Multicore Resource Management for Embedded Real-Time Systems

Heechul Yun
University of Kansas
High-Performance Multicores for Embedded Real-Time Systems

• Why?
  – Intelligence → more performance
  – Space, weight, power (SWaP), cost
Time Predictability Challenge

- Hardware resources are shared among the cores
- Tasks can suffer significant **interference delays**
  - unpredictable, non-deterministic $\Rightarrow$ non-certifiable, unsafe
Example: Real-Time Obstacle Detection and Avoidance

- Co-runners were launched on idle CPU cores
- 5X slowdown in detection speed (≈10fps → 2fps)
  - can fail to avoid obstacle
    - e.g., 10m/s aircraft (MAV) can move 1m in 100ms
Research Mission

• Our research goal is to build **predictable**, **efficient**, and **safe** computing infrastructure for the next generation of intelligent embedded real-time systems, a.k.a., Cyber Physical Systems (CPS).

• Approach
  – Develop software/hardware mechanisms
  – Develop analysis framework
Research Results

Certifiable Multicore Architecture

- **Core1**
- **PMC**
- **Shared DRAM**
- **High Performance Real-Time Memory Controller**

**Operating System**

- **RT CPU Application**
- **NonRT CPU Application**
- **RT GPU Application**
- **NonRT GPU Application**
- **B/W mgmt. API**
- **QoS mgmt. API**
- **System Library**

**Certifiable Multicore Architecture**

- **Core2**
- **PMC**
- **Core3**
- **PMC**
- **Core4**
- **PMC**
- **Accelerators (GPU, FPGA)**

**Certifiable Multicore Architecture**

- **Medusa**
  - **CPSNA’15**
- **Memory delay analysis**
  - **ECRTS’15, RTAS’16**

**MemGuard**

- **TC’15, RTAS’13, ECRTS’12**

**PALLOC**

- **RTAS’14**

**Shared cache, MSHR**

- **OSPERT’15, RTAS’16**

**BWLOCK**

- **In submission**

**Research Results**

- **UAV simplex**
  - **RTCSA’16**

**BWLOCK**

- **In submission**

**MemGuard**

- **TC’15, RTAS’13, ECRTS’12**

**PALLOC**

- **RTAS’14**

**Shared cache, MSHR**

- **OSPERT’15, RTAS’16**

**Medusa**

- **CPSNA’15**

**Memory delay analysis**

- **ECRTS’15, RTAS’16**
PALLOC

- DRAM bank-aware kernel memory allocator
- Can void bank conflict

Improved Isolation

- **Private banking**
  - *Allocate* pages on certain *exclusively* assigned banks

- **Better Performance Isolation**
Real-Time Performance

- Setup: HRT $\rightarrow$ Core0, X-server $\rightarrow$ Core1
- Buddy: no bank control (use all Bank 0-15)
- Diffbank: Core0 $\rightarrow$ Bank0-7, Core1 $\rightarrow$ Bank8-15

Buddy(solo)  Buddy  PALLOC(diffbank)
MemGuard

- Goal: guarantee *minimum memory b/w* for each core
- How: b/w reservation + best effort sharing

Impact of Reservation

**W/o MemGuard**

![Graph showing LLC misses/ms over time for W/o MemGuard.]

**MemGuard (1GB/s)**

![Graph showing LLC misses/ms over time for MemGuard (1GB/s).]
Evaluation Results

**Shared Memory**

Intel Core2

- C0: 1GB/s
- C2: 0.2GB/s

462.Libquantum (foreground)

memory hogs (background)

Normalized IPC

- Foreground (462.libquantum)

<table>
<thead>
<tr>
<th></th>
<th>run-alone</th>
<th>co-run</th>
<th>run-alone</th>
<th>co-run</th>
<th>run-alone</th>
<th>co-run</th>
</tr>
</thead>
<tbody>
<tr>
<td>w/o Memguard</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Memguard</td>
<td></td>
<td></td>
<td>Memguard</td>
<td></td>
<td>Memguard</td>
<td></td>
</tr>
<tr>
<td>(reservation only)</td>
<td></td>
<td></td>
<td>(reclaim+share)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Guaranteed performance

Reclaiming and Sharing maximize performance
OS Controlled MSHR Partitioning

• Cache partitioning != cache performance isolation
  – Due to contention in Miss Status Holding Registers (MSHRs)
• Architecture and OS collaborative solution
  – Small changes (add two registers) in hardware
  – Small changes in OS scheduler

Case Study

- 4 RT tasks (EEMBC)
  - One RT per core
  - Reserve 2 MSHRs
  - P: 20, 30, 40, 60ms
  - C: ~8ms

- 4 NRT tasks
  - One NRT per core
  - Run on slack

- Up to 20% WCET reduction
  - Compared to cache partition only
References


• [RTAS16-1] Prathap Kumar Valsan, Heechul Yun, Farzad Farshchi. Taming Non-blocking Caches to Improve Isolation in Multicore Real-Time Systems. In IEEE Intl. Conference on Real-Time and Embedded Technology and Applications Symposium (RTAS), 2016 (Best Paper Award.)


On-going Projects

• Multicore Resource Management
  – OS, architecture research for time predictability
  – Funding Agencies: NSF, ETRI

• UAV Software Platform
  – ROS (Robot Operating System) based autopilot and real-time sensor (radar and vision) processing
  – Funding Agencies: NASA
Autonomous Racing

F1/10

Dr. Madhur Behl
University of Pennsylvania

http://www.f1tenth.org/
Prospective Students

• Solid background in operating systems and computer architecture
• Good system programming skills
• Interests and experiences in building Intelligent cyber-physical systems
  – ROS, python, Linux, OpenCV, CUDA
  – PID control, real-time sensor fusion

• Send me your CV and schedule a meeting