A Framework for the Optimization of the WCET of Programs on Multi-Core Processors

Max John, Michael Jacobs
Saarland University, Saarbrücken

Introduction

Scenario

- Embedded systems
- Timing-critical applications
- Strict deadlines, e.g. in automotive applications
- Need of tight WCET bounds

Implementation (HW & SW)

- Multi-core processor with shared bus
- Exploit task parallelism
- However: cores interfere
- Usage of a TDMA bus
- Cores no longer interfere
- Use static task scheduling

Challenge

- Find static system schedule and bus schedule
- However: Optimal schedule hard to obtain
- Approximation framework needed

System Model

- Tasks
  - Single execution behavior
  - Determined by length and bus accesses
- System schedule
  - Assigns tasks to processor cores
  - Determines the tasks' execution order
- Bus schedule
  - If a bus request is denied
  - Processor core blocked until access is granted
- Bus schedule heuristic
  - Determines the tasks’ execution order
  - Assigns tasks to processor cores
  - Determined by length and bus accesses
- Data:
  - tasks: set of tasks,
  - \( n \): number of processor cores,
  - \( th \): task selection heuristic,
  - \( bh \): bus schedule heuristic

Optimization Framework

- Goal: Reduce the overall WCET
- By integrated construction of
  - System schedule
  - Bus schedule

Real-World Programs in our System Model

- Real-world programs have multiple behaviors
  - Soundly over-approximate them by a single one
  - Simple overlay
  - Access-aligned overlay

Steps towards more General Systems

- Task dependencies
  - Constraint task selection accordingly
  - Side effect: sometimes none of the remaining tasks selectable
  - Solution: return dummy task \( \tau_d \) in those cases: \( \tau_d \)
- Task priorities
  - Constraint task selection and bus schedule heuristic accordingly
  - Restrict the granularity of the bus schedule
  - Many systems have a bus processor ratio \( K \), i.e.
    \[ \forall n \in \mathbb{N}, n \neq 0 \quad \text{mod} \quad K = \text{bus}(n) = \text{bus}(n - 1) \]
  - Thus some bus schedules are no longer legal
    \[ \text{bus: } P_1, P_2, P_1, P_2, P_1, P_2, P_1, P_2, P_1, P_2 \]
    \[ \Rightarrow \text{Illegal bus schedule for } K = 4 \]
    \[ \text{bus: } P_1, P_2, P_1, P_2, P_1, P_2, P_1, P_2, P_1, P_2 \]
    \[ \Rightarrow \text{Illegal bus schedule for } K = 4 \]
  - Constraint bus heuristic to produce legal schedules only

Future Work

- Develop access-aware task selection heuristics
  - Only non-access-aware heuristics exist
- Experiments
  - Extract traces from real-world programs
  - Evaluate effectiveness of heuristics
  - Soundly combine several traces to one
    - Determine degree of over-approximation

Acknowledgement

AVACS

http://www.uni-saarland.de  http://www.avacs.org

Details

- Data: tasks: set of tasks,
  - \( n \): number of processor cores,
  - \( th \): task selection heuristic,
  - \( bh \): bus schedule heuristic

\( sy(s, b) \leftarrow \) empty schedules for \( n \) processor cores;

\( \text{while} \) tasks \( \neq \emptyset \) \( \text{do} \)

- \( \text{task} \leftarrow th(\text{tasks, sys, bus}); \)
- \( p_{idle} \leftarrow \) find first idle core in \( sys \), \( bus \);
- \( sys \leftarrow \text{add task in sys to } p_{idle}; \)
- \( bus \leftarrow bh(sys, bus, "partial"); \)
- \( \text{tasks} \leftarrow \text{tasks} \setminus \{\text{task}\}; \)

- \( \text{end} \)
- \( \text{bus} \leftarrow bh(sys, bus, "complete"); \)
- \( \text{return} (sys, bus); \)

- Modularity: plug in heuristics
  - Task selection heuristic \( th \)
  - Bus schedule heuristic \( bh \)