CONTENTION IN MULTICORE HARDWARE SHARED RESOURCES: UNDERSTANDING OF THE STATE OF THE ART

Gabriel Fernandez¹, Jaume Abella², Eduardo Quiñones², Christine Rochange³, Tullio Vardanega⁴ and Francisco J. Cazorla²,⁴

14th International Workshop on Worst-Case Execution Time Analysis (WCET 2014)
Multicores: benefits and challenges

• Multicores
  – Allow higher “guaranteed performance”
    • Guaranteed as opposed to average-case
  – Interference on execution time and WCET due to contention in the access to HW shared resources
    • Challenge timing analysis
    • Higher impact than in singlecore

• Contention in multicores has been deeply studied by the research community
  – Different approaches taken to contention
    • At different levels of abstraction
  – The solutions space is difficult to fully understand
Motivation of this work

• Provide a sensible taxonomy of the SoA techniques
  – Identifying ‘families’ of techniques
  – Singling out representative works for each class
    • Without seeking absolutely exhaustive coverage

• Review each family
  – Seeking overlaps and gaps with others
  – Understanding assumptions and challenges of use
  – Gaging confidence in WCET bounds and assurance guarantees for industrial use

• Capture cross-cutting techniques
Taxonomy

System Centric
- Time Analysis Frameworks
  - Contention oblivious
  - Contention aware
- Task Assignment and Scheduling
  - Joint Analysis
  - Independent Analysis

WCET Centric
- Handling Contention

Architecture Centric
- Independent Analysis

COTS Centric

Bottom-up / Top-down

Idealistic-innovative / Practical-pragmatic
Timing analysis frameworks

• Assume replicated on-chip resources
  – SW on core suffers no parallel contention

• Model off-chip shared resources in isolation
  – Provide worst-case access timing bounds
  – Contention captured compositionally: off-chip contention in the presence of co-runners
    • TDMA arbiter
      – Co-running tasks do not affect one another’s execution time
      – Worst-case alignment of the requests in the TDMA
    • Dynamic arbiter
      – Co-running tasks do affect one another’s execution time
      – Focus on deriving bounds for the number of accesses per task in a given period of time
Task allocation and scheduling

• Contention oblivious
  – The WCET of all tasks is given in input
    • WCET bounds may be determined before decisions are made on task mapping and on scheduling
  – Escape circularity in the mutual dependence between WCET analysis and schedulability analysis

• Contention aware
  – Focus on the shared last-level cache
  – Benefit from HW techniques for cache partitioning or allocate program data to different pages
  – Assume partitioned scheduling and augment assignment with colouring
WCET-centric

Handling Contention

WCET Centric

Joint Analysis

Independent Analysis
Including contention costs in WCETs

- Stall times integrated in the ILP formulation used to derive WCETs (IPET method)
  - Worst-case memory instruction latencies
  - Worst-case number of L2 cache misses

- Two philosophies to capture worst cases
  - Contextual
    - The set of concurrent threads/tasks is known at analysis time ➔ \textit{joint} analysis
  - Universal
    - Concurrent tasks are unknown ➔ \textit{independent} analysis
    - Needs hardware/software support
Joint analysis of concurrent tasks

• Approach A
  – Iterative computation of interferences

• Approach B
  – Timed automata
    + model checking
      show: WCET(A) < x
Independent analysis

- No assumption on the concurrent workload
  - Independent of task assignment and scheduling

- Requires hardware/software support
  - To derive worst-case latencies and worst-case behaviours
  - Examples include
    - Partitioned caches: eliminate impact from concurrent tasks
    - Static bus arbiters: make it possible to derive worst-case latencies
Architecture-Centric

Handling Contention

Architecture Centric
Hardware support for handling contention

- Bound contention impact on access time to hardware shared resources
  - TTA (<‘00), PRET (’06), CompSOC (‘09), MERASA (‘07), ...  

- Time composability
  - WCET estimates
    - The execution time of a task varies under different workloads its WCET estimate does not
  - Execution time
    - Same execution time under any workload

- Time composability is achieved by ‘resource reservation’ → performance degradation
Hardware support for handling contention

• Bound contention impact on access time to hardware shared resources
  – Indirectly: bandwidth guarantees
  – Directly: access time guarantees

• Type of resources
  – Stateless (e.g. bus): access policy
  – Stateful (e.g. cache): partition to prevent task interaction

• NoC
COTS

Handling Contention

COTS Centric
Challenge

• Time analyzability properties of real COTS multicores
  – No assumptions can be made
  – Analyze hardware shared resources
  – Analyze their impact on execution time
  – Bounds derived by ad-hoc experiments

• Understanding timing behavior of hardware shared resources
  – The way they challenge timing analyzability

• Software cache partitioning on ARM A9
Critique
System-centric

- **Time Analysis frameworks: assumptions**
  - One shared resource, blocking and no split
  - Program broken down into superblocks with resource usage bounds per block
  - Dynamic arbiters
    - WCET estimate dependent on co-runners: this can be tightened but it is no longer time composable

- **Task assignment and scheduling**
  - Static task-to-CPU assignment determines opponents
    - This is good but not enough unless you have a viable technique to avoid exploring the space of all possible contentions
    - Static over-provisioning is never good news and may defeat the purpose
## WCET-centric techniques

### Assumptions

<table>
<thead>
<tr>
<th>Independent analysis</th>
<th>Joint analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static (boundable) arbitration of shared resources</td>
<td>One task per core, schedule known</td>
</tr>
</tbody>
</table>

### Limits

<table>
<thead>
<tr>
<th>Independent analysis</th>
<th>Joint analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pessimism (blind estimation of contention)</td>
<td>Not time composable</td>
</tr>
<tr>
<td></td>
<td>Complexity (state explosion)</td>
</tr>
</tbody>
</table>
Architecture-centric

- Will the proposed designs ever see the silicon?
  - Applies to all hardware designs ;-
  - Cache partitioning mechanisms: won battle
  - Proposed changes are ‘simple’

- Timing Anomalies
  - Design hardware that prevents appearance of TA
COTS-centric

- Architectural support for isolation or controlled contention
  - Not fully adopted!
- This generates uncertainty
  - Build confidence arguments in accordance with requirements and practices of the application domain
  - How safety assurance relate to stipulating bounds on execution time
Concluding remarks

• More understanding of existing techniques is needed
  – Do they form a consistent picture from which a user can choose sensibly?

• What is the top priority for the industrial user
  – *Question for the audience*

• Seeking time composability vs. guaranteed performance
  – First negatively affects the second
  – Not possible in the single-core sense → compositional
Work mainly funded by ...