MIRROR: Symmetric Timing Analysis for Real-Time Tasks on Multicore Platforms with Shared Resources

Wen-Hung Kevin Huang, Jian-Jia Chen, and Jan Reineke

TU Dortmund and University of Saarlands

09,06,2016 at DAC, Austin, USA
Periodic Control System (Liu and Layland 1973)

Pseudo-code for this system

set timer to interrupt periodically with period $T$;

at each timer interrupt
do

• perform analog-to-digital conversion to get $y$;
• compute control output $u$;
• output $u$ and do digital-to-analog conversion;

od
Periodic Control System (Liu and Layland 1973)

**Pseudo-code for this system**

set timer to interrupt periodically with period $T$;

at each timer interrupt

**do**

- perform analog-to-digital conversion to get $y$;
- compute control output $u$;
- output $u$ and do digital-to-analog conversion;

**od**

**Control System**

- A/D
- Control–law computation
- D/A
- sensor
- plant (The system being controlled)
- actuator

**Liu and Layland Model:**

- $T_i$: period of task $\tau_i$
- $D_i$: relative deadline of task $\tau_i$
- $C_i$: worst-case execution time of task $\tau_i$
- $U_i$: utilization $C_i / T_i$
Task is executed on Core 1
Predictability Due to Resource Sharing

L1 cache misses

Multi Core CPU 1

Multi Core CPU 2
Predictability Due to Resource Sharing

L2 cache misses

Multi Core CPU 1

Multi Core CPU 2

Core 1

Core 2

L1 Cache

L2 Cache

Main Memory

L2 Cache

L1 Cache

Core 1

Core 2
Predictability Due to Resource Sharing

Access memory

Multi Core CPU 1

Multi Core CPU 2
Task is executed on Core 2
Predictability Due to Resource Sharing

Multi Core CPU 1

Core 1
L1 Cache
L2 Cache

Core 2
L1 Cache

Multi Core CPU 2

Core 1
L1 Cache

Core 2
L1 Cache

L1 cache misses
Predictability Due to Resource Sharing

L2 is blocked

Multi Core CPU 1

Multi Core CPU 2
Predictability Due to Resource Sharing

Task is executed on Core 3
Predictability Due to Resource Sharing

L1 cache misses

Multi Core CPU 1

Multi Core CPU 2
Predictability Due to Resource Sharing

L2 cache misses

Multi Core CPU 1

Multi Core CPU 2
Memory access is blocked
Memory access for core 1 finishes
Typical Two-Phase Analysis Approaches

- Phase 1: Worst-case execution time (WCET) of a stand-alone program
  - WCET analyzers such as aIT or Chronous.
Typical Two-Phase Analysis Approaches

- Phase 1: Worst-case execution time (WCET) of a stand-alone program
  - WCET analyzers such as aiT or Chronous.
- Phase 2: Worst-case response time (WCRT) of a periodic/sporadic task by considering the competition with the other tasks
  - worst-case interference from the other tasks
  - utilization-based tests, response time analysis, busy-interval techniques, real-time calculus, max-plus algebra, etc.
- The notion of WCET is destroyed in multicore systems due to shared resources.
  - WCET depends on how the tasks on the other cores are co-executed
  - Assume the worst-case interference is too pessimistic
Self-Suspending Behavior

- Multiple cores may share a bus
- The contention on the bus can be considered as a suspension problem (with respect to the bus access)
Suppose that we know the suspension time of each $\tau_i$ and would like to analyze the schedulability of the tasks on a core. (Constrained-deadline $D_i \leq T_i$)

- A self-suspending task $\tau_i$ is a periodic task with jitter (PJD task)
  - Period: $T_i$
  - Self-suspension-time: $S_i$
Suppose that we know the suspension time of each $\tau_i$ and would like to analyze the schedulability of the tasks on a core. (Constrained-deadline $D_i \leq T_i$)

- A self-suspending task $\tau_i$ is a periodic task with jitter (PJD task)
  - Period: $T_i$
  - Self-suspension-time: $S_i$
- Schedulability test of task $\tau_k$:
  \[
  \exists t \text{ with } 0 < t \leq T_k \text{ and } C_k + S_k + \sum_{j=1}^{k-1} \left\lfloor \frac{t + S_j}{T_j} \right\rfloor C_j \leq t.
  \]
Self-Suspension Tasks in Real-Time Systems

Suppose that we know the suspension time of each $\tau_i$ and would like to analyze the schedulability of the tasks on a core. (Constrained-deadline $D_i \leq T_i$)

- A self-suspending task $\tau_i$ is a periodic task with jitter (PJD task)
  - Period: $T_i$
  - Self-suspension-time: $S_i$
- Schedulability test of task $\tau_k$:
  \[ \exists t \text{ with } 0 < t \leq T_k \text{ and } C_k + S_k + \sum_{j=1}^{k-1} \left\lceil \frac{t + D_j}{T_j} \right\rceil C_j \leq t. \]
Self-Suspension Tasks in Real-Time Systems

Suppose that we know the suspension time of each $\tau_i$ and would like to analyze the schedulability of the tasks on a core. (Constrained-deadline $D_i \leq T_i$)

- A self-suspending task $\tau_i$ is a periodic task with jitter (PJD task)
  - Period: $T_i$
  - Self-suspension-time: $S_i$

- Schedulability test of task $\tau_k$:

$$\exists t \text{ with } 0 < t \leq T_k \text{ and } C_k + S_k + \sum_{j=1}^{k-1} \left\lceil \frac{t + T_j}{T_j} \right\rceil C_j \leq t.$$
Platform Model

- Multicore with a share resource
- For example, atomic (non-split-transaction) bus
  - Bus sits idle while memory processes the request and sends the response
- Fixed-priority arbitration
Task and Scheduling Model

- Resource access task $\tau_i$ ($C_i, A_i, T_i, D_i, \sigma_i$)
  - $C_i$: upper bound on local computation
  - $A_i$: upper bound on resource accesses
  - $T_i$: period
  - $D_i$: relative deadline ($D_i \leq T_i$)
  - $\sigma_i$: the maximum number segments of consecutive resource accesses

- Path analysis

- Fixed-priority scheduling (we use deadline-monotonic scheduling)
Task and Scheduling Model

- Resource access task $\tau_i (C_i, A_i, T_i, D_i, \sigma_i)$
  - $C_i$: upper bound on local computation
  - $A_i$: upper bound on resource accesses
  - $T_i$: period
  - $D_i$: relative deadline ($D_i \leq T_i$)
  - $\sigma_i$: the maximum number segments of consecutive resource accesses

- Path analysis

- Fixed-priority scheduling (we use deadline-monotonic scheduling)
Task and Scheduling Model

- Resource access task $\tau_i$ ($C_i, A_i, T_i, D_i, \sigma_i$)
  - $C_i$: upper bound on local computation
  - $A_i$: upper bound on resource accesses
  - $T_i$: period
  - $D_i$: relative deadline ($D_i \leq T_i$)
  - $\sigma_i$: the maximum number segments of consecutive resource accesses

- Path analysis
- Fixed-priority scheduling (we use deadline-monotinic scheduling)
Task and Scheduling Model

- Resource access task $\tau_i$ ($C_i, A_i, T_i, D_i, \sigma_i$)
  - $C_i$: upper bound on local computation
  - $A_i$: upper bound on resource accesses
  - $T_i$: period
  - $D_i$: relative deadline ($D_i \leq T_i$)
  - $\sigma_i$: the maximum number segments of consecutive resource accesses

- Path analysis

- Fixed-priority scheduling (we use deadline-monotonic scheduling)

Assume compositional properties: 75 is a safe upper bound.
Key Observations: Symmetric Property

- From the core perspectives for $\tau_2$
  - accessing or waiting: $[3,4), [8,12), [15, 16)$
  - suspension: $[4,8), [12, 15)$

- From the shared resource perspectives for $\tau_2$
  - executing or waiting: $[4,8), [12, 15)$
  - suspension: $[3,4), [8,12), [15, 16)$
**Schedulability Test for Task** $\tau_k$

- WCRT is upper bounded by the minimum $t|0 < t \leq D_k$

\[
(C_k + \text{exec\_core}(t)) + (A_k + \text{exec\_resource}(t)) \leq t
\]
Schedulability Test for Task $\tau_k$

- WCRT is upper bounded by the minimum $t | 0 < t \leq D_k$

$$
\left( C_k + \sum_{\tau_i \in hp(\tau_k, c)} \left\lceil \frac{t + T_i}{T_i} \right\rceil C_i \right) + \sigma_k B + \left( A_k + \sum_{\tau_i \in hp(\tau_k, r)} \left\lceil \frac{t + T_i}{T_i} \right\rceil A_i \right) \leq t
$$

- $\sigma_k B$: the maximum blocking time by the lower priority tasks on the shared resource
- $hp(\tau_k, c)$: higher-priority tasks than $\tau_k$ on the same core
- $hp(\tau_k, r)$: higher-priority tasks than $\tau_k$ on shared resource
Schedulability Test for Task $\tau_k$

- WCRT is upper bounded by the minimum $t | 0 < t \leq D_k$

$$\left( \frac{C_k}{T_i} \sum_{\tau_i \in hp(\tau_k, c)} \left\lceil \frac{t + T_i}{T_i} \right\rceil C_i \right) + \sigma_k B + \left( \frac{A_k}{T_i} \sum_{\tau_i \in hp(\tau_k, r)} \left\lceil \frac{t + T_i}{T_i} \right\rceil A_i \right) \leq t$$

- $\sigma_k B$: the maximum blocking time by the lower priority tasks on the shared resource
- $hp(\tau_k, c)$: higher-priority tasks than $\tau_k$ on the same core
- $hp(\tau_k, r)$: higher-priority tasks than $\tau_k$ on shared resource

- Pessimism of the above response time analysis: number of resource access segments was not exploited

- In our paper, we explain how to calculate and utilize the information $\sigma_k$ in a symmetric and more precise manner
• Schedulability tests are based on the previous slide.
• Fitting can be First-Fit (FF), Worst-Fit (WF), Best-Fit (BF)
Schedulability tests are based on the previous slide.

Fitting can be First-Fit (FF), Worst-Fit (WF), Best-Fit (BF)
Schedulability tests are based on the previous slide.

Fitting can be First-Fit (FF), Worst-Fit (WF), Best-Fit (BF)
• Schedulability tests are based on the previous slide.
• Fitting can be First-Fit (FF), Worst-Fit (WF), Best-Fit (BF)
• Schedulability tests are based on the previous slide.
• Fitting can be First-Fit (FF), Worst-Fit (WF), Best-Fit (BF)
• Schedulability tests are based on the previous slide.
• Fitting can be First-Fit (FF), Worst-Fit (WF), Best-Fit (BF)
Schedulability tests are based on the previous slide.

Fitting can be First-Fit (FF), Worst-Fit (WF), Best-Fit (BF)
Schedulability tests are based on the previous slide.

Fitting can be First-Fit (FF), Worst-Fit (WF), Best-Fit (BF)
Task Assignment (Partition)

- Schedulability tests are based on the previous slide.
- Fitting can be First-Fit (FF), Worst-Fit (WF), Best-Fit (BF)
Schedulability tests are based on the previous slide.
Fitting can be First-Fit (FF), Worst-Fit (WF), Best-Fit (BF)
Experiments

- Configuration
  - 4-core platform ($m=4$)
  - 20 tasks
  - Periods [10-1000ms]
  - Each utilization level: 100 task sets

- Existing results:
  - Exact-MC (Bonifaci et al. in RTNS 2015): do memory access and then do execution
  - MIRROR-SPIN (This resembles the test from Altmeyer et al. in RTNS 2015)

- Evaluation Metrics:
  - The acceptance ratio of a level: the number of task sets that are schedulable by the test divided by the number of task sets.
The number of resource access segments $\sigma_i$: 1 (rare access, type=R), 2 (moderate access, type=M), and 10 (frequent access, type=F).
Conclusion and Extensions

- Fixed-priority, deadline-monotonic scheduling bus + bus-aware timing analysis + FFDM = high schedulability
  - A general treatment to handle multicore resource accesses
  - The treatment is compatible with existing task partitioning methods
  - The view points are symmetric
  - First result with worst-case resource augmentation guarantees (i.e., speedup factors) for this research line

- Extensions
  - Similar techniques can be applied for multiple shared resources
  - The pessimism can be further reduced by counting the interference more precisely