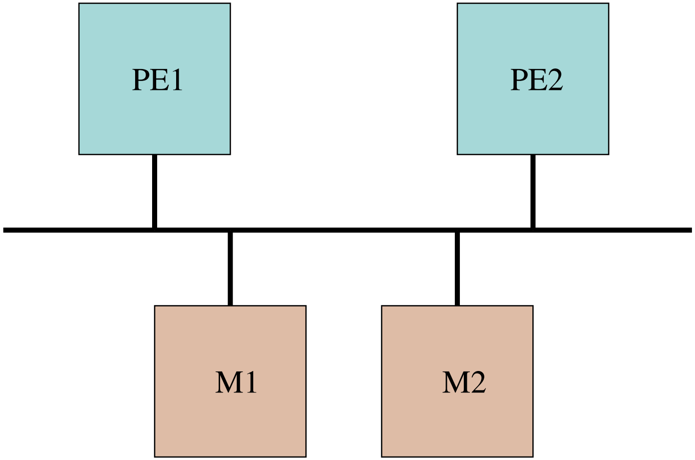
Sharing and Isolation

- **Full Isolation** to accommodate the worst case:
  - Dedicated resources for a given task;
  - Addresses real-time and safety requirements;
  - Costly due to over-provisioning;
  - Complete isolation = no interference.
- **Full sharing** to accommodate the average case:
  - Best average performance for given cost;
  - Lowest cost for target average performance;
  - Highest efficiency;
  - Unbounded worst case performance.
- Sharing with **bounded interference**:
  - Allowing and controlling interference can provide real-time and safety requirement;
  - Reduced costs;
  - Applicable for mix of critical and best effort tasks.

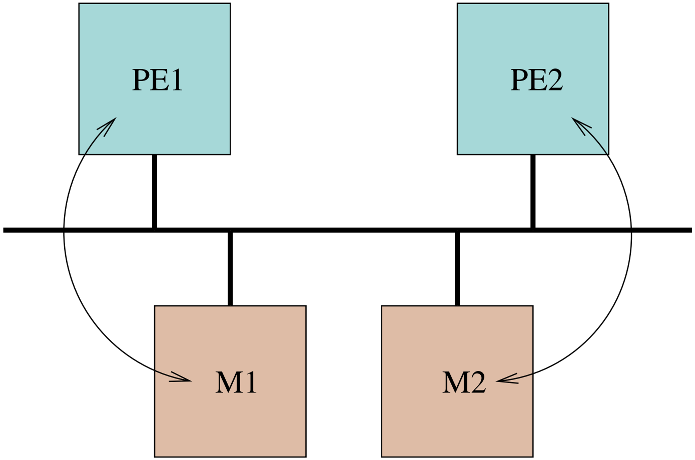
Interference Effects

- Direct interference and competition for the same resource;
- Indirect interference through shared resources:
    - Performance inversion,
    - Over-synchronization.

Performance Inversion

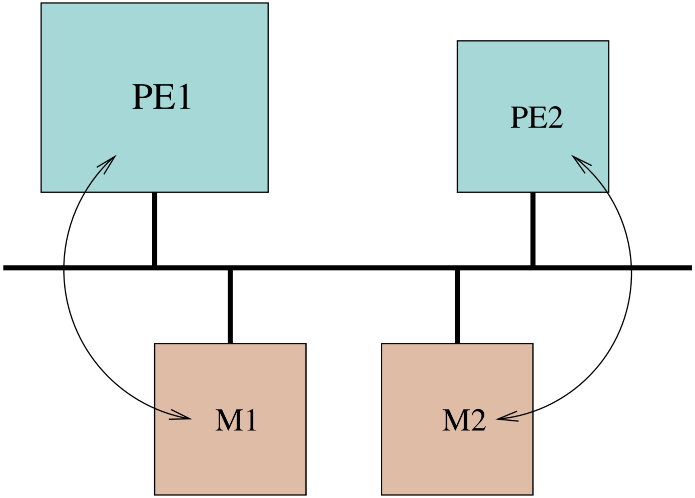


Performance Inversion



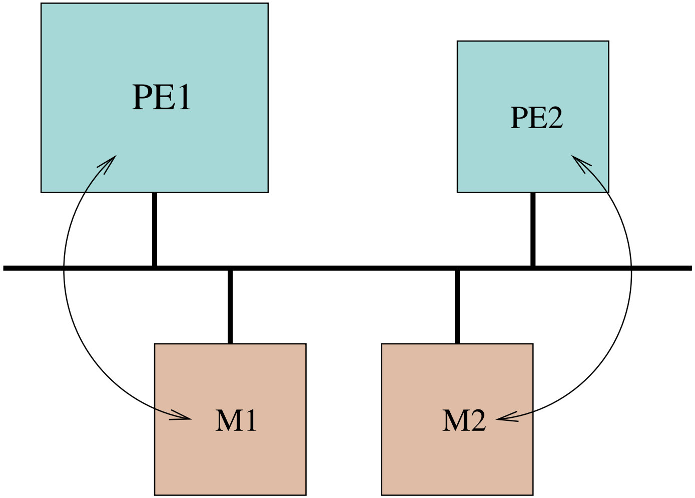
- PE1 communicates with M1 and PE2 communicates with M2;
- All tasks meet deadlines;

Performance Inversion



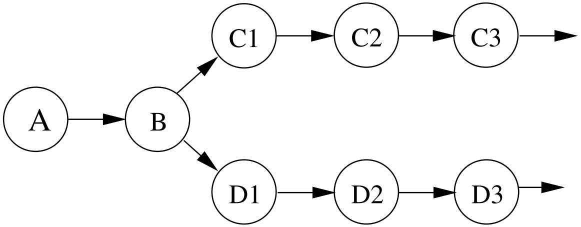
- Replacing PE1 with a faster PE;

Performance Inversion



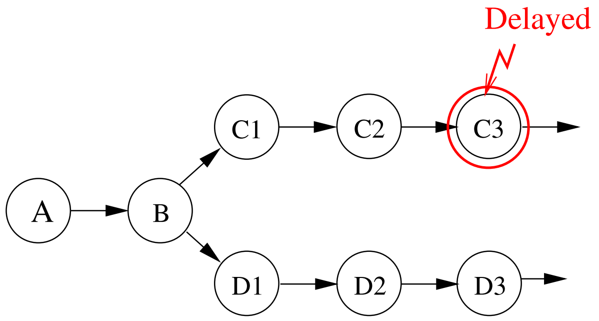
- Replacing PE1 with a faster PE;
- **May increase execution time of PE2 tasks.**

Over-Synchronization



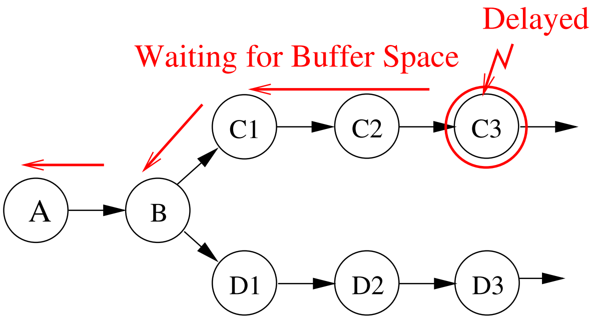
- Assumption: Bounded buffers between tasks;
- No control or data dependency between C and D branches;

Over-Synchronization



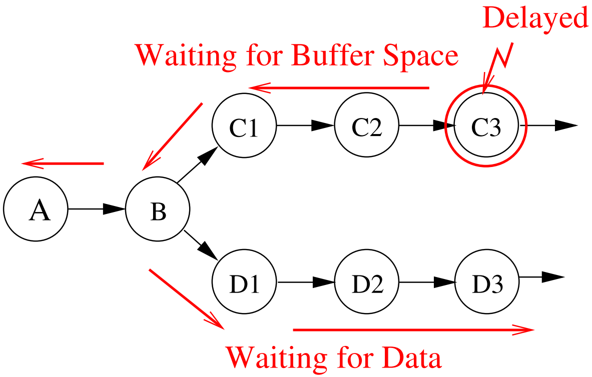
- Assumption: Bounded buffers between tasks;
- No control or data dependency between C and D branches;

Over-Synchronization



- Assumption: Bounded buffers between tasks;
- No control or data dependency between C and D branches;

Over-Synchronization

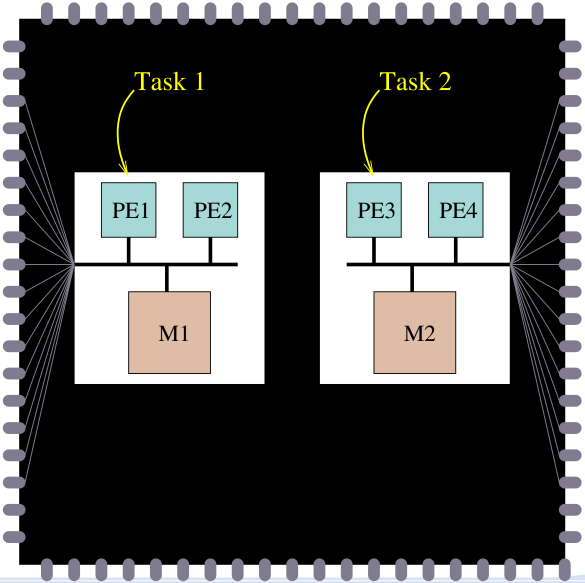


- Assumption: Bounded buffers between tasks;
- No control or data dependency between C and D branches;
- **If C is delayed or stuck, D suffers.**

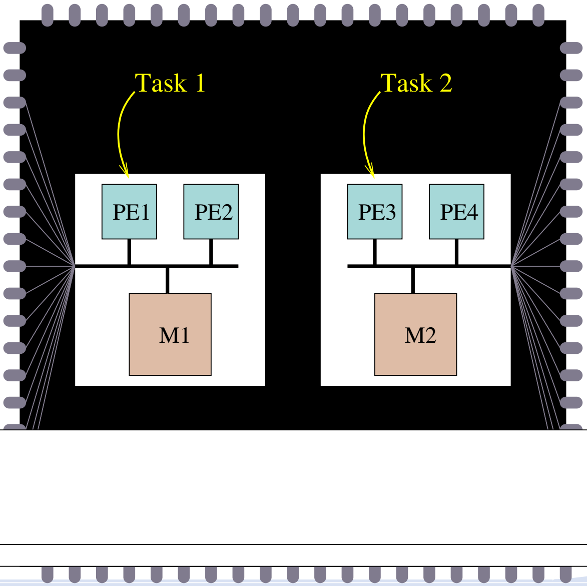
Spatial and Temporal Isolation

- Spatial isolation: No sharing at any time;
- Temporal isolation: Sharing at pre-defined time periods.

Spatial Isolation

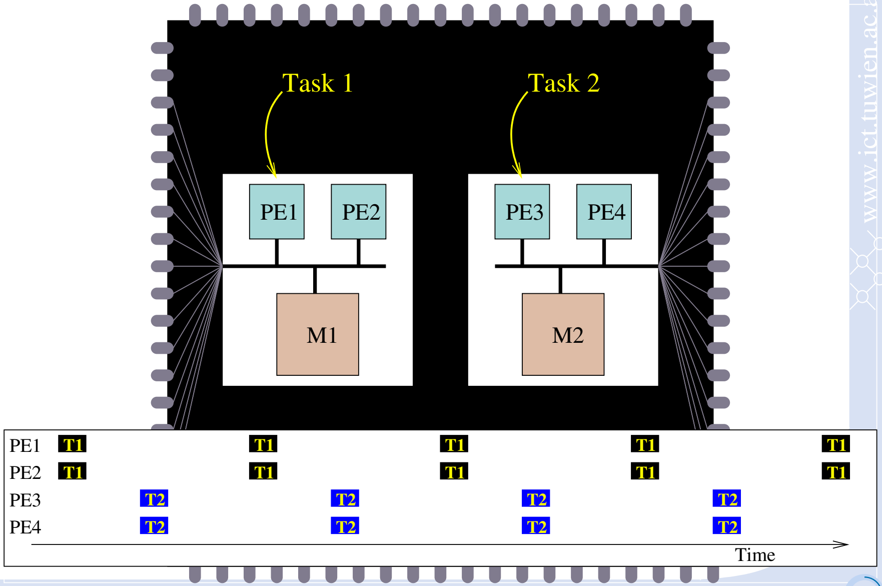


Spatial Isolation

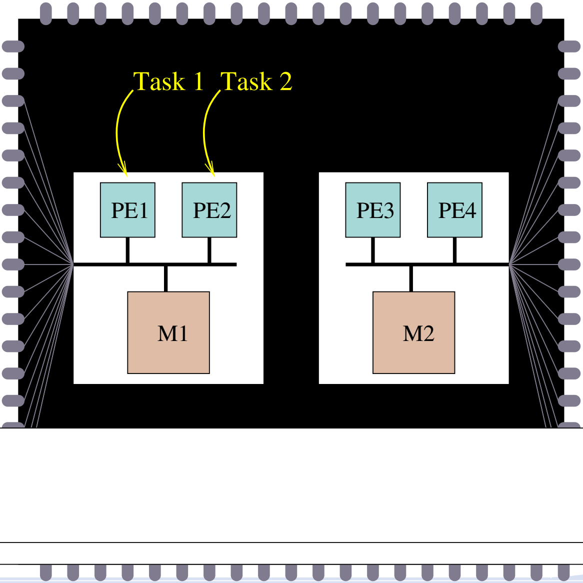


PE1
PE2
PE3
PE4
Time

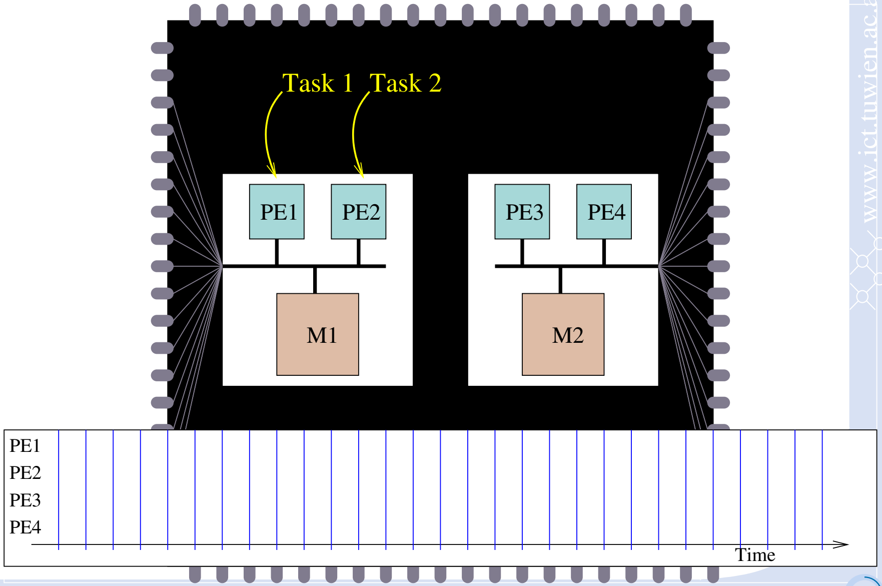
Spatial Isolation



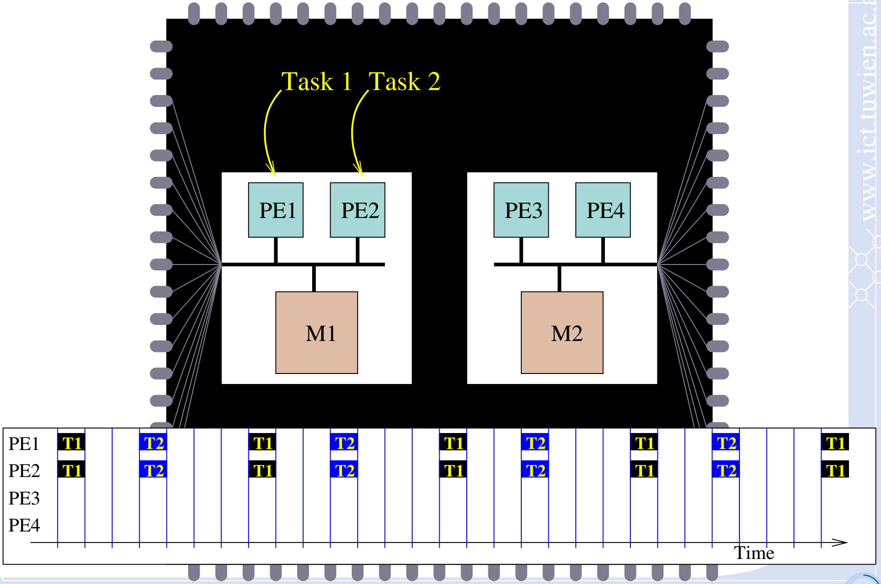
Temporal Isolation



Temporal Isolation



Temporal Isolation



Resource Management in Many-Core SoCs

Resource Management in Many-Core SoCs

- Many resources lead to a huge design space;

Resource Management in Many-Core SoCs

- Many resources lead to a huge design space;
- Types of Resources:

Resource Management in Many-Core SoCs

- Many resources lead to a huge design space;
- Types of Resources:
    - Computation: Processing elements,

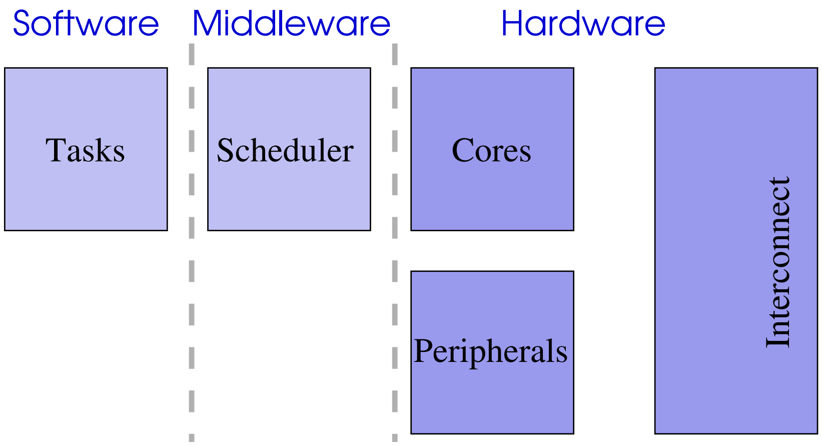
Resource Management in Many-Core SoCs

- Many resources lead to a huge design space;
- Types of Resources:
  - Computation: Processing elements,
  - Storage: Buffers, caches, off-chip memory,

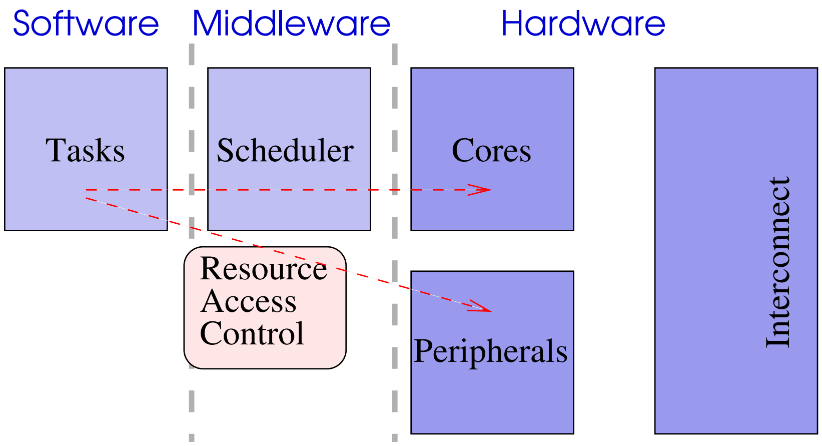
Resource Management in Many-Core SoCs

- Many resources lead to a huge design space;
- Types of Resources:
    - Computation: Processing elements,
    - Storage: Buffers, caches, off-chip memory,
    - **Communication: Buses, links, NoCs.**

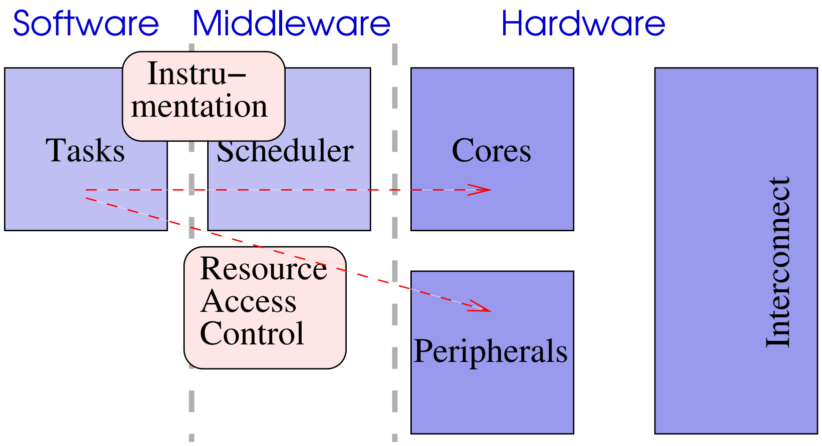
Overview


Software
Middleware
Hardware
Tasks
Scheduler
Cores
Peripherals
Interconnect

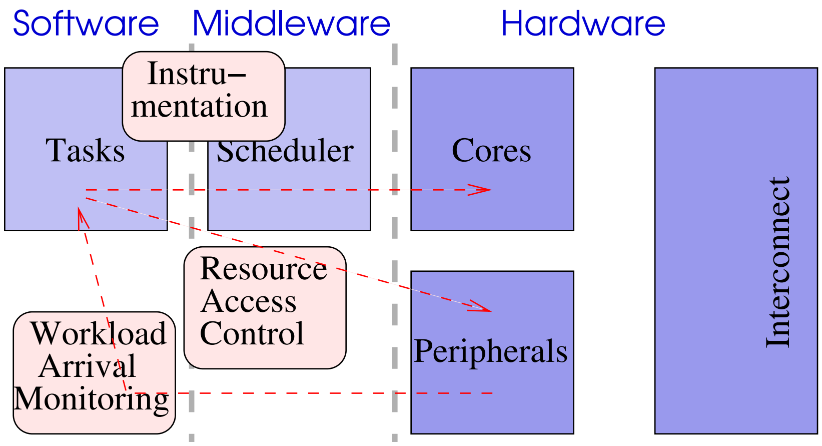
Overview


Software
Middleware
Hardware
Tasks
Scheduler
Cores
Interconnect
Resource Access Control
Peripherals

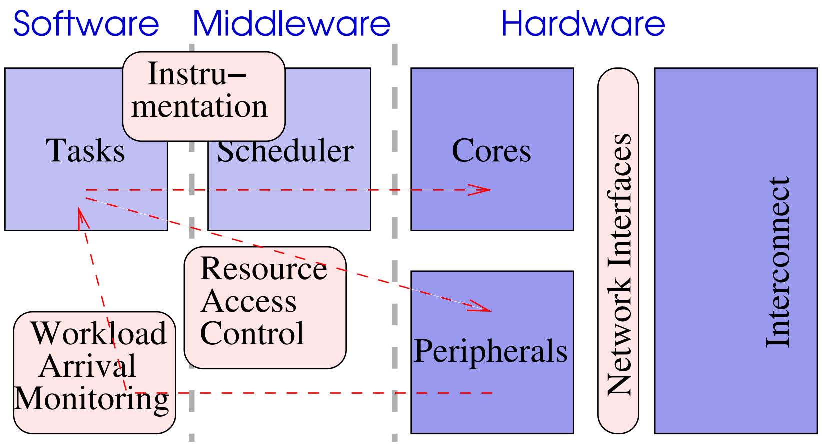
Overview


Software
Middleware
Hardware
Tasks
Instru–
mentation
Scheduler
Cores
Resource
Access
Control
Peripherals
Interconnect

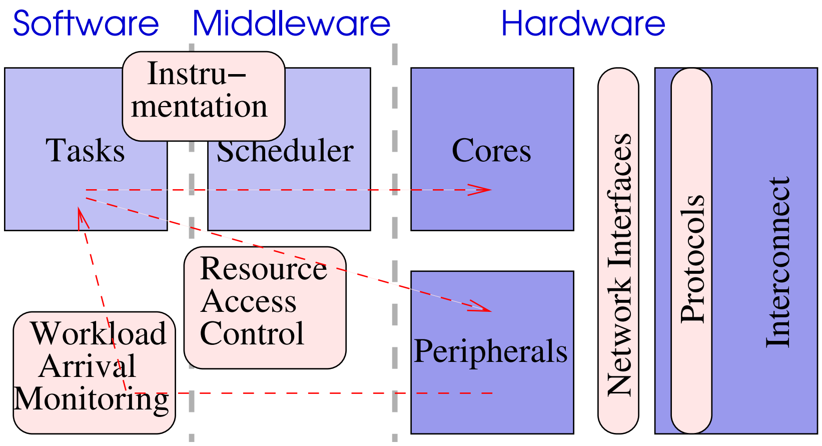
Overview


Software
Middleware
Hardware
Tasks
Instru–mentation
Scheduler
Cores
Interconnect
Resource Access Control
Peripherals
Workload Arrival Monitoring

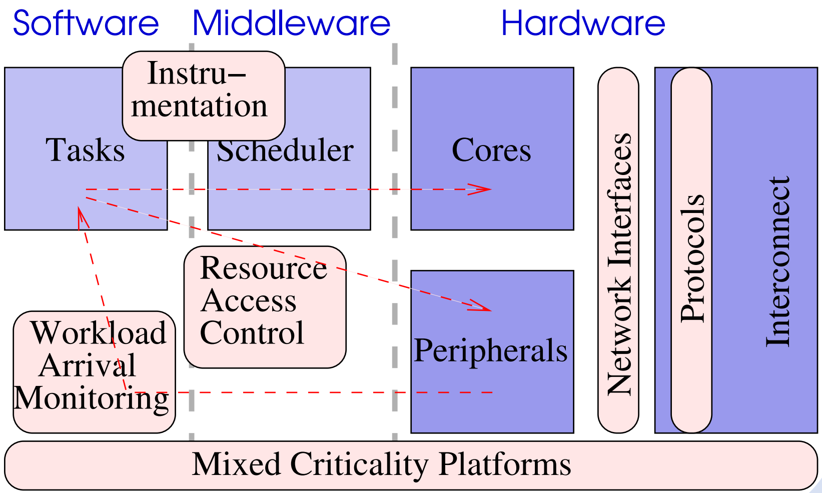
Overview


Software
Middleware
Hardware
Tasks
Instru–mentation
Scheduler
Cores
Network Interfaces
Interconnect
Resource Access Control
Workload Arrival Monitoring
Peripherals

Overview


Software
Middleware
Hardware
Instru–mentation
Tasks
Scheduler
Cores
Network Interfaces
Protocols
Interconnect
Resource Access Control
Workload Arrival Monitoring
Peripherals

Overview



Software
Middleware
Hardware
Tasks
Instru–
mentation
Scheduler
Cores
Network Interfaces
Protocols
Interconnect
Resource
Access
Control
Peripherals
Workload
Arrival
Monitoring
Mixed Criticality Platforms

Resource Management for Mixed-Criticality Systems

Real-Time Systems

Real-Time Systems

Real-Time Constraints from Event to System Response

Real-Time Systems

Real-Time Constraints from Event to System Response

- Deadlines based on real-time requirements

Real-Time Systems

Real-Time Constraints from Event to System Response

- Deadlines based on real-time requirements
- No deadline misses allowed ever

Real-Time Systems

Real-Time Constraints from Event to System Response

- Deadlines based on real-time requirements
- No deadline misses allowed ever
- Prepare for the worst-case

Real-Time Systems – Scheduling Example

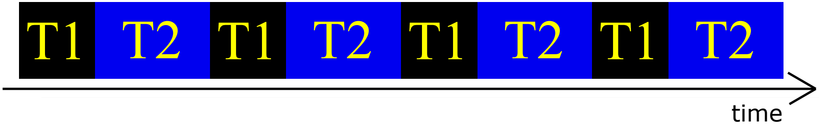
Task Set

Task	Period	Deadline	WCET
$T1$	10	5	4
$T2$	10	10	6

Real-Time Systems – Scheduling Example

Task Set

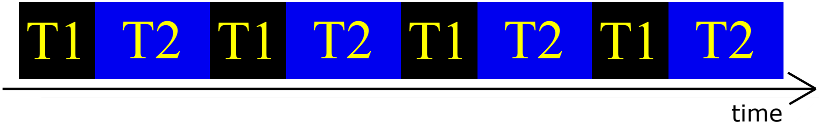
Task	**Period**	**Deadline**	**WCET**
$T1$	10	5	4
$T2$	10	10	6



Real-Time Systems – Scheduling Example

Task Set

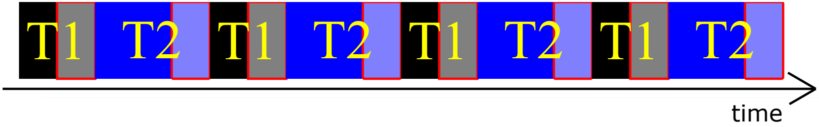
Task	Period	Deadline	WCET	ACET
$T1$	10	5	4	2
$T2$	10	10	6	4



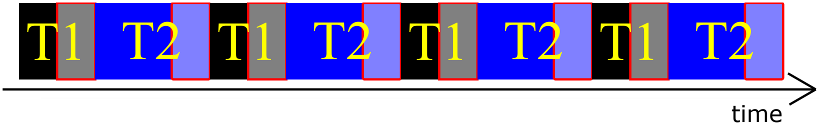
Real-Time Systems – Scheduling Example

Task Set

Task	Period	Deadline	WCET	ACET
$T1$	10	5	4	2
$T2$	10	10	6	4

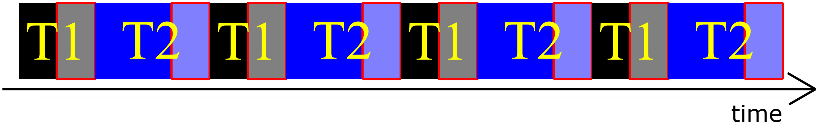


Real-Time Systems – Pros and Cons



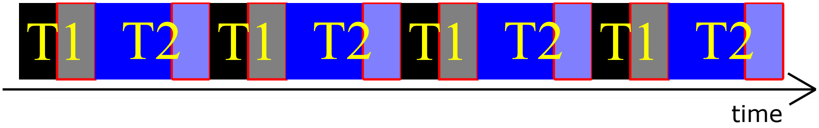
Real-Time Systems – Pros and Cons

- Static guarantees even for the worst-case



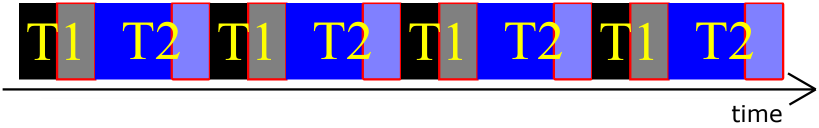
Real-Time Systems – Pros and Cons

- Static guarantees even for the worst-case
- Overprovisioning for the average-case



Real-Time Systems – Pros and Cons

- Static guarantees even for the worst-case
- Overprovisioning for the average-case



Economic need for trading off guarantees for utilization

Resource Management for Mixed-Criticality Systems

Mixed Criticality Levels

Mixed Criticality Levels

Worst-Case Execution Time Estimates

Mixed Criticality Levels

Worst-Case Execution Time Estimates

- Various components are validated against various assumptions

Mixed Criticality Levels

Worst-Case Execution Time Estimates

- Various components are validated against various assumptions
- The stricter assumptions, the longer estimated WCET

Mixed Criticality Levels

Worst-Case Execution Time Estimates

- Various components are validated against various assumptions
- The stricter assumptions, the longer estimated WCET

Criticality Levels of Tasks

Mixed Criticality Levels

Worst-Case Execution Time Estimates

- Various components are validated against various assumptions
- The stricter assumptions, the longer estimated WCET

Criticality Levels of Tasks

- Strictness of assumptions

Mixed Criticality Levels

Worst-Case Execution Time Estimates

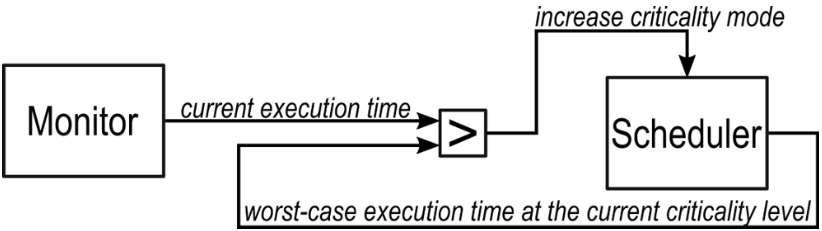
- Various components are validated against various assumptions
- The stricter assumptions, the longer estimated WCET

Criticality Levels of Tasks

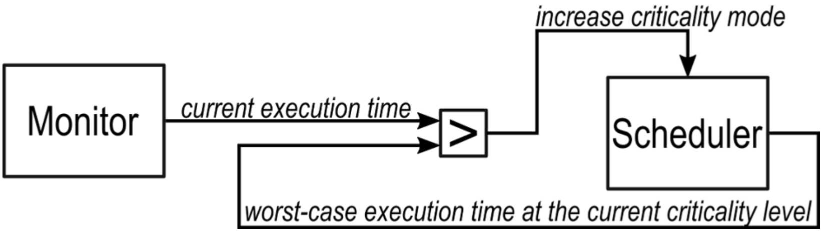
- Strictness of assumptions
- Ordered set of levels

Mixed-Criticality Scheduling with Criticality Mode Switches

Mixed-Criticality Scheduling with Criticality Mode Switches

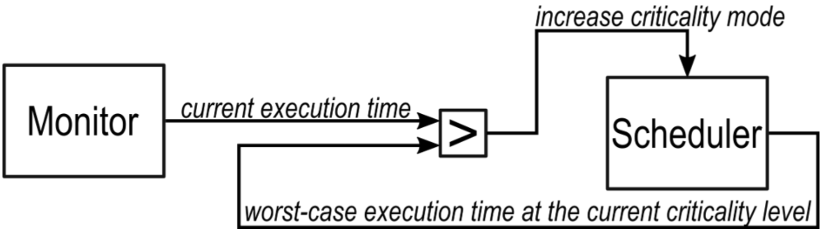

increase criticality mode
Monitor
current execution time
>
Scheduler
worst-case execution time at the current criticality level

Mixed-Criticality Scheduling with Criticality Mode Switches


increase criticality mode
Monitor
current execution time
>
Scheduler
worst-case execution time at the current criticality level

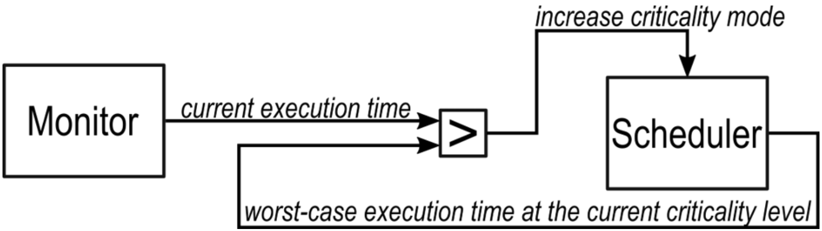
- Start system in the lowest criticality mode

Mixed-Criticality Scheduling with Criticality Mode Switches


increase criticality mode
Monitor
current execution time
>
Scheduler
worst-case execution time at the current criticality level

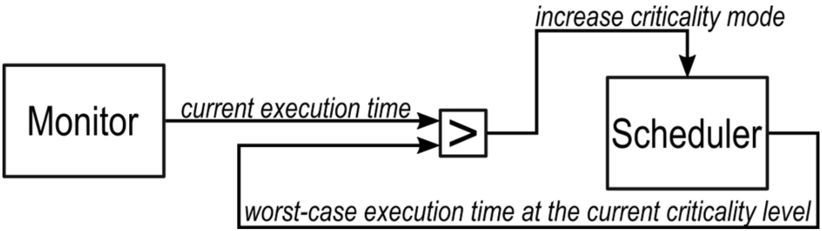
- Start system in the lowest criticality mode
- Schedule tasks with criticality not below the current criticality mode

Mixed-Criticality Scheduling with Criticality Mode Switches


increase criticality mode
Monitor
current execution time
>
Scheduler
worst-case execution time at the current criticality level

- Start system in the lowest criticality mode
- Schedule tasks with criticality not below the current criticality mode
- Monitor whether execution violates current assumptions

Mixed-Criticality Scheduling with Criticality Mode Switches


Monitor
current execution time
>
increase criticality mode
Scheduler
worst-case execution time at the current criticality level

- Start system in the lowest criticality mode
- Schedule tasks with criticality not below the current criticality mode
- Monitor whether execution violates current assumptions
- Switch to higher criticality mode for stricter assumptions

Mixed-Criticality Scheduling with Mode Switches

Task Set *HI*

Task	Period	Deadline	WCET
$T1$	10	5	4
$T2$	10	10	6

Mixed-Criticality Scheduling with Mode Switches

Task Set *HI*

Task	Period	Deadline	WCET
$T1$	10	5	4
$T2$	10	10	6

Task Set *LO*

Task	Period	Deadline	WCET
$T1$	10	5	2
$T2$	10	10	4
$T3$	10	9	3

Mixed-Criticality Scheduling with Mode Switches

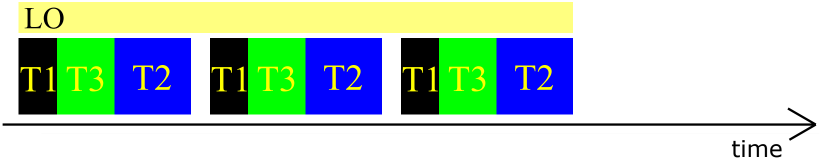
Mixed-Criticality Task Set

Task	CL	Period	Deadline	WCET
$T1$	HI	10	5	$[2, 4]$
$T2$	HI	10	10	$[4, 6]$
$T3$	LO	10	9	$[3]$

Mixed-Criticality Scheduling with Mode Switches

Mixed-Criticality Task Set

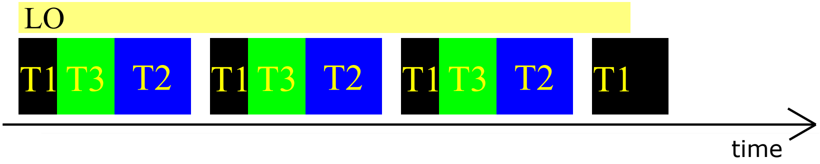
Task	**CL**	**Period**	**Deadline**	**WCET**
$T1$	HI	10	5	$[2, 4]$
$T2$	HI	10	10	$[4, 6]$
$T3$	LO	10	9	$[3]$



Mixed-Criticality Scheduling with Mode Switches

Mixed-Criticality Task Set

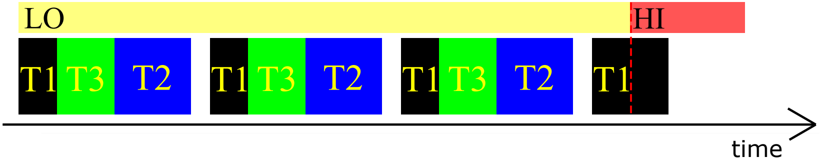
Task	CL	Period	Deadline	WCET
$T1$	HI	10	5	$[2,4]$
$T2$	HI	10	10	$[4,6]$
$T3$	LO	10	9	$[3]$



Mixed-Criticality Scheduling with Mode Switches

Mixed-Criticality Task Set

Task	**CL**	**Period**	**Deadline**	**WCET**
$T1$	HI	10	5	$[2, 4]$
$T2$	HI	10	10	$[4, 6]$
$T3$	LO	10	9	$[3]$

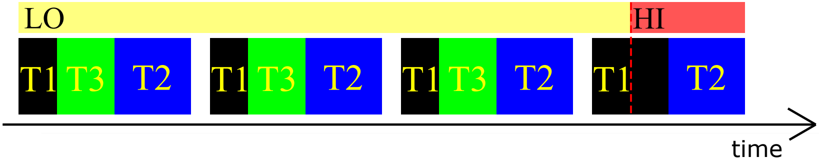


Ignoring *LO* criticality tasks in *HI* mode

Mixed-Criticality Scheduling with Mode Switches

Mixed-Criticality Task Set

Task	**CL**	**Period**	**Deadline**	**WCET**
$T1$	HI	10	5	$[2, 4]$
$T2$	HI	10	10	$[4, 6]$
$T3$	LO	10	9	$[3]$



Ignoring *LO* criticality tasks in *HI* mode

Limitations of Mode Switching Scheduling

Limitations of Mode Switching Scheduling

Static Verification

Runtime Robustness

Limitations of Mode Switching Scheduling

Static Verification

- Timing guarantees for each mode

Runtime Robustness

Limitations of Mode Switching Scheduling

Static Verification

- Timing guarantees for each mode
- **No link to requirements of safety standards**

Runtime Robustness

Limitations of Mode Switching Scheduling

Static Verification

- Timing guarantees for each mode
- **No link to requirements of safety standards**

Runtime Robustness

- Switching mode to restrict assumptions as necessary

Limitations of Mode Switching Scheduling

Static Verification

- Timing guarantees for each mode
- **No link to requirements of safety standards**

Runtime Robustness

- Switching mode to restrict assumptions as necessary
- **Ignoring low-criticality tasks**

Limitations of Mode Switching Scheduling

Static Verification

- Timing guarantees for each mode
- **No link to requirements of safety standards**

Runtime Robustness

- Switching mode to restrict assumptions as necessary
- **Ignoring low-criticality tasks**
- **Returning to low-criticality mode**

Resource Management for Mixed-Criticality Systems

Memory Accesses and Execution Time

Memory Accesses and Execution Time

Cache Misses

Memory Accesses and Execution Time

Cache Misses

- Memory needs to be accessed

Memory Accesses and Execution Time

Cache Misses

- Memory needs to be accessed
- Critical tasks may be penalized by interference

Memory Accesses and Execution Time

Cache Misses

- Memory needs to be accessed
- Critical tasks may be penalized by interference

Limited Interference for Critical Tasks

Memory Accesses and Execution Time

Cache Misses

- Memory needs to be accessed
- Critical tasks may be penalized by interference

Limited Interference for Critical Tasks

- Reduce WCET of critical tasks by limiting interference

Memory Accesses and Execution Time

Cache Misses

- Memory needs to be accessed
- Critical tasks may be penalized by interference

Limited Interference for Critical Tasks

- Reduce WCET of critical tasks by limiting interference
- Budget of cache misses for non-critical tasks (cores)

Memory Accesses and Execution Time

Cache Misses

- Memory needs to be accessed
- Critical tasks may be penalized by interference

Limited Interference for Critical Tasks

- Reduce WCET of critical tasks by limiting interference
- Budget of cache misses for non-critical tasks (cores)
- Suspend execution of tasks with depleted budget

Memory Accesses and Execution Time

Cache Misses

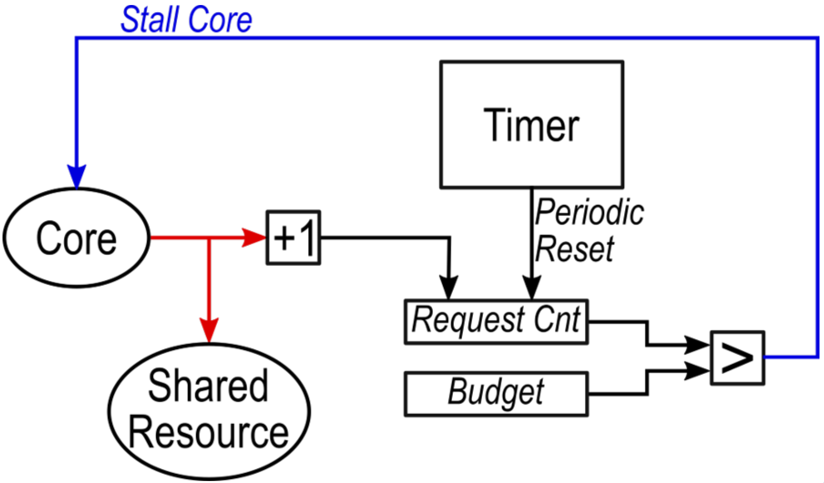
- Memory needs to be accessed
- Critical tasks may be penalized by interference

Limited Interference for Critical Tasks

- Reduce WCET of critical tasks by limiting interference
- Budget of cache misses for non-critical tasks (cores)
- Suspend execution of tasks with depleted budget

Can be generalized to any shared resource

Memory Access Budgeting



Memory Access Budgeting – Pros and Cons

Pros

Cons

Memory Access Budgeting – Pros and Cons

Pros

- Low overhead on COTS

Cons

Memory Access Budgeting – Pros and Cons

Pros

- Low overhead on COTS
  - Performance counters
  - Interprocessor interrupts

Cons

Memory Access Budgeting – Pros and Cons

Pros

- Low overhead on COTS
    - Performance counters
    - Interprocessor interrupts

Cons

- Limited to 2 levels, critical and non-critical

Memory Access Budgeting – Pros and Cons

Pros

- Low overhead on COTS
  - Performance counters
  - Interprocessor interrupts

Cons

- Limited to 2 levels, critical and non-critical
- No general algorithm for deriving budgets

Memory Access Budgeting – Pros and Cons

Pros

- Low overhead on COTS
  - Performance counters
  - Interprocessor interrupts

Cons

- Limited to 2 levels, critical and non-critical
- No general algorithm for deriving budgets
- Not scaling with the number of non-critical cores

Resource Management for Mixed-Criticality Systems

Monitoring Execution Time and Remaining WCET

Monitoring Execution Time and Remaining WCET

WCET and Observation Points for Critical Tasks

Monitoring Execution Time and Remaining WCET

WCET and Observation Points for Critical Tasks

- Remaining WCET for each observation point

Monitoring Execution Time and Remaining WCET

WCET and Observation Points for Critical Tasks

- Remaining WCET for each observation point
- Worst-case interference between observation points

Monitoring Execution Time and Remaining WCET

WCET and Observation Points for Critical Tasks

- Remaining WCET for each observation point
- Worst-case interference between observation points

Safety Condition at each Observation Point

The critical task will not miss its deadline even if worst-case happens until the next observation point

Execution Time Monitoring

Evaluate safety condition at each observation point:

Execution Time Monitoring

Evaluate safety condition at each observation point:

code segment $k-$

code segment k

code segment $k+$

Execution Time Monitoring

Evaluate safety condition at each observation point:

code segment $k-$
observation point k
code segment k
observation point $k+1$
code segment $k+$

Execution Time Monitoring

Evaluate safety condition at each observation point:

code segment $k-$		execution time
observation point k		
code segment k	$+$	$WCET_{interference}(k, k+1)$
observation point $k+1$		
code segment $k+$	$+$	$WCET_{isolation}(k+1, end)$
	$+$	observation overhead
	$<$	deadline

Execution Time Monitoring

Evaluate safety condition at each observation point:

code segment $k-$		execution time
observation point k		
code segment k	$+$	$WCET_{interference}(k, k+1)$
observation point $k+1$		
code segment $k+$	$+$	$WCET_{isolation}(k+1, end)$
	$+$	observation overhead
	$<$	deadline

Suspend non-critical tasks if safety condition does not hold.

Execution Time Monitoring – Pros and Cons

Pros

Cons

Execution Time Monitoring – Pros and Cons

Pros

- Code instrumentation is easy to implement on COTS

Cons

Execution Time Monitoring – Pros and Cons

Pros

- Code instrumentation is easy to implement on COTS
- Monitoring execution time limits pessimism

Cons

Execution Time Monitoring – Pros and Cons

Pros

- Code instrumentation is easy to implement on COTS
- Monitoring execution time limits pessimism
- Evaluating condition at observation points limits pessimism

Cons

Execution Time Monitoring – Pros and Cons

Pros

- Code instrumentation is easy to implement on COTS
- Monitoring execution time limits pessimism
- Evaluating condition at observation points limits pessimism

Cons

- Limited to 2 levels, critical and non-critical

Execution Time Monitoring – Pros and Cons

Pros

- Code instrumentation is easy to implement on COTS
- Monitoring execution time limits pessimism
- Evaluating condition at observation points limits pessimism

Cons

- Limited to 2 levels, critical and non-critical
- Large execution time overhead of monitoring

Resource Management for Mixed-Criticality Systems

Workload Arrival Monitoring

Workload Arrival Monitoring

Workload Arrival Function (WAF) for each Task

Workload Arrival Monitoring

Workload Arrival Function (WAF) for each Task

- Task is activated by events

Workload Arrival Monitoring

Workload Arrival Function (WAF) for each Task

- Task is activated by events
- The events are mapped to WCET values

Workload Arrival Monitoring

Workload Arrival Function (WAF) for each Task

- Task is activated by events
- The events are mapped to WCET values
- WAF accumulates required execution time

Workload Arrival Monitoring

Workload Arrival Function (WAF) for each Task

- Task is activated by events
- The events are mapped to WCET values
- WAF accumulates required execution time

Arrival Monitoring and Admission

Workload Arrival Monitoring

Workload Arrival Function (WAF) for each Task

- Task is activated by events
- The events are mapped to WCET values
- WAF accumulates required execution time

Arrival Monitoring and Admission

- Actual workload is calculated based on WAFs

Workload Arrival Monitoring

Workload Arrival Function (WAF) for each Task

- Task is activated by events
- The events are mapped to WCET values
- WAF accumulates required execution time

Arrival Monitoring and Admission

- Actual workload is calculated based on WAFs
- Check whether workload would exceed serviceable level

Workload Arrival Monitoring

Workload Arrival Function (WAF) for each Task

- Task is activated by events
- The events are mapped to WCET values
- WAF accumulates required execution time

Arrival Monitoring and Admission

- Actual workload is calculated based on WAFs
- Check whether workload would exceed serviceable level

Individual WAFs enforce interference bounds between real-time tasks

Group Monitoring Scheme

Group Monitoring Scheme

Criticality Groups

Group Monitoring Scheme

Criticality Groups

- Tasks of one criticality level are grouped together

Group Monitoring Scheme

Criticality Groups

- Tasks of one criticality level are grouped together
- A group of tasks is one virtual task for workload arrival

Group Monitoring Scheme

Criticality Groups

- Tasks of one criticality level are grouped together
- A group of tasks is one virtual task for workload arrival
- Correlation of task activations improves utilization

Group Monitoring Scheme

Criticality Groups

- Tasks of one criticality level are grouped together
- A group of tasks is one virtual task for workload arrival
- Correlation of task activations improves utilization

Combining Monitors

Group Monitoring Scheme

Criticality Groups

- Tasks of one criticality level are grouped together
- A group of tasks is one virtual task for workload arrival
- Correlation of task activations improves utilization

Combining Monitors

- Monitors for groups separately

Group Monitoring Scheme

Criticality Groups

- Tasks of one criticality level are grouped together
- A group of tasks is one virtual task for workload arrival
- Correlation of task activations improves utilization

Combining Monitors

- Monitors for groups separately
- Each monitor respects WAF upper bounds

Group Monitoring Scheme

Criticality Groups

- Tasks of one criticality level are grouped together
- A group of tasks is one virtual task for workload arrival
- Correlation of task activations improves utilization

Combining Monitors

- Monitors for groups separately
- Each monitor respects WAF upper bounds
- Monitors are synchronized by guarantee interface tuples

Group Monitoring Scheme

Criticality Groups

- Tasks of one criticality level are grouped together
- A group of tasks is one virtual task for workload arrival
- Correlation of task activations improves utilization

Combining Monitors

- Monitors for groups separately
- Each monitor respects WAF upper bounds
- Monitors are synchronized by guarantee interface tuples
- Pareto interface tuples with maximized individual WAFs

Workload Arrival Monitoring – Pros and Cons

Workload Arrival Monitoring – Pros and Cons

Properties

Workload Arrival Monitoring – Pros and Cons

Properties

- **Event-triggered**

Workload Arrival Monitoring – Pros and Cons

Properties

- **Event-triggered**
- **Considering worst-case is highly pessimistic**

Workload Arrival Monitoring – Pros and Cons

Properties

- **Event-triggered**
- **Considering worst-case is highly pessimistic**
- **Group monitoring provides more accurate assumptions**

Resource Management for Mixed-Criticality Systems

Goals of Mixed-Criticality Communication Protocols

Resource Utilization

- Enable the derivation of tight latency bounds for guaranteed latency traffic
- Analyzable in terms of schedulability

Goals of Mixed-Criticality Communication Protocols

Resource Utilization

- Enable the derivation of tight latency bounds for guaranteed latency traffic
- Analyzable in terms of schedulability

Quality of Service

- Reduce the average latency of best-effort traffic
- Not applicable to guaranteed latency traffic

Goals of Mixed-Criticality Communication Protocols

Resource Utilization

- Enable the derivation of tight latency bounds for guaranteed latency traffic
- Analyzable in terms of schedulability

Quality of Service

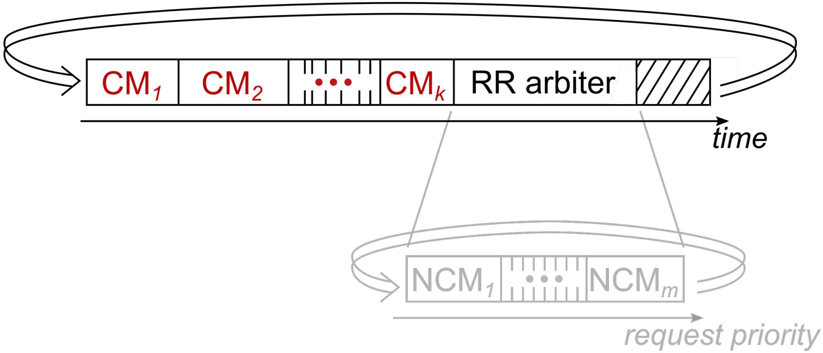
- Reduce the average latency of best-effort traffic
- Not applicable to guaranteed latency traffic

Scalability

- Scale well with the number of bus masters or NoC nodes

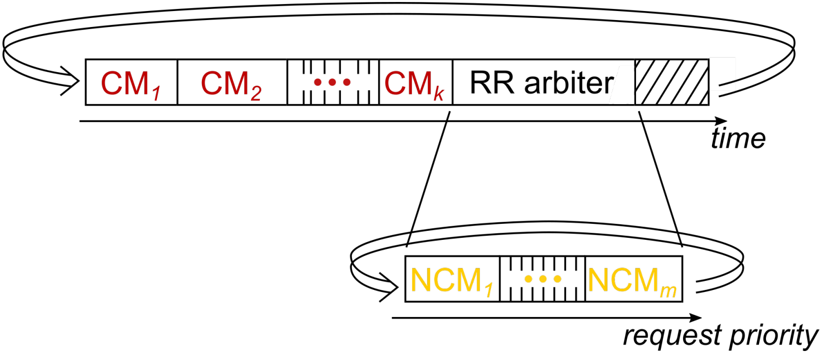
TDMA-RR Dual Layer Arbiter

- First layer : TDMA for critical bus masters
- Second layer : RR for non-critical bus masters



TDMA-RR Dual Layer Arbiter

- First layer : TDMA for critical bus masters
- Second layer : RR for non-critical bus masters



Criticality- and Requirement-Aware Bus Arbiter

- First layer : arbitration between criticality classes
- Second layer : arbitration between tasks of the elected criticality class

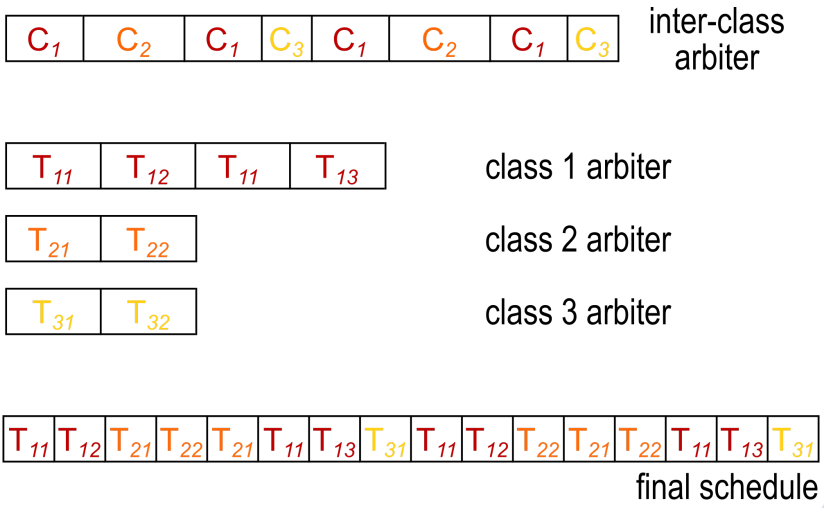
Criticality- and Requirement-Aware Bus Arbiter

- First layer : arbitration between criticality classes
- Second layer : arbitration between tasks of the elected criticality class
- Both layers are Weighted Harmonic RR

Weighted Harmonic Round Robin

- Bus control granted for 1 request only
- Each master can get several slots per period
- Slots are evenly distributed

CArb - Example



C_1 C_2 C_1 C_3 C_1 C_2 C_1 C_3
inter-class arbiter
T_{11} T_{12} T_{11} T_{13}
class 1 arbiter
T_{21} T_{22}
class 2 arbiter
T_{31} T_{32}
class 3 arbiter
T_{11} T_{12} T_{21} T_{22} T_{21} T_{11} T_{13} T_{31} T_{11} T_{12} T_{22} T_{21} T_{22} T_{11} T_{13} T_{31}
final schedule

CArb - Arbiter-level Mode Switch

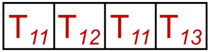
$\mathrm{WCET}_{\mathit{total}}(\mathrm{T}_{\mathit{13}}) < \mathit{exec\_time}(\mathrm{T}_{\mathit{13}})$

$\rightarrow$ Drop low-critical tasks, system-wide mode switch

CArb - Arbiter-level Mode Switch

$\text{WCET}_{total}(\text{T}_{13}) < exec\_time(\text{T}_{13})$
$\rightarrow$ Drop low-critical tasks, system-wide mode switch

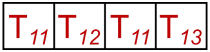
$\text{WCET}_{iso}(\text{T}_{13}) < exec\_time(\text{T}_{13}) < \text{WCET}_{total}(\text{T}_{13})$
$\rightarrow$ Cut interference, arbiter-level mode switch



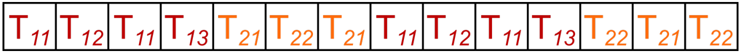
CArb - Arbiter-level Mode Switch

$\text{WCET}_{total}(\text{T}_{13}) < exec\_time(\text{T}_{13})$
$\rightarrow$ Drop low-critical tasks, system-wide mode switch

$\text{WCET}_{iso}(\text{T}_{13}) < exec\_time(\text{T}_{13}) < \text{WCET}_{total}(\text{T}_{13})$
$\rightarrow$ Cut interference, arbiter-level mode switch

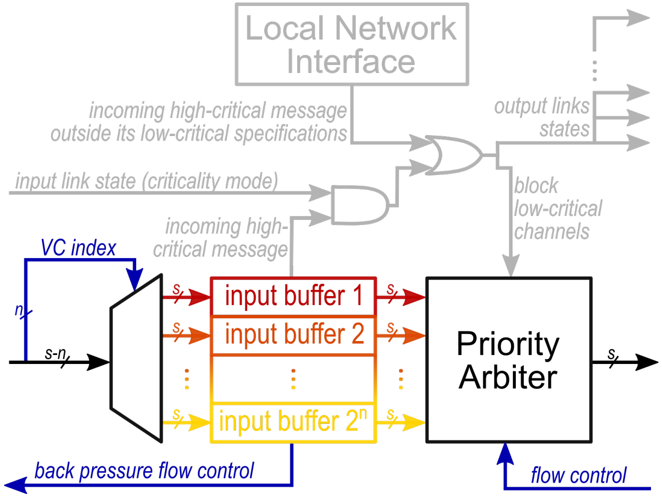


If $exec\_time(\text{T}_{13}) - \text{WCET}_{iso}(\text{T}_{13}) < threshold$
$\rightarrow$ Cut only part of the interference



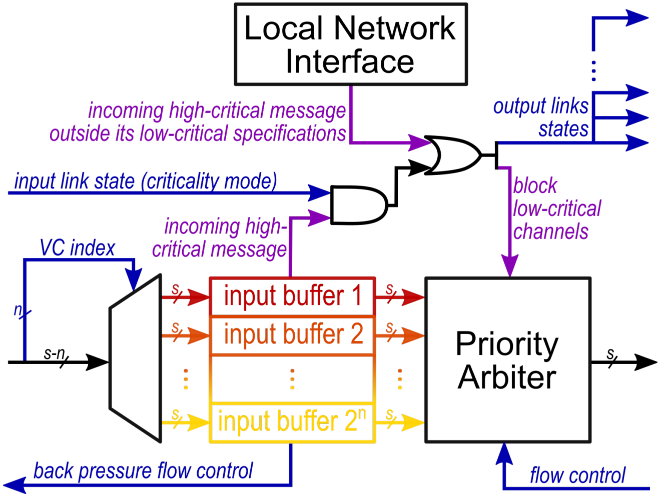
Resource Management for Mixed-Criticality Systems

Wormhole Protocol for Mixed-Criticality

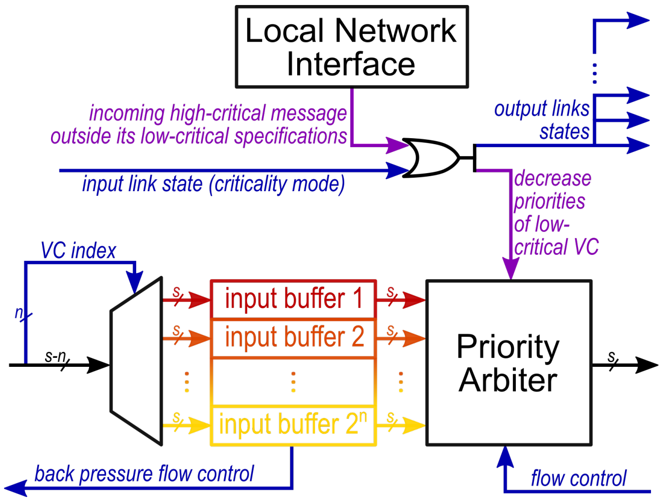


Local Network Interface
incoming high-critical message outside its low-critical specifications
input link state (criticality mode)
output links states
block low-critical channels
incoming high-critical message
VC index
n
s-n
s
input buffer 1
input buffer 2
input buffer 2^n
Priority Arbiter
back pressure flow control
flow control

Wormhole Protocol for Mixed-Criticality


Local Network Interface
incoming high-critical message outside its low-critical specifications
output links states
input link state (criticality mode)
block low-critical channels
incoming high-critical message
VC index
n
s-n
s
input buffer 1
input buffer 2
input buffer 2^n
Priority Arbiter
back pressure flow control
flow control

WPMC-Flood


Local Network Interface
incoming high-critical message outside its low-critical specifications
input link state (criticality mode)
output links states
decrease priorities of low-critical VC
VC index
n
s-n
s
input buffer 1
input buffer 2
⋮
input buffer 2ⁿ
Priority Arbiter
back pressure flow control
flow control

Wormhole with Blocking Counter

The header flit of critical messages contains a *Blocking Counter* field.

<u>Network Interface</u>

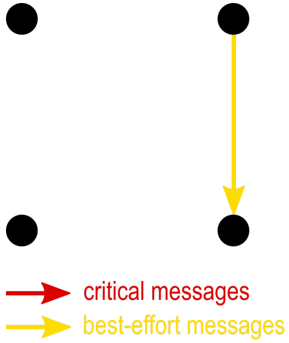
```
initialize BC value

(with maximum
number of times
the message can
be delayed)
```

<u>Router</u>

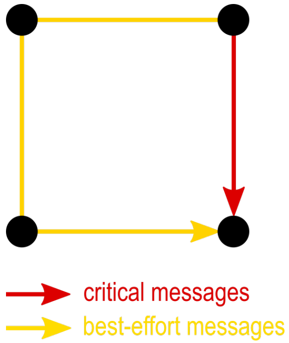
```
if message is stalled
    decrement BC
    if BC = 0
        prioritize message
    end if
end if
```

Adaptive Routing



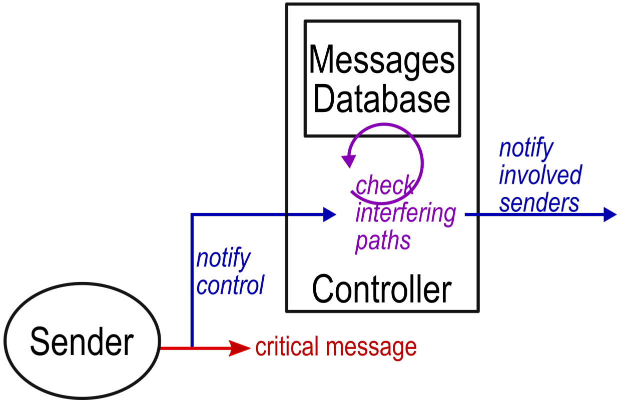
- Message BE1
  Path list : {(1,2), (2,2)}, {(1,2), (1,1), (2,1), (2,2)}

Adaptive Routing



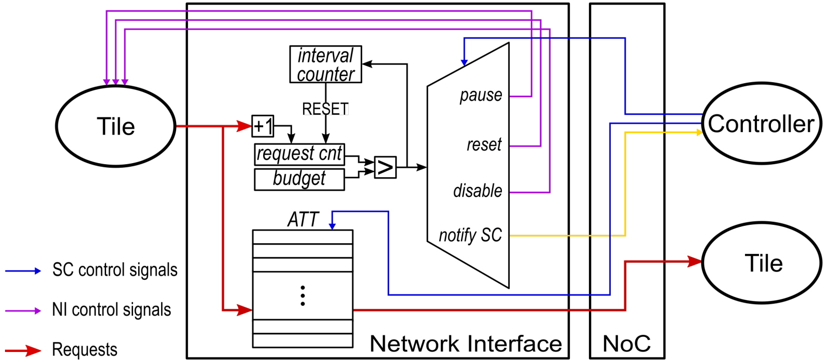
- Message BE1
  Path list : {(1,2), (2,2)}, {(1,2), (1,1), (2,1), (2,2)}
- Message C1
  Path list: {(1,2), (2,2)}

Adaptive Routing - Control Scheme



Resource Management for Mixed-Criticality Systems

Integrated Dependable Architecture for Many-Cores



Integrating Scheduling and Data mapping

Hardware model:

- 16 Clusters

Integrating Scheduling and Data mapping

Hardware model:

- 16 Clusters
- Each containing 16 cores, 3 NoC interfaces, and 16 SRAM banks

Integrating Scheduling and Data mapping

Hardware model:

- 16 Clusters
- Each containing 16 cores, 3 NoC interfaces, and 16 SRAM banks
- Each SRAM bank has its own dedicated controller, which arbitrates between all cores and interfaces

Integrating Scheduling and Data mapping

Hardware model:

- 16 Clusters
- Each containing 16 cores, 3 NoC interfaces, and 16 SRAM banks
- Each SRAM bank has its own dedicated controller, which arbitrates between all cores and interfaces
  - NoC receiver interface always has priority
  - RR arbitration is performed between other components

Integrate Scheduling and Data mapping

Software model:

- Independent task set and scheduler for each cluster

Integrate Scheduling and Data mapping

Software model:

- Independent task set and scheduler for each cluster
- Criticality mode switch

Integrate Scheduling and Data mapping

Software model:

- Independent task set and scheduler for each cluster
- Criticality mode switch
- Scheduling data:
  - Execution times **without memory accesses**
  - Number of memory accesses
  - Dependency graph

Integrate Scheduling and Data mapping

Software model:

- Independent task set and scheduler for each cluster
- Criticality mode switch
- Scheduling data:
    - Execution times **without memory accesses**
    - Number of memory accesses
    - Dependency graph
- Tasks of different criticality cannot be scheduled on the same time frame

Integrate Scheduling and Data mapping

Software model:

- Independent task set and scheduler for each cluster
- Criticality mode switch
- Scheduling data:
    - Execution times **without memory accesses**
    - Number of memory accesses
    - Dependency graph
- Tasks of different criticality cannot be scheduled on the same time frame
- Joint computation of task schedule and data mapping to SRAM banks

Controlling interference from other tiles

When a task requires data to be fetched through the NoC

$$\begin{tikzcd}
\textit{Task A} \arrow[r] \arrow[d, "\textit{dependency constraint: delay t}"] & \textit{send request for data} \\
\textit{Task B} \arrow[r] & \textit{use data}
\end{tikzcd}$$

$t = 2 * \text{NoC worst-case latency} + \text{resource worst-case response time}$

Higher level modeling - Resource Servers

- A resource server "encapsulates" the access to a resource

Higher level modeling - Resource Servers

- A resource server "encapsulates" the access to a resource
- This can allow to consider only the server and the requesting component in modeling

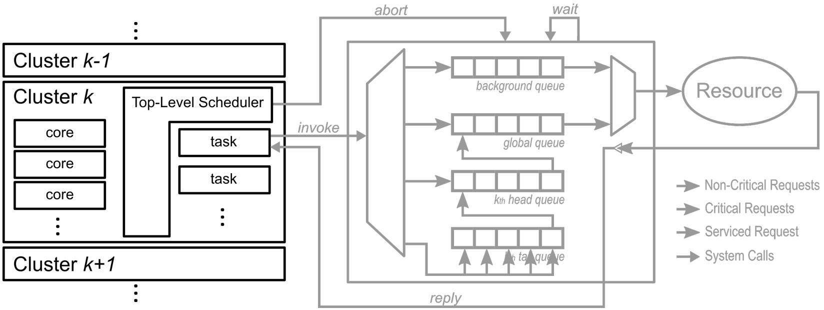
Higher level modeling - Resource Servers

- A resource server "encapsulates" the access to a resource
- This can allow to consider only the server and the requesting component in modeling
- The resource itself and the interfering components can be laid aside **provided the inter-process communication protocol is adequate**

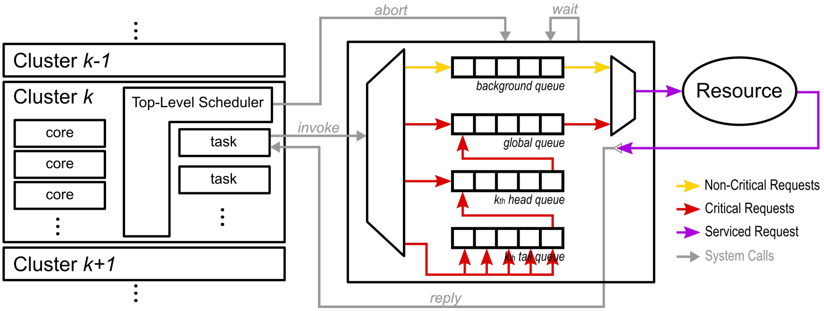
Higher level modeling - Resource Servers

- A resource server "encapsulates" the access to a resource
- This can allow to consider only the server and the requesting component in modeling
- The resource itself and the interfering components can be laid aside **provided the inter-process communication protocol is adequate**
- For MCS, the IPC protocol should also take criticality into account

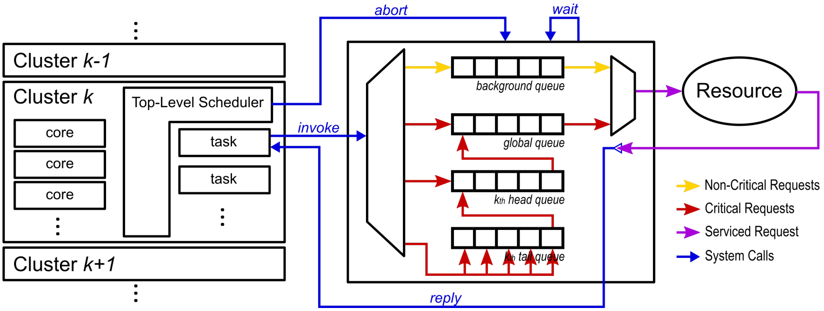
Resource Server for MCS



Resource Server for MCS



Resource Server for MCS



Resource Management for Mixed-Criticality Systems

Mixed Criticality Systems

- Response to a pressing industrial need:

Mixed Criticality Systems

- Response to a pressing industrial need:
    - How to provide safety and real-time guarantees in efficient implementations?

Mixed Criticality Systems

- Response to a pressing industrial need:
    - How to provide safety and real-time guarantees in efficient implementations?
- Mixed Criticality scheduling Theory;

Mixed Criticality Systems

- Response to a pressing industrial need:
  - How to provide safety and real-time guarantees in efficient implementations?
- Mixed Criticality scheduling Theory;
  - Design time verification;

Mixed Criticality Systems

- Response to a pressing industrial need:
  - How to provide safety and real-time guarantees in efficient implementations?
- Mixed Criticality scheduling Theory;
  - Design time verification;
  - Run-time robustness, when the system violates assumptions;

Mixed Criticality Systems

- Response to a pressing industrial need:
  - How to provide safety and real-time guarantees in efficient implementations?
- Mixed Criticality scheduling Theory;
  - Design time verification;
  - Run-time robustness, when the system violates assumptions;
- Providing sound mechanisms for sharing:

Mixed Criticality Systems

- Response to a pressing industrial need:
  - How to provide safety and real-time guarantees in efficient implementations?
- Mixed Criticality scheduling Theory;
  - Design time verification;
  - Run-time robustness, when the system violates assumptions;
- Providing sound mechanisms for sharing:
  - Processing elements;

Mixed Criticality Systems

- Response to a pressing industrial need:
  - How to provide safety and real-time guarantees in efficient implementations?
- Mixed Criticality scheduling Theory;
  - Design time verification;
  - Run-time robustness, when the system violates assumptions;
- Providing sound mechanisms for sharing:
  - Processing elements;
  - Buffers, caches, memory;

Mixed Criticality Systems

- Response to a pressing industrial need:
  - How to provide safety and real-time guarantees in efficient implementations?
- Mixed Criticality scheduling Theory;
  - Design time verification;
  - Run-time robustness, when the system violates assumptions;
- Providing sound mechanisms for sharing:
  - Processing elements;
  - Buffers, caches, memory;
  - I/O: Pins, drivers;

Mixed Criticality Systems

- Response to a pressing industrial need:
  - How to provide safety and real-time guarantees in efficient implementations?
- Mixed Criticality scheduling Theory;
  - Design time verification;
  - Run-time robustness, when the system violates assumptions;
- Providing sound mechanisms for sharing:
  - Processing elements;
  - Buffers, caches, memory;
  - I/O: Pins, drivers;
  - **Communication.**

¿ Questions ?