HARDWARE IN REAL-TIME SYSTEMS

HERMANN HÄRTIG, MARCUS VÖLP
How to create a precise timestamp for the event?
Hardware-sources of unpredictability
- Interrupt latency
- Memory subsystem
- Pipelining
- System Management Mode
- Special-purpose hardware
- Taming unpredictable hardware: Cache Partitioning
Sources of Delay

- disabled/enabled interrupts
- await instruction boundary
- kernel entry
- select handler code in OS
- cache state

Some are not easily predictable.
SYSTEM MANAGEMENT MODE

- On PC/x86 platforms
- Underneath operating system and virtualization layers
- Invoked using non-maskable interrupt
- Used for platform specific management
  - correction of design errors
  - thermal management: fan control
- Can be switched off, but better do not try
- Adds unpredictable delays
Special Purpose Hardware for Embedded Real-Time

- Microcontrollers, ARM Cortex-R
- Low-latency interrupt mode
- Interrupt handlers consisting of a single instruction
- Capture/Compare units
- Scratchpad memories
- Real-time clocks
Problem

- precise time stamp of event
- trigger event at precise time
- software handler: jitter too high
Objective: zero overhead for one instruction per signal

Peripheral Event Controller (PEC)

- upon a signal, a PEC channel performs a single transfer between any two memory locations
- counter decremented on signal, interrupt when zero
- optional increment for address
PEC semantic as pseudo-code:

```c
signal () {
    if (counter--)
        *dest++ = *src;  // minimal response time
        // PEC can execute at external clock rate
    else
        trigger_interrupt();
}
```

Example: Peripheral Event Controller SAB 80C166
LOW-LATENCY INTERRUPTS

- Additional Interrupt Mode, e.g. ARM IRQ + FIQ
  - FIQ interrupts IRQ
  - no need to save registers in FIQ handler, 5 registers immediately available
- Worst case FIQ (PL190 VIC, if some rules are followed):
  - Interrupt synch: 3 cycles
  - Worst case for current instruction: 7 cycles
  - Entry of first instruction: 2 cycles
MEMORY SUBSYSTEM

Virtual Addr. -> TLB -> Page tables -> Physical Addr.

core -> L1 Cache -> L2 Cache -> L3 Cache -> Interconnect

CPU/Devices -> Memory C. -> DRAM

TU Dresden Hardware 11
SCRATCHPAD MEMORY

core

(S)RAM

TU Dresden Hardware 12
THE PROBLEM

- competing accesses to addresses that share cache lines
- especially critical: sharing across address spaces
CACHE PARTITIONING

Goal
- partition cache to reduce interference between RT/RT or RT/non-RT applications

General method
- address regions map to cache sets
- control allocation to address regions

Approaches
- compiler / linker / operating system / hardware
HOW CACHES WORK

cache line
TRUE: use cache line
FALSE: write out and reload cache line
within-cache line address
MULTI-WAY CACHES

Address

index

tag

tag

tag

tag

tag

tag

tag

tag

tag

tag

tag

tag

tag

tag

tag

tag

tag

tag

tag

tag

tag

tag

tag

tag

tag

tag

tag

tag

tag

tag

tag

tag

tag

tag

tag

tag

tag

tag

= 

= 

= 

= 

= 

Hardware

way 0

way 1

way 2

way n-1
Hardware Support

- Locking of entire ways or assigning ways to cores
- Advantages:
  - transparent to applications
  - relatively simple for the operating system
- Disadvantages:
  - coarse granularity
  - there may be not enough ways
CACHE COLORING

- use addresses with same index-value for isolation
- assign indices to code/data pieces
- manually by programmer, or compiler / operating system
- can we do it transparently?
simple example:
2 way cache
16 bit address machine
16 Byte cache lines,
4 bit index, 8 bit tag

disadvantages: “gaps”+“waste”:
index C:
00C0..F
01C0..F
02C0..F
...
FFC0..F
- gaps:
  address space is fragmented into many small pieces
  compiler/programmer must make sure that only these pieces are used
  not useful for legacy applications

- waste:
  these pieces can be used by one code/data piece

- we remove these limitation in two steps
MEMORY SUBSYSTEM

- Virtual Addr.
- Physical Addr.
  - TLB
  - Page tables

- L1 Cache
- L2 Cache
- L3 Cache

- Interconnect
- Memory C.
- DRAM

- CPU/Devices
- Core

VIRTUAL OR PHYSICAL INDEX

virtual caches:
- slightly faster
- Problems with aliasing

most architectures:
- physical tags
- L1: virtual indices
  L2/L3: physical indices
PAGE COLORING

- physical indices required
- Mapping of VPN to PPN is under control of OS
- PPN.index under control of OS
- the index part in the PPN defines a set of cache lines referred to as colors
- OS can control cache usage by assigning colors to address spaces
- no gaps in virtual address space
- Siemens SAB 80C166 Handbook
- More on (results in) Cache Partitioning: Liedtke, Härtig, Hohmuth (RTAS '97): Operating system control cache predictability for real-time applications