Multicore Resource Sharing

Prof. Dr. Jian-Jia Chen

LS 12, TU Dortmund

12, July, 2016
Motivations for Multicore Systems

- MPSoc or Multicore Systems
  - Performance
  - Flexibility
  - Cost
- Applications
  - Multimedia Devices (HDTV, SDR,..)
  - Avionics, automotive applications,
- Commercial, off-the-shelf (COTS)
  - cheap
  - reduced time-to-market
State-of-the-Art Multicore Systems

- Architecture with shared resource
  - shared memory, communication peripherals, I/O peripherals
- Blocking access to shared resource
  - one request at a time is served
  - stalling due to contention
- Possible approaches to reduce the contention
  - structure of tasks on the cores
  - arbitration policy on the shared resource (static/dynamic)
Predictability Due to Resource Sharing

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

- 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

L2 cache misses
Predictability Due to Resource Sharing

Access memory

Multi Core CPU 1

Core 1

L1 Cache

Core 2

L1 Cache

L2 Cache

Multi Core CPU 2

Core 1

L1 Cache

Core 2

L1 Cache

L2 Cache

Main Memory
Predictability Due to Resource Sharing

Task is executed on Core 2
Predictability Due to Resource Sharing

L1 cache misses

Multi Core CPU 1

- Core 1
  - L1 Cache
- Core 2
  - L1 Cache

Multi Core CPU 2

- Core 1
  - L1 Cache
- Core 2
  - L1 Cache

L2 Cache
Main Memory
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

Multi Core CPU 1

Multi Core CPU 2
Predictability Due to Resource Sharing

L1 cache misses
Predictability Due to Resource Sharing

L2 cache misses

Multi Core CPU 1

Multi Core CPU 2
Predictability Due to Resource Sharing

Memory access is blocked
Predictability Due to Resource Sharing

Memory access for core 1 finishes
Approach with Timed Automata

## A Very Simple Problem

### Input:

- **Structure of tasks on the core**
  - Tasks modeled as sequential / time-triggered superblocks
- **Arbitration policy on the shared resource**
  - Static arbitration (TDMA)
  - Dynamic arbitration (FCFS, RR)
  - Adaptive arbitration (TDMA + RR or FCFS)
- **The timing behavior of the architecture is predictable**
  - Resource access time is bounded and tight after granted
- **Only timing interference is considered**
  - Resource access time is bounded and tight after granted
A Very Simple Problem

Input:

- Structure of tasks on the core
  - Tasks modeled as sequential / time-triggered superblocks
- Arbitration policy on the shared resource
  - Static arbitration (TDMA)
  - Dynamic arbitration (FCFS, RR)
  - Adaptive arbitration (TDMA+ RR or FCFS)
- The timing behavior of the architecture is predictable
  - Resource access time is bounded and tight after granted
- Only timing interference is considered
  - Resource access time is bounded and tight after granted

Output: What is the worst-case response time?
How to determine the schedulability due to timing interference?
References

Worse-Case Response Time

- Depends on the resource access model
  - Uncertainty, when resource accesses happen

- Static arbitration (TDMA)
  - Cores only interfere with the TDMA arbiter

- Dynamic arbitration (FCFS, RR)
  - What is the worst-case?
    (non-compositionality)
  - Approximation of interference

- Adaptive arbitration (TDMA + FCFS, RR)
  - Approximation of interference

[DAC 2010]
3

[RTAS 2010]
2

[DATE 2010]
1

[RTAS 2011]
4
Worse-Case Response Time

- Depends on the resource access model
  - Uncertainty, when resource accesses happen

- Static arbitration (TDMA)
  - Cores only interfere with the TDMA arbiter

- Dynamic arbitration (FCFS, RR)
  - What is the worst-case?
    (non-compositionality)
  - Approximation of interference

- Adaptive arbitration (TDMA + FCFS, RR)
  - Approximation of interference

[DAC 2010] 3

[RTAS 2010] 2

[DATE 2010] 1

[RTAS 2011] 4
Task Model

- A sequence of subsequent super-blocks
- Each super-block \( j \) of task \( i \) is defined by
  - upper \( e_{i,j}^{\text{max}} \) / lower \( e_{i,j}^{\text{min}} \) execution time
  - upper \( \mu_{i,j}^{\text{max}} \) / lower accesses \( \mu_{i,j}^{\text{min}} \) to a shared resource
  - static analysis for two phases (1) assuming resource access requires no additional time to get the execution time \( e_{i,j} \) (2) only considering the number of shared resource accesses.
  - communication delay \( c \) depends on resource
- Executing periodically
- Access to shared resources can happen anytime
Task Model (cont.)

- Multiple tasks execute in a time wheel
- Periodic sequence of statically scheduled tasks
  - gap $g$ between two tasks is variable

![Diagram of task model with time wheel and task execution intervals]
Deriving Interfering (Arrival) Curves

- find all well defined time-windows and count the number of events that can happen

\[ \hat{i} = \left< \mu_{1,1}^{\text{max}}, 0 \right> \]

\[ \hat{i} = \left< \mu_{4,1}^{\text{max}} + \mu_{4,2}^{\text{max}}, \mu_{4,3}^{\text{max}} C \right> \]

\[ \hat{i} = \left< \sum_{k=1}^{4} \mu_{k,1}^{\text{max}} + \sum_{k=1}^{3} \mu_{k,1}^{\text{max}} C + \text{exec}_{2,1}^{\text{min}} \right> \]

\[ \hat{i} = \left< \sum_{k=1}^{7} \mu_{k,1}^{\text{max}} + \sum_{k=1}^{6} \mu_{k,1}^{\text{max}} C + \text{exec}_{2,1}^{\text{min}} + \text{exec}_{5,1}^{\text{min}} + g \right> \]

maximize/minimize to compute the gap

- superblocks considered for gap computation
- relevant time window
- superblocks considered for time window computation
Resulting Interfering Curves

- Interference of each processing element
- $\tilde{\alpha}_i^l(\Delta)$ is the interference of PE $i$ onto resource $l$ by assuming PE $i$ executes in isolation

\[
\hat{\alpha}_i(\Delta) = \begin{cases} 
\tilde{\alpha}_i(\Delta) & 0 \leq \Delta \leq p \\
\max \left\{ \tilde{\alpha}_i(\Delta), \tilde{\alpha}_i(\Delta - p_i) + \sum_{j} (\mu_{i,j}^{\text{max}}) \right\} & p_i \leq \Delta \leq 2p \\
\tilde{\alpha}_i(\Delta - k \cdot p_i) + k \sum_{j} (\mu_{i,j}^{\text{max}}) & \text{otherwise}
\end{cases}
\]
Dynamic Arbitration

- Interference from other elements as arrival curve
- Access profile from superblock under analysis
Analysis for Dynamic Arbitration

Use Dynamic Programming:

\[ U_b_{1,1} = \{ D_{1,1} \} \text{ max delay for miss 1} \]

\[ U_b_{1,2} = U_b_{1,1} + D_{1,2} \text{ max delay for misses 1+2} \]

\[ U_b_{1,3} = U_b_{1,1} + D_{2,2} \text{ max delay for 1+2+3} \]

\[ U_b_{1,4} = U_b_{1,1} + D_{3,2} \]

\[ U_b_{1,5} + D_{3,2} \]
Worse-Case Response Time

- Depends on the resource access model
  - Uncertainty, when resource accesses happen

- Static arbitration (TDMA)
  - Cores only interfere with the TDMA arbiter

- Dynamic arbitration (FCFS, RR)
  - What is the worst-case? (non-compositionality)
  - Approximation of interference

- Adaptive arbitration (TDMA + FCFS, RR)
  - Approximation of interference

[DAC 2010] 3
[RTAS 2010] 2
[DATE 2010] 1
[RTAS 2011] 4
TDMA on the Shared Resource

- Independence between tasks
- Single source of interference
Superblocks with Phases

Tasks are structured as sequences of superblocks

- fixed order of execution
- upper bound on execution and communication
- phases may be present (acquisition/execution/replication)
Access Models

**General Model**

- $S_1$ $W_1$ $S_1$ $W_1$
- $s_{1,1}$ $s_{2,1}$ $s_{3,1}$ $s_{1,1}$
- $\ell_{1,1}$ $\ell_{2,1}$ $\ell_{3,1}$ $\ell_{1,1}$
- $\rho_{1,1}$ $\rho_{1,1} + W_1$

**Dedicated Model**

- $A$ $E$ $R$ $A$ $E$ $R$ $A$ $E$ $R$ $A$ $E$
- $\rho_{1,1}$ $\rho_{1,1} + W_1$

**Hybrid Model**

- $\rho_{1,1}$ $\rho_{1,1} + W_1$
Experiments

• sequences of superblocks
  • 82 superblocks
  • 8 superblocks

• different access models, but
  • equal amount of computation
  • equal amount of accesses to the shared resource

• TDMA arbiters
  • irregular arbiter (randomly placed slots)
Regular TDMA

Different access models, regular TDMA arbiter

82 superblocks
8 superblocks
deadline

Worst Case Response Time [ms]

no Interference (WCET)  GSS  HSS  DSS
Irregular TDMA

Different access models, irregular TDMA arbiter

- 82 superblocks
- 8 superblocks
- deadline

Worst Case Response Time [ms]

no Interference (WCET)
GSS
HSS
DSS
Resource Access Models (Revisit)

- **DS** dedicated sequential phases, sequential superblocks
- **HS** hybrid sequential phases, sequential superblocks
- **HTS** hybrid sequential phases, time-triggered superblocks
- **HTT** hybrid time-triggered phases, time-triggered superblocks
- **GS** general sequential phases, sequential superblocks
- **GTS** general sequential phases, time-triggered superblocks
Resource Access Models (Revisit)

- GTS
- HTS
- HTT
- GS
- HS
- DS

Schedulability relation

Time-triggered
Sequential

Prof. Dr. Jian-Jia Chen (LS 12, TU Dortmund)
Experimental Results for 3 sets of superblocks and different access models

- 125 superblocks
- 84 superblocks
- 9 superblocks

Resource Access Model

Worst-Case Response Time (WCRT) [ms]
Adaptive Arbitration

PU1 PU2 PU3

superblock $s_{1,1}$ superblock $s_{1,2}$ superblock $s_{1,3}$

$\mu_{1,1}^{\text{max, } a}$ $\mu_{1,1}^{\text{max, } r}$ $\mu_{1,2}^{\text{max}}$ $\mu_{1,3}^{\text{max, } a}$ $\mu_{1,3}^{\text{max, } e}$ $\mu_{1,3}^{\text{max, } r}$

$\text{exec}_{1,1}^{\text{max}}$ $\text{exec}_{1,2}^{\text{max}}$ $\text{exec}_{1,3}^{\text{max}}$

$\ell$

$M_\ell = 3$ $
\theta_{1,1} = 1$ $\theta_{2,2} = 1$ $\theta_{3,3} = 1$ $\theta_{4,1} = 1$

$\delta_3$

$\sigma_1$ $\sigma_2$ $\sigma_3$ $\sigma_4$ $\sigma_5$ $\Delta_\sigma$

static dynamic

shared resource

minislot

$\text{A } \text{E } \text{R } \text{A/E/R } \text{A } \text{A/E/R } \text{R } ...$

$t$

Use a much more complicated dynamic programming...
Adaptive Arbitration

- Use a much more complicated dynamic programming
Remarks

- Resource sharing in multicore systems is an important issue in terms of
  - Predictability
  - Efficiency
- Modern microarchitectures are optimized for average-case performance at the expense of timing predictability and repeatability.
- Analytical solutions are quite complicated now even for very simple cases
- The area is still a very active and attractive domain to be explored in the following years
Related Approaches: PRET (Precision-Time Machines)

Precise and repeatable timing behavior for an instruction (from UC Berkeley and University of Columbia)

Round-robin thread scheduling

Scratchpad memories

Thread-interleaved pipeline

Time-triggered main memory access
EU Project: MERASA (2nd phase is running)

- Make the analysis of each task independent from the coscheduled tasks by enforcing hardware features
- Design a multicore architecture that allows safe and tight WCET estimations
EU Project: PREDATOR (2008-2011)

- Analyze the timing behavior of the system
- Try to improve the efficiency while maintaining the timing predictability
- Design a multicore architecture that allows safe and tight WCET estimations
- Joint considerations that cover
  - Architecture
  - Compiler
  - Code Analysis
  - Scheduling
  - Software Tool
  - Software Integration
Really That Simple?

No! If you consider some architectures we have now

\[ p_1 \]

\[ p_2 \]

Crossbar Bus

FIFO

Instruction
Flash

Shared
Memory

Prof. Dr. Jian-Jia Chen (LS 12, TU Dortmund)