Embedded Computing Systems in the Multi-Core Era

Wolfgang Schröder-Preikschat
Background

- multi/many-core systems (since 2008)
- embedded & real-time systems (since 1995)
- multi-/many-core systems (1981-1986)
- uni-processor systems (1986-1995)
- Operating Systems
Background

Sparc LEON
IA-32/64

MCS6502
M6800 & M6809
PDP 11/40e
TMS9900
IBM PC

AVR
HC08
C167
MPC60x
TC1796 (AUDO)
TC277T (AURIX)

320 node M68020-
16 node (dual CPU)
i860-based machines
Leitmotif

- embedded ≡ parallel: corresponds to embedded system as epitome of concurrent operation
- parallel ~ embedded: similar to parallel system as epitome of power guzzler and being sensitive to jitter
Outline

✔ prologue

❖ stock taking
  ❖ embedded computing system

❖ multi-core as reference point
  ❖ embedded ⇐ parallel
  ❖ parallel ∼ embedded

❖ epilogue
Embedded Computing System
Embedded system

- a microprocessor-based system
- that is built to control a function or range of functions and
- is not designed to be programmed by the end user in the same way that a PC is

(Heath, Embedded Systems Design, 1997)
Embedded system

- is designed to perform one particular task
- albeit with choices and different options
- has to communicate with the outside world
  - done by [a zoo of] peripherals

(Heath, Embedded Systems Design, 1997)
Embedded system

- any computer system hidden inside a product
- other than a computer

(Simon, An Embedded Software Primer, 1999)
Embedded systems

- have a microprocessor and a memory
- some have a serial port and a network connection
- they usually don't have keyboards, screens, or disk drives

(Simon, An Embedded Software Primer, 1999)
An exception that proves the rule...
Embedded system

- a computer system with a dedicated function
- within a larger mechanical or electrical system
- often with real-time computing constraints

(Wikipedia, 2013)
Embedded system

- range from portable devices
  - such as digital watches and MP3 players
- to large stationary installations
  - like traffic lights, factory controllers
- and large complex systems
  - like hybrid vehicles, MRI, and avionics

(Wikipedia, 2013)
often have several things to do at once
- they must respond to external events
- their work is subject so deadlines
- they must cope with all unusual conditions without human intervention

(Simon, An Embedded Software Primer, 1999)
An exception that proves the rule...
Extremes meet

- mass product
- small appliance
- resource shortage
- best effort
- non-/soft real-time
- planned obsolescence
- custom-built machinery
- giant equipment
- needs-based design
- dependable
- firm/hard real-time
- non-stop operation
Embedded $\equiv$ Parallel
Simultaneous operation

stirred-tank reactor

- functional units
- sensor system
- specific processing
- actuating elements

- mixed mode
  - periodic
  - aperiodic/sporadic
Simultaneous operation

personal trainer
- functional units
- sensor system
- specific processing
- actuating elements
- mixed mode
- periodic
- aperiodic/sporadic
Latent concurrency

- not because of internal constraints such as to improve system utilisation but...

induced by the characteristics of the actual object to be monitored or controlled

positioned through hardware features used to interact with the external process and

reflected by the logical structure of the corresponding internal process
Mix of parallelism: pseudo and real

- hardware multiplexing (CTSS, 1961)*
  - processing unit
  - address space, if applicable

- hardware multiplication (B5000, 1961)
  - processing unit, at least

partitioning in time or space, respectively

*a.k.a. partial virtualisation
Bottom line

- for embedded computing systems, multi-core technology is an implication
- "free lunch" never was an option in that domain — and never will be
- but the "menu" shows an even greater selection
Multi-core roots

MC68356, 1994

- first embedded triple-core
- CISC (MC68302)
- RISC (CP, 16550)
- DSP (MC56002)

heterogeneous
Multi-core roots

POWER4, 2001

first non-embedded dual-core, homogeneous
Being brought back down to earth...

- parallelism is challenging, but not the real problem in embedded systems
- and so is multi-core
- much more challenging is the handling of the multitude of different functional units
  - system control, power management
  - security, multimedia, connectivity, ...
Concrete example
— thousands of manual pages, excl. CPU —
Multi-core/processor

System on chip (MPSoC)

i.MX6
System on module
System in field
System in field
Favourite plaything
Rolling embedded system
Intranet on wheels

— but not for much longer —
Hybrid network
Electronic control unit

- engine management
- chassis applications
- body control module
- driver information system
- safety functions
- gateway operations
Breadboarding of a motor vehicle

Audi A6 (C6), detail

source: Audi AG
Network complexity

- number of ECUs: Audi A8

D2, 1993

5 vs. > 100

D4, 2010
Through the ages

Audi V8, 1991

Audi A3 (8P), 2012
Consumption factor

- length of 3km, weight of 60kg: not unusual...
- including ECUs ≈ 1l/100km or (US) 235mpg
Streamlining needed...
Consolidation

Virtualisation

Multi-Core
Consolidation

- **logical**
  - simplified operations, common processes

- **physical**
  - co-location of multiple platforms, fewer sites

- **workload**
  - more users, same application, fewer platforms

- **application**
  - combine mixed workloads, fewer platforms
Application consolidation

combines multiple applications of different types onto the same physical platform (i.e., ECU)
Constraint: Two-tier system

- **soft real-time**
  - QNX, CE, Linux

- **firm/hard real-time**
  - ITRON, AUTOSAR
Constraint: Transparency

- adopt application software as it stands
- library-like operating system (OS)
- OS and application program as a package
- ECU = casing = protection domain

- firm/hard
- real-time

application program

operating system

operating-system
machine level
Physical consolidation

- one application per ECU
- co-location of multiple ECUs
- single site: motor vehicle
- operating-system machine level (OSML)
- instruction set architecture level (ISAL)
Rationalised consolidation

- multiple applications per ECU, fewer ECUs
- system virtual machine level (SVML)
Rationalised consolidation

partitioning in time

OSML

SVML

ISAL

partitioning in space

OSML

SV

ML

IS

AL

* interference with (guest) operating system
Performance handicaps

- partial interpretation of system requests
  - traps, interrupts
- maintenance of real-machine state
  - processor state, shadow page tables, ...
- interference with guest operating system
  - scheduling, synchronisation
- interference with guest system(s) in general
  - cache-aware (machine) programs
Partitioning techniques

- with HW support
  - physical
  - logical
    - microprogramm
    - hypervisor
  - efficiency

- without HW support
  - SVM-based
  - homogeneous
  - heterogeneous
  - OS-based
  - flexibility
Partial virtualisation

- address-space/memory protection
- I/O-channel mapping
- static IRQ forwarding
- prevent false sharing
- cache lines!!!
- interference may break deadlines!!!
Safety applications

MPC564xL
Power-train applications

MPC5746M
Audi Calamaro Flying Car
Parallel $\sim$ Embedded
Parallel processing
Parallel processor: CPU
Parallel processor: GPU
Parallel system: HPC
Collective operations

- **gather**
  - collect data from all nodes

- **scatter**
  - split a set of data into pieces
  - send a different piece to all nodes

- **broadcast**
  - send same data to all nodes
Collective operations

- **reduce**
  - collect data from all nodes
  - combine collected data in some way
  - if applicable, send result to all nodes

- **barrier**
  - suspend the arriving process until all of one's peers have arrived
Outline of the problem

P1 — collective operation
P2 — idle
P3 — idle

P1 — delay
P2 — noise — idle — lag
P3 — idle — lag

theory
practice
Detrimental factors

- **process skew**
  - parallel operations cannot start at once
  - system noise delays processes by chance
  - process lags keep other processes waiting

- **data skew**
  - unbalanced (distributed) data sets
  - overloaded processes thwart under- or normally loaded processes, resp.
Solution statement

unbalanced (distributed) data sets
  partitioning, static load balancing

time-shifted start of parallel operations
  latency-aware process and data structures
  predictable operating-system processing

sporadic process delays
  co- or gang scheduling, resp., of processes
  holistic operating-system design
Energy consumption

Tianhe-2 (i.e., three-million-something cores)

- 17.6 MW the computing machine, alone
- 24 MW for external cooling, to be added
ultimate consumer
- high-speed train TGV: ≈ 20 MW
- medium-sized town in Germany: ≈ 48 MW

power generator: wind engine, 2.3 MW
- Tianhe-2 uncooled needs 9 installations
- Tianhe-2 cooled, a complete wind farm...
Potential „power supply“
Observing of predictions

- load-dependent power allocation
- stipulated by contract
- minimum payment clause
- chargeable unexpected underload

contract-aware deployment and scheduling

economise: waste energy to avoid a fine...
Near embedded systems

- "a priori" knowledge is all the world
- worst-case execution time (WCET)
- process and data dependency
- predictable run-time behavior

- special-purpose mode of operation
- foreseeable and timely processes

- resource-aware programming
- feature-oriented and holistic approach
Epilogue
Challenges

✔ consolidation

interference suppressed, temporal isolation

mode of operation

asymmetric, symmetric, bound

RAMS plus security (RAMSS)

reliability, availability, maintainability, safety
embedded computing systems

- are dedicated to handle a specific task
- life cannot possibly be imagined without it
- were forerunner of multi-core technology
- stop at nothing, neither virtualisation
- can serve as role models for "green HPC"