Increasing the Predictability of Modern COTS Hardware through Cache-Aware OS-Design

11th Workshop on Operating Systems Platforms for Embedded Real-Time Applications

Hendrik Borghorst

hendrik.borghorst@udo.edu
https://ess.cs.tu-dortmund.de/~hb

Embedded System Software Group
Computer Science 12, TU Dortmund
Cyber-physical systems

**Situation today:**
- Specialized systems are widespread
- Tight time bounds critical

**Future:**
- Multiple small real-time task-sets on one system preferable
Cyber-physical systems

**Situation today:**
- Specialized systems are widespread
- Tight time bounds critical

**Future:**
- Multiple small real-time task-sets on one system preferable
- Why is off-the-shelf hardware not used for such systems?
Random data access

access time in cycles

access number

Random memory access within a 64kB range
Random data access

![Graph showing L1 and L2 hits with cache misses.]
Random data access

Unstable execution times

Random memory access within a 64kB range

Cache misses

L1 hits

L2 hits
Causes for unpredictability

- DRAM\(^1\)
  - Unstable access latency
- Shared buses between multiple cores
  \(\rightarrow\) Overall system response time unstable

Causes for unpredictability

- DRAM\(^1\)
  - Unstable access latency
- Shared buses between multiple cores
  → Overall system response time unstable

**Approach:**

Causes for unpredictability

- DRAM\(^1\)
  - Unstable access latency
- Shared buses between multiple cores
  → Overall system response time unstable

**Approach:**

- Caches can reduce unpredictability

---

Causes for unpredictability

- DRAM\(^1\)
  - Unstable access latency
- Shared buses between multiple cores
  \(\rightarrow\) Overall system response time unstable

**Approach:**

- Caches can reduce unpredictability
- Can the OS control which data stays in the cache?

Causes for unpredictability

- DRAM\(^1\)
  - Unstable access latency
- Shared buses between multiple cores
  \(\rightarrow\) Overall system response time unstable

Approach:
- Caches can reduce unpredictability
- Can the OS control which data stays in the cache?

\(\rightarrow\) No unexpected cache misses

OS-controlled cache

Random memory access within a 64kB range (prefetched/locked)
OS-controlled cache

But how can the whole system fit in the cache?

Random memory access within a 64kB range (prefetched/locked)
OS-controlled cache

But how can the whole system fit in the cache? It doesn't have to

Random memory access within a 64kB range (prefetched/locked)
OS-Model: Component

- Small components
- Mostly independent
- No external calls/data accesses (cache misses)
- All necessary data confined
OS-Model: Interaction

- OS-Init
  - start_Scheduler
  - invoke_Scheduler

- Scheduler
  - schedule_Task2
  - schedule_Task1

- Task1
  - Task2
  - UART-Buffer
  - Ethernet-Driver
  - Timer-Driver

- Interrupt-Handler
  - Timer-Interrupt
  - Ethernet-Interrupt
  - UART-Interrupt

- Operating system component
- Shared data
- Critical OSC

- Shared data connection
- Event-trigger connection
- Event
- Trigger
OS-Model: Interaction

Components connect via events

- Interrupt Handler
- Timer-Driver
- Ethernet-Driver
- UART-Driver
- Socket
- UART-Buffer
- Task1
- Task2
- Scheduler
- OS-Init

- event
- trigger
- shared data
- critical OSC

- operating system component
- shared data
- event-trigger connection
OS-Model: Interaction

Components connect via events

Interrupt

Timer-Handler

Timer-Interrupt

Ethernet-Interrupt

UART-Interrupt

Timer-Driver

invoke_Scheduler

Scheduler

start_Scheduler

Task2

schedule_Task2

Task1

schedule_Task1

Shared data for multiple components

interrupt

shared data

trigger

OS-Init

Event

Shared data

for multiple components

OS-Model: Interaction

operating system component

shared data

critical OSC

shared data

event-trigger connection

OS-Model: Interaction

Components connect via events

Interrupt

Timer-Handler

Timer-Interrupt

Ethernet-Interrupt

UART-Interrupt

Timer-Driver

invoke_Scheduler

Scheduler

start_Scheduler

Task2

schedule_Task2

Task1

schedule_Task1

Shared data for multiple components

interrupt

shared data

trigger

OS-Model: Interaction

Components connect via events

Interrupt

Timer-Handler

Timer-Interrupt

Ethernet-Interrupt

UART-Interrupt

Timer-Driver

invoke_Scheduler

Scheduler

start_Scheduler

Task2

schedule_Task2

Task1

schedule_Task1

Shared data for multiple components

interrupt

shared data

trigger

OS-Model: Interaction

Components connect via events

Interrupt

Timer-Handler

Timer-Interrupt

Ethernet-Interrupt

UART-Interrupt

Timer-Driver

invoke_Scheduler

Scheduler

start_Scheduler

Task2

schedule_Task2

Task1

schedule_Task1

Shared data for multiple components

interrupt

shared data

trigger

OS-Model: Interaction

Components connect via events

Interrupt

Timer-Handler

Timer-Interrupt

Ethernet-Interrupt

UART-Interrupt

Timer-Driver

invoke_Scheduler

Scheduler

start_Scheduler

Task2

schedule_Task2

Task1

schedule_Task1

Shared data for multiple components

interrupt

shared data

trigger

OS-Model: Interaction

Components connect via events

Interrupt

Timer-Handler

Timer-Interrupt

Ethernet-Interrupt

UART-Interrupt

Timer-Driver

invoke_Scheduler

Scheduler

start_Scheduler

Task2

schedule_Task2

Task1

schedule_Task1

Shared data for multiple components

interrupt

shared data

trigger

OS-Model: Interaction

Components connect via events

Interrupt

Timer-Handler

Timer-Interrupt

Ethernet-Interrupt

UART-Interrupt

Timer-Driver

invoke_Scheduler

Scheduler

start_Scheduler

Task2

schedule_Task2

Task1

schedule_Task1

Shared data for multiple components

interrupt

shared data

trigger

OS-Model: Interaction

Components connect via events

Interrupt

Timer-Handler

Timer-Interrupt

Ethernet-Interrupt

UART-Interrupt

Timer-Driver

invoke_Scheduler

Scheduler

start_Scheduler

Task2

schedule_Task2

Task1

schedule_Task1

Shared data for multiple components

interrupt

shared data

trigger

OS-Model: Interaction

Components connect via events

Interrupt

Timer-Handler

Timer-Interrupt

Ethernet-Interrupt

UART-Interrupt

Timer-Driver

invoke_Scheduler

Scheduler

start_Scheduler

Task2

schedule_Task2

Task1

schedule_Task1

Shared data for multiple components

interrupt

shared data

trigger
OS-Model: Interaction

Components connect via events

Not predictable

Operating system component

shared data

critical OSC

trigger

event

shared data

connect via events

Timer-Interrupt

Ethernet-Interrupt

UART-Interrupt

Interrupt-Handler

Timer-Driver

Ethernet-Driver

UART-Driver

Scheduler

Task1

Task2

Socket

UART-Buffer

Components for multiple components
OS-Model: Interaction

Components connect via events

Not predictable

How does this design help?

- Components connect via events
- Shared data for multiple components

Operating system component

Shared data

Critical OSC

Timer-Handler

Timer-Driver

Ethernet-Interrupt

Ethernet-Driver

Ethernet-Interrupt

Interrupt-Handler

UART-Interrupt

UART-Driver

UART-Buffer

Socket

Task1

Task2

invoke_Scheduler

start_Scheduler

schedule_Task1

schedule_Task2

start_Scheduler

Notes:
- Not predictable
- Components connect via events
- Shared data for multiple components

Legend:
- Operating system component
- Shared data
- Critical OSC
- Shared data
- Event-trigger connection
- Event
- Trigger

How does this design help?

- Components connect via events
- Shared data for multiple components
- Not predictable
Component handling

- All data confined to one continuous data block
- Enables complete knowledge over necessary data
- Components can be prefetched in one bulk transfer
  - Bulk transfers evaluated → stable execution times

![Graph showing cycles per byte vs prefetch size](image)
Component handling

- All data confined to one continuous data block
- Enables complete knowledge over necessary data
- Components can be prefetched in one bulk transfer
  - Bulk transfers evaluated → stable execution times

Each dot is one bulk transfer
Component handling

- All data confined to one continuous data block
- Enables complete knowledge over necessary data
- Components can be prefetched in one bulk transfer
  - Bulk transfers evaluated → stable execution times

All components needed for execution are now loaded & locked to the cache

Each dot is one bulk transfer
OS-Transition

L2-Cache

Core 0

Ethernet-Driver

Task1

Interrupt-Handler

Core 1

UART-Driver

UART-Buffer

Cache-Management

predicted random access times

unpredictable random access times

locked permanently

loads & locks the cache

Main Memory

Task2

Scheduler

OS-Init

Timer-Driver

Socket

operating system component

shared data

critical OSC
OS-Transition

- **Core 0**
  - Ethernet-Driver
  - Task1
  - Interrupt-Handler

- **Core 1**
  - UART-Driver
  - Task2

**L2-Cache**

- Locked permanently
- Osc-transition
- Predictable random access times
- Unpredictable random access times

**Main Memory**

- Task2
- Scheduler
- OS-Init
- Timer-Driver
- Socket

- Operating system component
- Shared data
- Critical OSC

**Cache-Management**

- Loads & locks the cache
OS-Transition

L2-Cache

locked permanently
predictable random access times
unpredictable random access times

Main Memory

operating system component
shared data
critical OSC
Architecture details

- Current approach needs HW support for
  - Cache prefetching
  - Cache locking
- Current platform: **Dual-core ARM Cortex-A9 (COTS)**
- Associative shared level cache
  - 16 cache ways (64kB each, 1MB total)
- Cache management features:
  - Cache **prefetching** of data/code
  - Cache **locking** per cache way & core
Architecture details

- Current approach needs HW support for
  - Cache prefetching
  - Cache locking
- Current platform: **Dual-core ARM Cortex-A9** (COTS)
- Associative shared level cache
  - 16 cache ways (**64kB** each, **1MB** total)
- Cache management features:
  - Cache *prefetching* of data/code
  - Cache *locking* per cache way & core

**HW allows complete control over the cache content**
Cache Management

Components aligned at way-size

way_0  way_1  way_2  ...  way_n

Int-Handler
Cache-Manager

Permanently locked
Temporarily locked
Temporarily unlocked
Cache Management

Components aligned at way-size

Int-Handler

Cache-Manager

way_0 way_1 way_2 \ldots way_n

unlock cache way & prefetch OSC

Permanently locked

Temporarily locked

Temporarily unlocked
Cache Management

Components aligned at way-size

way\(_0\)  way\(_1\)  way\(_2\)  \ldots  way\(_n\)

Int-Handler
Cache-Manager

unlock cache way & prefetch OSC

way\(_0\)  way\(_1\)  way\(_2\)  \ldots  way\(_n\)

Int-Handler
Cache-Manager

Task 1

Permanently locked

Temporarily locked

Temporarily unlocked
Cache Management

Components aligned at way-size

Int-Handler
Cache-Manager

way₀  way₁  way₂  \ldots  wayₙ

unlock cache way & prefetch OSC

Int-Handler
Cache-Manager

Task 1

way₀  way₁  way₂  \ldots  wayₙ

lock cache

Permanently locked  Temporarily locked  Temporarily unlocked
Cache Management

Components aligned at way-size

Int-Handler
Cache-Manager

way_0  way_1  way_2  ...  way_n

unlock cache way & prefetch OSC

Int-Handler
Cache-Manager

way_0  way_1  way_2  ...  way_n

lock cache

Int-Handler
Cache-Manager

way_0  way_1  way_2  ...  way_n

Permanently locked

Temporarily locked

Temporarily unlocked
Ongoing & Future Work

- Optimize event scheduling
- Automatic adaptation to HW platform
- Eliminate dead code/data prefetching
- Reduce the dependency on hardware cache management
- Compare against other RTOS
Summary

- Modern COTS-HW unpredictable (DRAM, buses, ...)
- Caches hide DRAM-access latency
- Small structured OS proposed
- Components fit in cache
  → Shift random DRAM-access to bulk transfer
  → Predictable access times
Summary

- Modern COTS-HW unpredictable (DRAM, buses, ...)
- Caches hide DRAM-access latency
- Small structured OS proposed
- Components fit in cache
  → Shift random DRAM-access to bulk transfer
  → Predictable access times

Questions?