Challenging VMs on Battery-Powered Embedded Devices

Roel Wuyts
IMEC
KULeuven

VMIL’09, October 25, 2009
(OOPSLA’09 Workshop)
Juggling Hats

IMEC
- embedded device
- runtime resource management

ULB
- object versioning
- AOP

KUL
- language design
• **Micro-Electronics research organization located Leuven, Belgium**
  - Mission “To perform R&D, ahead of industrial needs by 3 to 10 years, in microelectronics, nanotechnology, design methods and technologies for ICT systems”

• **Numbers**
  - Budget: ± 200 M€
  - Staff: ± 1700
  - Cleanroom: ± 10,000 m²
Nomadic Devices
Nomadic Device Characteristics

- Power and energy constraints (battery)
- Design time constraints (time to market)
- Cost (area)
- Real-time constraints
- Flexibility and performance
Handheld Battery-powered Devices

![Pie chart showing the components of handheld devices:]

- 33% Wireless Modems
- 20% Application Processor
- 15% Backlight
- 15% Camera
- 7% Bluetooth
- 5% Speaker
- 3% Display
- 2% Other

[Kimmo Kuusilinna, Nokia, Date’08]
Current Market
Concrete Numbers for the Belgian market

- **Motorola:** 17 models, 66 models in support
- **Nokia:** 87 models, 185 models in support
- **Samsung:** 79 models, 238 models in support
- **LG:** 16 models, 77 models in support
• 6 ADRES processors
  - 4x4 array, 3-issue VLIW
  - 32-bit datapath
  - 16 video CODEC specific instructions
  - 8 FUs with multipliers
  - Performance: 300MHz

• 13 Communication assist
  - Performance: 75/150MHz

• ARTERIS NoC
  - Separate instr. and data NoC
  - Bandwidth: 5Gbps@150MHz

• ARM926
  - System control
  - Performance: 75MHz

• L2 memory
  - L2I: 2 banks of 512kB
  - L2D: 4 banks of 256kB

• Voltage islands
  - ADRES processors
  - L2I and L2D banks

• Multiple clock domains
Heterogeneous Multicore is here to stay

Key question: “How to write efficient programs cost-effectively?”
Programming Nomadic Devices

C/C++

Java
Dealing with variability: IMEC’s Approach

- Potential user & environment influence
- Potential platform parameters

MPSoC Mapping Flow/Tools

Design time

Implementation optimized application

Run time

Platform monitoring Information

Resource Management

Pareto management

User & environment constraints

Exploration information: Pareto surface

Execution time

Energy
Solution for MPSoC: IMEC MPA Tool

- **Parallelization directives**
  - Application code
    - `*.c`

- **Supported types of parallelism**
  - Functional split
  - (Coarse) Data-level split
  - Combinations

- **Correct-by-construction multi-threaded code**
- **Higher level than OpenMP**
- **Directives in separate file**

- **Parallelizes sequential Clean-C source code**
  - Correct-by-construction multi-threaded code
  - Higher level than OpenMP
  - Directives in separate file

- **Dumps parallel code**
- **Sets up communication**
  - Communication by means of FIFO’s
  - DMA transfers
  - FIFO sizes determined by tool (initial version)
Design Time Exploration

Per application

Set of operating points
In a multi-dimension space

> Energy consumption
> ...

> Number of used processors
> Memory usage
> Communication bandwidth
> ...

> Speed (Execution time)
> ...

> MPSoC RTM component configuration 1

> MPSoC RTM component configuration 2

Costs

Resource Constraints

Resource Usage

Sunday 25 October 2009
Run-time management

- For each thread frame, run-time scheduler changes management dynamically according to the run-time situations.

- Scheduler options based on design time exploration.
Run-time management

- For each thread frame, run-time scheduler changes management *dynamically* according to the run-time situations

- Scheduler options based on design time exploration
Run-time management

• For each thread frame, run-time scheduler changes management **dynamically** according to the run-time situations

---

![Graph showing workload, execution time, and power consumption over time](image)

• scheduler options based on design time exploration
Run-time management

- For each thread frame, run-time scheduler changes management *dynamically* according to the run-time situations.

- Scheduler options based on design time exploration.

![Graph showing workload vs. execution time](image)

- Workload (e.g., complexity of 3D object rendered) vs. Time
- Execution time vs. Power consumption
- Scheduler config 3
Adaptive Runtime Resource Management

- Assigns processing elements to tasks based on hardware and software static and dynamic data.
Recap

• **Hardware connection**
  - Use multiple heterogeneous hardware processing elements (CPU/GPU)
  - Control over the memory hierarchy (what data in what memory element)

• **Modularized VM**
  - General VM with parts that can be customized for particular hardware or applications, e.g. use scratchpad memory

• **Two-way communication between application and virtual machine**
  - Application can adapt to VM and VM can do global optimizations

• **VM that adapts its internal working to context**
  - Adapt GC algorithm
Questions...

App 1

App 2

OS 1 + drivers

OS 2 + drivers

bare metal

Sunday 25 October 2009
Questions...

- App 1
- App 2

what components?

- OS 1 + drivers
- OS 2 + drivers
- bare metal
Questions...

App 1

OS 1 + drivers

App 2

OS 2 + drivers

bare metal

what components?

composition technology?
AOP? traits? ...
Questions...

- Language?
- Runtime swapping?
- Deployment?
- VM language?

What components?

Composition technology?
AOP? traits? ...

OS 1 + drivers
OS 2 + drivers
bare metal