Real-Time Multi/Many-Core Architecture

Heechul Yun
Real-Time Multi/Many-Core Architecture

• Projects on Real-Time CPU Architectures

• Assigned Papers
  – Shedding the Shackles of Time-Division Multiplexing, RTSS, 2018
  – Deterministic Memory Abstraction and Supporting Multicore System Architecture. ECRTS, 2018
Trends in Automotive E/E Systems


Centralization & High-Performance HW
Modern System-on-a-Chip (SoC)

- Integrate multiple cores, GPU, accelerators
- Good performance, size, weight, power
- Challenges: time predictability
Worst-Case Execution Time (WCET)

- Real-time scheduling theory is based on the assumption of known WCETs of real-time tasks.

Image source: [Wilhelm et al., 2008]
Computing WCET

• Static analysis
  – Input: program code, architecture model
  – output: WCET
  – Problem: architecture model is hard and pessimistic

• Measurement
  – No guarantee on true worst-case
  – But, widely used in practice
Memory Hierarchies, Pipelines, and Buses for Future Architectures in Time-Critical Embedded Systems

IEEE TCAD, 2009
“Problematic” CPU Features

• Architectures are optimized to reduce average performance
• WCET estimation is hard because of
  – Pipelining
  – TLBs/Caches
  – Super-scalar
  – Out-of-order scheduling
  – Branch predictors
  – Hardware prefetchers
  – Basically anything that affect processor state
Static Timing Analysis

- Binary Executable
- CFG Reconstruction
- Control-flow Graph

Legend:
- Data
- Phase

- Value Analysis
- Loop Bound Analysis
- Control-flow Analysis

- Annotated CFG

- Micro-architectural Analysis
- Basic Block Timing Info
- Global Bound Analysis
Control Flow Graph (CFG)

- Analyze code
- Split basic blocks
- Compute per-block WCET
  - use abstract CPU model
Timing Anomalies

• Locally faster != globally faster

Branch Condition Evaluated

Cache Hit

A → Prefetch

C - Miss due to Prefetch

Cache Miss

A

C

Image source: [Wilhelm et al., 2008]
Timing Anomalies

- Locally faster != globally faster

Image source: [Wilhelm et al., 2008]
Challenge: Shared Memory Hierarchy

- **Memory performance** varies widely due to interference
- **Task WCET** can be **extremely pessimistic**
Effect of Memory Interference

- DNN control task suffers >10X slowdown
  - When co-scheduling different tasks on idle cores.

Cache Denial-of-Service Attacks

- Observed worst-case: >300X (times) slowdown
- On simple in-order multicores (Raspberry Pi3, Odroid C2)

Difficult to guarantee predictable timing

Real-Time CPU Architectures

- PRET
  - UC Berkeley.
- MERASA/parMERASA project
  - EU
- ACROSS
  - EU
- ARAMIS
  - Germany
- EMC2
  - EU
FlexPRET: A Processor Platform for Mixed-Criticality Systems

RTAS, 2014
FlexPRET
Hard-Real-Time (HRT) Threads
Interleaved with Soft-Real-Time (SRT) Threads

HRT threads have deterministic timing. SRT threads share available cycles

SRAM scratchpad shared among threads

DRAM main memory provides deterministic latency for HRT threads. Conventional behavior for the rest.

Michael Zimmer
FlexPRET Pipeline

- 32-bit, 5-stage thread interleaved pipeline, RISC-V ISA
  - **Hard real-time HW threads**: scheduled at constant rate for isolation and predictability
  - **Soft real-time HW threads**: share all available cycles (e.g. HW thread sleeping) for efficiency
- Deployed on Xilinx FPGA (area comparable to Microblaze)
Hardware Support for WCET Analysis of Hard Real-Time Multicore Systems

ISCA 2009
Analyzable Multicore Architecture

• Idea1: Bound interference on shared resources
  – On-chip shared bus
  – (shared) L2 cache

• Idea2: WCET computation mode
Architecture
Round-Robin Bus Arbitration

- $\text{UBD} = (\text{NHRT} - 1) \times \text{Lbus}$
Request vs. Job-level WCET Analysis

- Request-level analysis
  - Assume worst-case interference for each access of the task under analysis
  - Pessimistic as not all accesses will get interference

- Job-level analysis
  - Assume the total number of competing memory access is known
  - Can reduce pessimism
Summary

• Timing anomalies
  – Locally fast != globally fast on non-timing compositional architectures (i.e., most architectures)

• Timing compositional architecture
  – Free of timing anomalies
Discussion

• Why is this interesting?

• Are assumptions realistic?
  – Task model
  – Cache model
  – Memory model
  – CPU (pipeline) model
Discussion

• Why is this interesting?

• Are assumptions realistic?
  – Task model
  – Cache model
  – Memory model
  – CPU (pipeline) model
Atomic vs. Split-Transaction Bus

(a) Simple bus with atomic transactions

(b) Split-transaction bus with separate requests and responses

A split-transaction bus enables higher throughput by pipelining requests, responses, and data transmission.

Figure 11.9
Simple Versus Split-Transaction Busses.
Announcement

• Mini Project #1

• DeepPicar Competition
  – Build a self-driving car
  – Based on DeepPicar
  – Competition format
Acknowledgement

• Some slides are from:
  – Prof. Rodolfo Pellizzoni, University of Waterloo
  – Prof. Edward A. Lee, University of Berkeley