Hardware Support for WCET Analysis of Hard Real-Time Multicore Systems

Marco Paolieri (BSC/UPC)
Eduardo Quiñones (BSC)
Francisco J. Cazorla (BSC)
Guillem Bernat (Rapita Systems)
Mateo Valero (BSC/UPC)

36th ISCA 2009
Austin, Texas, 22nd June
Real-time embedded systems

- Real-time embedded systems (RTESs) are in everyday life

- RTESs require correctness in the value and in the time domain

- Time correctness:
  - Applications must finish before a given deadline
  - In safety-critical real time embedded systems missing a deadline can have catastrophic consequences!
To ensure correct timing behavior of HRT

- WCET analysis

Each HRT is characterized by

- A Deadline
- A Worst-Case Execution Time (WCET)

**Abbreviations:**
- **d**: deadline
- **WCET**: Worst Case Execution Time
- **Avg ET**: Average Execution Time
- **BCET**: Best Case Execution Time
- **WCET_{EST}**: Worst Case Execution Time
Multicores in RTESs: advantages

- Current real-time embedded systems require higher performance than provided by current processors
  - Increasing safety, comfort, number and quality of services
- Multicore processors represent a good solution

!! but multicores have a drawback ... !!

- Maintain the design of the core simple
- Low cost
- Low power consumption
Multicores in RTESs: disadvantages

- It is harder to perform WCET analysis for multicore processors than for single-core because of **Inter-thread Interferences**

  - Inter-thread interferences accessing shared resources make the execution time vary

Where:

- \( WCET_a \) \( \) WCET est without interferences
- \( d \) \( \) deadline
- \( ET_{a,b} \) \( \) ET of \( a \) running with \( b \)
- \( ET_{a,c} \) \( \) ET of \( a \) running with \( c \)

- Execution time, and so the WCET of a HRT depend on the workload
Multicores in RTESs: disadvantages

- Hard real-time systems (e.g. automotive) are composed by tasks developed by different sub-suppliers
- In a multicore environment changing a task requires to **re-analyze** the whole system again

Multicore offer advantages to RTESs BUT it is required to deal with Inter-thread interferences
Agenda

- **Intuitive Solutions to enable CMPs in RTESs**

- **Our solution**
  - Proposal 1: Multicore architecture enforcing an UBD
    - On-chip shared bus
    - L2 cache
  - Proposal 2: The WCET computation mode

- **Experimental results**

- **Conclusions**
Intuitive Solutions to enable CMPs in RTESs

- Complete analysis of how interferences from different threads affect their execution time (i.e. run all possible thread combinations)
  - It is required to consider the entire workload a priori
    - The exact timing of each request for every task that composing the workload accessing a shared resource
  - Every time a task changes the system must be re-analyzed

- Static partitioning of resources for each core
  - Each core is given a fixed part of the cache and using TDMA a fixed bandwidth
  - Advantages: easy to provide WCET estimations
  - Disadvantages: Low performance
Agenda

- Intuitive Solutions to enable CMPs in RTESs

- **Our solution**
  - Proposal 1: Multicore architecture guaranteeing an UBD
    - On-chip shared bus
    - L2 cache
  - Proposal 2: The WCET computation mode

- Experimental results

- Conclusions
Our Proposals

- (P1) Our new multicore architecture guarantees by design that the maximum delay a request accessing a shared resource may suffer due to inter-thread interferences has an Upper Bound Delay (UBD)

- Inter-thread interferences < UBD

- (P2) On top of this multicore processor architecture, we introduce an execution mode in our multicore that allows computing a WCET estimation of HRT based on the UBD
We focus on

- Shared BUS
- L2 cache
Computing an UBD: Shared Bus (P1)

- The existence of UBD depends on the arbitration policy

  - **Fixed priority** does NOT provide UBD
    
    - Fixed priority
    
    - Round Robin provides UBD based on the number of requestors
      
      \[
      UBD = (N_{HRT} - 1) \times L_{BUS}
      \]
      
      \[
      UBD = (2-1) \times 2 = 2
      \]
      
      where
      
      \( L_{BUS} \) is the latency of the bus and
      \( N_{HRT} \) the total number of HRTs running at the same time
Computing an UBD: Our BUS Arbiter (P1)

- Two level bus arbiter
  - Intra-Core Bus Arbiter (ICBA)
    - Intra-thread interferences
  - Inter-Core Bus Arbiter (XCBA)
    - Inter-thread interferences

- XCBA
  - Round Robin between different HRT requests
  - HRT requests have priority over NHRT requests to reduce the effect of NHRT on HRT
    - \( \text{UBD} = \text{NHRT} \times \text{Lbus} - 1 \)
Computing an UBD: Shared L2 Cache (P1)

- Two different types of inter-thread interferences
  - Bank access interference
  - Storage interference

- Bank access interference
  - Two requests want to access the same bank
  - New UBD is required

\[
\text{UBD} = N \cdot \text{HRT} \cdot \max(\text{LBANK}, \text{LBUS}) - 1
\]

where \( \text{LBANK} \) is the latency to access a L2 bank.
Storage interferences

- A thread evicts valid data of another thread

To solve **storage interferences** we use cache partitioning

- **Bankization**: Prevents bank access interferences
  
  \[ \text{UBD} = \text{NHRT} \times \text{Lbus} - 1 \]

- **Columnization**
  
  \[ \text{UBD} = \text{NHRT} \times \text{Lbank} - 1 \]
Agenda

- Intuitive Solutions to enable CMPs in RTESs
  - **Our solution**
    - Proposal 1: Multicore architecture guaranteeing an UBD
      - On-chip shared bus
      - L2 cache
    - Proposal 2: The WCET computation mode
  - Experimental results
  - Conclusions
The WCET execution mode (P2)

- Our architecture provides an UBD allowing to perform WCET analysis of HRTs running on multicore processor

- We introduce a WCET computation mode

  - Every time a request from the HRT is ready to access the bus, it is delayed by UBD cycles

  - Each analyzed HRT is run in isolation assuming the maximum inter-thread interference scenario
    - Single core WCET analysis tools can be used

  - It only requires to know the total number of HRTs that the analyzed task will run with
The WCET execution mode (P2)

- Hardware implementation:

  - **WCET execution mode** example:
    - \( \text{UBD} = N_{\text{HRT}} \cdot L_{\text{BANK}} - 1 \)
    - \( N_{\text{HRT}} = 3; L_{\text{BANK}} = 4; \text{NHRTs}= \text{YES} \)
    - \( \text{UBD} = 11 \)
Agenda

- Intuitive Solutions to enable CMPs in RTESs

- Our solution
  - Proposal 1: Multicore architecture enforcing an UBD
    - On-chip shared bus
    - L2 cache
  - Proposal 2: The WCET computation mode

- Experimental results

- Conclusions
Experimental Setup

- We use RapiTime, a commercial tool for WCET analysis with no single change

- As HRT benchmarks we use:
  - A real HRT application provided by Honeywell
    - 3D collision avoidance algorithm
  - EEMBC Automotive
    - We classify the EEMBC according to shared resources demanding into 3 groups: High, Medium, Low

- As NHRT benchmarks we use
  - MediaBench, MiBench, SPEC CPU 2006
Baseline: WCET for the HRT when it runs in isolation (full cache) and without inter-thread interferences (with WCET computation mode disabled)
Baseline: WCET for the HRT when it runs in isolation (full cache) and without inter-thread interferences (with WCET computation mode disabled)
We do NOT degrade performance to NHRTs

Baseline: the NHRTs running together without any HRTs

Workload: {H,M,susan,bzip2}

HRT tasks use 12 banks leaving 4 for susan-bzip2

The throughput of susan-bzip2 is 73% w.r.t. when they run alone {susan,bzip2}
Agenda

- Intuitive Solutions to enable CMPs in RTESs
- Our solution
  - Proposal 1: Multicore architecture enforcing an UBD
    - On-chip shared bus
    - L2 cache
  - Proposal 2: The WCET computation mode
- Experimental results
- Conclusions
Conclusions

- Our architecture guarantees that each request to a shared resource is upper bounded by UBD

- We introduce the WCET computation mode that considers the UBD in the WCET analysis

- Major contributions of our proposal are:
  - We can estimate a safe and tight WCET for HRTs running on mixed application workload
  - It is possible to change after the integration phase the threads without the need of re-analyze the whole system
  - Current single-core WCET analysis techniques can be used without modification
  - NHRTs can be executed together with HRTs
Thanks for the attention!
Backup Slides
Related Work

  - Similar hw mechanism for soft real-time systems

  - Solution limited to scalar pipeline with only one of the hw thread selected for the execution on the pipeline at the time

  - It is required to know the workload a priori, to define the scheduling
Experimental Setup

- Cycle-accurate execution driven simulator
  - Tricore-ISA compatible
  - 4-cores
    - 12-stages pipeline
    - No branch prediction
    - Fetch bandwidth: 8
    - Pre-issue bandwidth: 4
    - Private Instruction and Data L1: 8KB
  - Shared L2: 128KB
  - L1 miss and L2 hit: 9 cc