ASCI Winter School on Embedded Systems 2010

Cool MPSoC Design

Rainer Leupers
Software for Systems on Silicon (SSS)
RWTH Aachen University
ISS institute
- Part of RWTH Aachen EE+IT faculty
- Jointly managed by 3 professors (Ascheid, Leupers, Meyr)
- ~25 Ph.D. students, ~20 M.Sc. students + 10 staff

Focus on wireless communication + multimedia system design
- Interdisciplinary approach
- Basic research + tight industry cooperations
- Several EDA spin-off companies (e.g. Cadis, LISATek)
- UMIC excellence cluster
UMIC cluster project @ RWTH Aachen

- Excellence initiative competition of German elite universities
- „Ultra high speed mobile information & communication“
- Broadband wireless internet as infrastructure for new applications and economic opportunities
- Interdisciplinary (driven by RWTH EE + CS depts.)
- Funding: > 30 M€ (5 years program)
Why MPSoC at all?
Implications of Moore’s Law

Additional functionality & new architectures
Example: wireless multimedia terminals

- Multistandard radio
  - UMTS
  - GSM/GPRS/EDGE
  - WLAN
  - Bluetooth
  - UWB
  - …

- Multimedia standards
  - H.264
  - MP3
  - AAC
  - GPS
  - DVB-H
  - …

**Key issues:**
- Time to market (≤ 12 months)
- Flexibility (ongoing standard updates)
- Efficiency (battery operation)
### ITRS roadmap for wireless terminals

- Massive parallelism to satisfy computational requirements
  - (Multi-Processor System-on-Chip, MPSoC)
- Low power, energy

#### Frequency (MHz)

<table>
<thead>
<tr>
<th>Year</th>
<th>2003</th>
<th>2009</th>
<th>2013</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>300</td>
<td>600</td>
<td>1500</td>
</tr>
</tbody>
</table>

#### Giga Operations per Second

<table>
<thead>
<tr>
<th>Year</th>
<th>2003</th>
<th>2009</th>
<th>2013</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.3</td>
<td>14</td>
<td>2458</td>
</tr>
</tbody>
</table>

#### Operations per Cycle

<table>
<thead>
<tr>
<th>Year</th>
<th>2003</th>
<th>2009</th>
<th>2013</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>23</td>
<td>1639</td>
</tr>
</tbody>
</table>
Future mobile application: Software Defined Radio (SDR)

On-the-fly Configuration

Today's Mobile Phone

- Bluetooth.exe
- GSM.exe

Free Area: Cost Savings or New Functionality

Future SDR Mobile Phone

- Bluetooth.exe
- UMTS.exe
- GSM.exe

Flexible SDR

Heterogeneous Multi-Processor System-on-Chip (MPSoC)

Source:
- Infineon Technologies
How to efficiently design and deploy heterogeneous MPSoC platforms?

ESL: Electronic System Level Design
So what’s the next level of abstraction?

Processor is the hand gate of the future

- 10-100X on Simulation
- 10-100X on design productivity
- Impose design restrictions

Effort

1978

Transistor level

1985

Gate/logic level

1992

Register transfer level

1999

Transactor Level

Results

(Transistors per engineering effort)
Designing application specific processing elements (ASIPs)
Processing elements: efficiency vs flexibility

- General Purpose Processors
- Digital Signal Processors
- Application Specific Instruction Set Processors
- Field Programmable Devices
- Application Specific ICs
- Physically Optimized ICs

Source: T.Noll, RWTH Aachen
Tool based ASIP architecture exploration

initial processor architecture

optimized processor architecture
LISATek ASIP design workbench

- Unified processor model in LISA 2.0 architecture description language (ADL)
- Integrated processor development environment
- Automatic generation of:
  - SW tools
  - HW models
- Now available as CoWare Processor Designer
- Future: Synopsys

The LISATek™ Solution

Automated Embedded Processor Design and Software Development Tool Generation

LISATek is an automatic unified processor design and optimization environment that enables designers to create processor cores with the processor development environment for processors. It generates software development tools for processors that have not been designed using HLS (Hardware Language). LISATek is a software development tool for processors and performance models. LISATek is a software development tool for processors and performance models.

LISATek is a software development tool for processors and performance models. LISATek is a software development tool for processors and performance models.

LISATek is a software development tool for processors and performance models. LISATek is a software development tool for processors and performance models.

LISATek is a software development tool for processors and performance models. LISATek is a software development tool for processors and performance models.

LISATek is a software development tool for processors and performance models. LISATek is a software development tool for processors and performance models.

LISATek is a software development tool for processors and performance models. LISATek is a software development tool for processors and performance models.

LISATek is a software development tool for processors and performance models. LISATek is a software development tool for processors and performance models.

LISATek is a software development tool for processors and performance models. LISATek is a software development tool for processors and performance models.

LISATek is a software development tool for processors and performance models. LISATek is a software development tool for processors and performance models.

LISATek is a software development tool for processors and performance models. LISATek is a software development tool for processors and performance models.

LISATek is a software development tool for processors and performance models. LISATek is a software development tool for processors and performance models.

LISATek is a software development tool for processors and performance models. LISATek is a software development tool for processors and performance models.

LISATek is a software development tool for processors and performance models. LISATek is a software development tool for processors and performance models.

LISATek is a software development tool for processors and performance models. LISATek is a software development tool for processors and performance models.

LISATek is a software development tool for processors and performance models. LISATek is a software development tool for processors and performance models.

LISATek is a software development tool for processors and performance models. LISATek is a software development tool for processors and performance models.

LISATek is a software development tool for processors and performance models. LISATek is a software development tool for processors and performance models.

LISATek is a software development tool for processors and performance models. LISATek is a software development tool for processors and performance models.

LISATek is a software development tool for processors and performance models. LISATek is a software development tool for processors and performance models.

LISATek is a software development tool for processors and performance models. LISATek is a software development tool for processors and performance models.

LISATek is a software development tool for processors and performance models. LISATek is a software development tool for processors and performance models.

LISATek is a software development tool for processors and performance models. LISATek is a software development tool for processors and performance models.

LISATek is a software development tool for processors and performance models. LISATek is a software development tool for processors and performance models.

LISATek is a software development tool for processors and performance models. LISATek is a software development tool for processors and performance models.
MPSoC architecture exploration and early SW performance estimation
MPSoC exploration principles

- Divide and conquer
- Separate processing elements from communication
- Early SW performance estimation
- VPU: virtual processing unit
- Abstract ISS, executing timing annotated native code
- Enables modeling spatial and temporal task-to-PE mapping
VPU based MPSoC Exploration and Partitioning

Task A1 → Task A2 → Task A3

VPU A
- memory
- local scheduler

VPU B
- cache
- local scheduler

Communication Architecture
- global scheduler
- Global Memory

© 2010 R. Leupers
Current work: VPU timing annotation

- Statistical model instead of detailed instrumentation
- Which modeling granularity?
- Which parameters are more dependent on PE type rather than SW task?

Diagram of VPU with Processing Model, Transactors, and Task interactions:

- Randomness
- Clk
- Data Port
- Instruction Port
- Data Bus
- Instruction Bus
- tm_consume()

Code snippet:

```c
... annotation.processing_cycles = 10;
annotation.fetch_ratio = 0.4;
annotation.load_ratio = 0.2;
annotation.store_ratio = 0.1;
tm_consume(&annotation);
...```
Virtual Platforms and Advanced MPSoC Simulation
What is a virtual platform?

- A SW model of a HW SoC platform

- Enables...
  - HW platform architecture exploration and optimization
  - SW development, debugging, and optimization
  - Concurrent HW/SW design ("HW/SW codesign")

- Requirements
  - High simulation speed
  - Speed/accuracy trade-off
  - Flexibility
  - Usability for non-HW-experts
**It’s Like Hardware – Only Better!**

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Early Availability</td>
<td>available before the chips come back from the fab and boards have been debugged</td>
</tr>
<tr>
<td>Enhanced Debugging</td>
<td>full visibility and control with non intrusive access to all components</td>
</tr>
<tr>
<td>Easy to Deploy</td>
<td>minimal user ramp up time and logistical effort to distribute and update</td>
</tr>
</tbody>
</table>

*J. Kunkel, Synopsys, MPSoC 2007*
Example: Chumby Virtual Platform

“The chumby is a compact device that displays useful and entertaining information from the web: news, photos, music, weather, celebrity gossip, webcams, sports scores - using your wireless internet connection”

- Open hardware and software
- Virtual platform allows HW/SW exploration with pure SW simulation model in almost real-time
Need for speed

- Instruction set simulator (ISS) is at the heart of virtual platforms
- ISS speed evolution
  - Early interpretive: few KIPS
  - Fast interpretive: ~100 KIPS
  - Compiled: ~10 MIPS
  - SW Sim. Cache: ~10 MIPS
  - Binary translation: 50-100 MIPS
- Challenge:
  - Instruction-accurate ISS technology has been pushed to its limits
  - How to handle future many-core MPSoC?
Three possibilities to speed up simulations:

- Raising the abstraction level
- Checkpoint/Restart
- Exploiting latest multi-core host processors
Hybrid Simulator (HySim) overview

Parallel SystemC simulation on multicore hosts

Small example:
- Multi-RISC target platform
- Multi-Core PC host
- Cooperative computation of a Mandelbrot set
## Performance Data – 4x4 AMD Barcelona speedup

<table>
<thead>
<tr>
<th>Simulation</th>
<th>Speed-up Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.00</td>
</tr>
<tr>
<td>2</td>
<td>2.00</td>
</tr>
<tr>
<td>3</td>
<td>3.00</td>
</tr>
<tr>
<td>4</td>
<td>4.00</td>
</tr>
<tr>
<td>5</td>
<td>5.00</td>
</tr>
<tr>
<td>6</td>
<td>6.00</td>
</tr>
<tr>
<td>7</td>
<td>7.00</td>
</tr>
<tr>
<td>8</td>
<td>8.00</td>
</tr>
</tbody>
</table>

The chart above illustrates the speed-up factor for different simulation numbers (1 to 16) as a function of the number of CPUs (x-axis: sinCPUs, y-axis: speed-up factor). The data suggests a linear increase in speed-up with increasing number of CPUs.
MPSoC programming challenges
Evolution of embedded compiler challenges

- **Past issue:**
  - Introduction of special purpose processor architectures (DSPs, NPUs, ...)
  - Compiler support as afterthought -> suboptimal HW/SW systems

- **Solution:**
  - Compiler/architecture codesign
  - Implemented in many designs

- **Today´s issue:**
  - Wide introduction of multi/many-core SoC in all computing areas
  - Compiler support as afterthought
Challenge 1: how to program MPSoC?

- Sequential C programming cannot be easily replaced
  - lots of certified legacy code
  - need new programmers education
- Algorithm design frequently delivers parallel spec
  - need support for standard MoCs e.g. KPN, SDF
- No chances for radically new programming languages
- Requirements
  - gradual SW migration path
  - see e.g. OpenMP in HPC
  - adapted to embedded SW and MPSoC platforms
  - including: automated C code partitioning
Challenge 2: how to program really cool?

- Mobile devices perform **multitasking just like PCs**
- But: much more **constrained**
  - Energy efficiency
  - Real-Time
    - hard
    - soft
    - best effort

• **Scheduling** for multitasking scenarios
  • PC: best effort only, „unlimited“ energy
  • mobile: must be able to handle **worst-case scenario**, not more and not less
  • e.g. HSPA download + H.264 + MP3 + Bluetooth + ...
  • mix of **static** and **dynamic** scheduling
Challenge 3: platform heterogeneity

- Classically: predetermined task-to-processor assignment
- Heterogenous multicore:
  - dynamic task migration
  - more flexibility
  - ambiguous task execution timing
- See e.g. TI OMAP
Challenge 4: how to program correctly?

- We can’t do MPSoC programming w/o new MPSoC debugging technology
- Current debuggers focused on single cores
- MPSoC debug requirements:
  - fast simulation
  - higher abstraction level
    - e.g. capture and visualize task communication/sync events
  - address MPSoC specific SW bugs
    - data races, deadlocks, ...
- Infrastructure: Virtual Platform technology
  - full SoC simulation
  - non-intrusive instrumentation
MPSoC programming: The MAPS approach
UMIC, Nucleus, and MAPS

- **UMIC cluster:**
  - large compound project at Aachen
  - focus on all aspects of next generation mobile internet

- **Nucleus project:**
  - novel SoC design methodology for wireless terminals
  - including algorithms, architectures, and tools
  - identification, implementation, and mapping to critical computational kernels (Nuclei)

- **MAPS tool suite:**
  - Nuclei aware MPSoC SW mapping methodology and tools
MAPS programming model

- N (potentially simultaneous) applications
  - Each given in either sequential C code or C extension for KPNs
  - Realtime class + constraints
  - „Preferred“ PE type
  - Concurrency graph models multitasking scenarios
  - Automated code (re)partitioning
  - Mixed static/dynamic task scheduling

```c
__fifo int A, B;
#pragma maps process rle_dec, in(A), out(B),
prefer(risc)
{int cnt, val, i;} // Local variables
{  // Process body:
cnt = A;    // Reads first token: count
val = A;    // Reads second token: value
for (i = 0; i < cnt; ++i) {
  B = val;  // Outputs count times val
}
```
Snapshot: parallel code functional verification

- **MAPS Virtual Platform (MVP)**
  - high-level: abstract PEs, SystemC based
  - low-level: ISS based Virtual Platform
- „mPhone“ virtual smartphone
Summary

- **MPSoc architecture design requires new sophisticated ESL design tools**
  - RTL design runs out of steam
- **ASIPs are key components of heterogeneous MPSoc**
  - ADL based design methodology established
  - Put C compiler into the architecture exploration loop
  - New trends: automatic ISE synthesis, reconfigurable ASIPs
- **Virtual platforms are key for complex MPSoc design**
  - Fast application SW performance estimation
  - Verification
  - HW/SW co-design (co = concurrent)
- **MPSoc compilation is the next grand challenge**
  - Paradigm shift from sequential to parallel code generation
  - Vision: hide HW complexity from programmer as far as possible
  - HW support for OS functionality required
Thank you!