Ubiquitous embedded systems

Embedded System Engineering (ESE) requires:
- Hardware (HW) / (E)Software (ESW) co-design
- Configure (CW) / ESW co-design
- HW / CW / ESW co-design

ESW becomes main vehicle to product differentiation
ESE becomes the main focus in system design:

Reiner Hartenstein, University of Kaiserslautern, Germany
http://hartenstein.de

Reconfigurable Computing:
a second programming domain

Migration of programming to the structural domain
The structural domain has become RAM-based
The opportunity to introduce the structural domain to programmers ...
... to bridge the gap by clever abstraction mechanisms using a simple new machine paradigm

Embedded System Design Crisis

More crises

[ST microelectronics] Mask & NRE cost

Reiner Hartenstein: A Mead-&-Conway-like Break-through is overdue; Seminar Nº 03301, Dynamically Reconfigurable Architectures; Dagstuhl, Germany, July 20-25, 2003
Foundries: Adoption Rate By Process

EDA industry shifts into CS mentality

"patches instead of engineering"
"innovation stalled many years ago"
"netlist-based: do not care about efficiency, …"
"… do not care about transistor density"
"85% users hate their tools"

Where are we heading?

Lacking Sense of Direction

Dead Supercomputer Society

© 2003, reiner@hartenstein.de
http://hartenstein.de
Some Supercomputing people now looking at us

Steroids for the aging microprocessor:

Reconfigurable Computing

PetaFlop/s (10^15) Initiative

© 2003, reiner@hartenstein.de

Kaiserslautern University of Technology

CS: young ? dynamic?
... but the von Neumann Paradigm is still the dominant doctrine ...
... still pushing he basic models from the times of mainframe dinosaurs
Microelectronics is ignored (since taking cost of computational effort)
A Re-orientation is over-due

© 2003, reiner@hartenstein.de

Kaiserslautern University of Technology

processor/memory communication bottleneck

vN: unbalanced
vN bottleneck

© 2003, reiner@hartenstein.de

Kaiserslautern University of Technology

MPU designs more complex
new kinds of concurrency are becoming important
chip-level multiprocessing + simultaneous multithreading
many bugs relate to concurrency issues
greatly complicates the verification process

© 2003, reiner@hartenstein.de

Kaiserslautern University of Technology

>> outline (3) <<

- Embedded System Design Crisis
- Supercomputing Crisis
- µP Crisis
- CS crisis
- CS for Embedded Systems?
- New Machine Paradigm
- final remarks

© 2003, reiner@hartenstein.de

Kaiserslautern University of Technology

© 2003, reiner@hartenstein.de

Kaiserslautern University of Technology
Reiner Hartenstein, University of Kaiserslautern, Germany
http://hartenstein.de
reiner@hartenstein.de
July 20, 2003

MPU performance stalled

Bill Gates' law:
relative computation time needed
doubles every 2 years
had been compensated by Moore’s law
Moore’s law will stall soon for MPUs

“Pollack’s Law” (simplified)
growth factor
area efficiency
performance

more and more efforts yield
only marginal improvements
areas fade away
dataflow machines dead
shrinking supercomputing conferences
98.5% VfN-only
this monopoly
is the problem

>> outline (4) <<

- Embedded System Design Crisis
- Supercomputing Crisis
- µP Crisis
- CS crisis
- CS for Embedded Systems?
- New Machine Paradigm
- final remarks

Crusty Computing Sciences

Stealthy CS Crisis

Lacking Sense of Direction ?

"we are o.k. !" (no new direction)

severe software quality problems
progress in CS stalled by qualification problems
in industry and academia
often hardware people needed to solve CS problems
communication barriers between disciplines

blinders:
for ignoring the impact of RC

Reiner Hartenstein: A Mead-&-Conway-like Break-through is overdue; Seminar Nº 03301, Dynamically Reconfigurable Architectures; Dagstuhl, Germany, July 20-25, 2003
What's the problem?

It's the gap between procedural and structural mind set.

Traditional CS: programming is (control-)procedural, instruction-stream-based - sources: software.

The typical programmer has problems to understand function evaluation without machine mechanisms....

... by signals rippling through a network of transistors.

What's the problem?

The brain hurts on paradigm shift?

no, it can't ...

Brain usage:

procedural-

-structural

hemisphere missing

Crossing the Hardware / Software Chasm

[Mike Butts]

>> outline (5) <<

- Embedded System Design Crisis
- Supercomputing Crisis
- µP Crisis
- CS crisis
- CS for Embedded Systems?
- New Machine Paradigm
- final remarks

SoC System level Design: Embedded SW (ESW) (ECW)

ESW becomes main vehicle to product differentiation

ECW

ESE becomes the main focus in system design:

CW: HW-(E)SW co-design onto highly programmable platforms (SoC)

new design automation from high level descriptions

CW and SW synthesis included (SoC)

CW: HW-(E)SW co-verification.

formal verification for (E)SW and CW

Complexity: System Level Design Challenge

"abstraction levels must be raised above present-day RT-level from HW = (processor-dependent embedded) C code level language infrastructures for complex models (SystemC etc.) must be leveraged by industry consensus on use-methodology and abstraction levels"
Reiner Hartenstein, University of Kaiserslautern, Germany
http://hartenstein.de

Reiner Hartenstein: A Mead-\&-Conway-like Break-through is overdue; Seminar Nº 03301, Dynamically Reconfigurable Architectures; Dagstuhl, Germany, July 20-25, 2003
Reiner Hartenstein, University of Kaiserslautern, Germany
http://hartenstein.de
reiner@hartenstein.de
July 20, 2003

---

Programming Language Paradigms

<table>
<thead>
<tr>
<th>Language Category</th>
<th>Computer Languages</th>
<th>Languages f. Anti-Machines</th>
</tr>
</thead>
<tbody>
<tr>
<td>both deterministic</td>
<td>procedural sequencing, branchable, checkpointable</td>
<td>read next instruction, jump to data addr., jump to instr. addr., branch by data loop nesting, parallel loops, escapes, instruction stream branching</td>
</tr>
<tr>
<td>procedural sequence driven by:</td>
<td></td>
<td>read next data item, jump to data addr., data loop, loop nesting, parallel loops, escapes, data stream branching</td>
</tr>
<tr>
<td>data register address, computation instruction fetch</td>
<td>machine memory cycle overhead, memory cycle overhead, data overhead, overhead avoided</td>
<td>overhead avoided, no restrictions</td>
</tr>
<tr>
<td>state register, program counter, data counter(s)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

Machine paradigms

von Neumann instruction stream machine

Flowware

- Instruction sequence
- Instruction stream generator
- Instruction stream

DPU or rDPU

Distributed Memory

SA: scrambling and descrambling the data?

Just in time: a new research area:

Application-specific distributed memory:

e.g. book by F. Catthoor et al. ...

Data address generators - 20 years research:

---

Synthesizable distributed memory architecture... for a Stream-based Soft Machine

Memory (data memory)

Compiler

Scheduler

rDPA

Sequencers (data stream generator)

---

Outline (7)

• Embedded System Design Crisis
• Supercomputing Crisis
• µP Crisis
• CS crisis
• CS for Embedded Systems?
• New Machine Paradigm
• Final remarks

http://www.uni-kl.de
Conclusion: all knowledge needed is available

- machine paradigm
- languages
- hw / sw partitioning methodology
- compilation techniques
- anti machine architectural resources
- sequencing methodology: hw & sw
- parallel memory IP core and module generator vendors
- anything else needed

The Situation in Computing Sciences

- Computing Sciences are in a severe crisis
- New fundamentals and R&D directions are inevitable
- my mission: getting you involved
- All knowledge needed is readily available ... 
- ... even from Computing Sciences
- Silicon application and EDA provide useful concepts
- Reconfigurable Computing has the remedy

We need a Mead- &- Conway-like text book

We need undergraduate lab courses on HW / CW / SW partitioning

We need new courses with extended scope on parallelism and algorithmic cleverness for HW / CW / SW migration / partitioning

What else do we need ? Your proposals ?

We need the support of the open-minded members of the classical CS community

Let us assemble a list with e-mail addresses

thank you for your patience
Reiner Hartenstein, University of Kaiserslautern, Germany
http://hartenstein.de
reiner@hartenstein.de
July 20, 2003

Reiner Hartenstein: A Mead- & Conway-like Break-through is overdue; Seminar Nº 03301, Dynamically Reconfigurable Architectures; Dagstuhl, Germany, July 20-25, 2003
2nd machine paradigm

The World of Matter
machine paradigm: the Atom

The World of Anti Matter
machine paradigm: Anti Atom

Matter & Antimatter of Informatics:

The Secret of Success: Co-Compilation

One more argument for coarse grain
we have already seen the first day:

High level PL source

“VH” machine paradigm

Partitioner

SW compiler

Analyzer / Profiler

SW code

CW Code

Resource Parameters

“VH” machine paradigm

High level PL source

“The Secret of Success: Co-Compilation

One more argument for coarse grain
we have already seen the first day:

High level PL source

“VH” machine paradigm

Partitioner

SW compiler

Analyzer / Profiler

SW code

CW Code

Resource Parameters

“The Secret of Success: Co-Compilation

One more argument for coarse grain
we have already seen the first day:

High level PL source

“VH” machine paradigm

Partitioner

SW compiler

Analyzer / Profiler

SW code

CW Code

Resource Parameters

“The Secret of Success: Co-Compilation

One more argument for coarse grain
we have already seen the first day:

High level PL source

“VH” machine paradigm

Partitioner

SW compiler

Analyzer / Profiler

SW code

CW Code

Resource Parameters

“The Secret of Success: Co-Compilation

One more argument for coarse grain
we have already seen the first day:

High level PL source

“VH” machine paradigm

Partitioner

SW compiler

Analyzer / Profiler

SW code

CW Code

Resource Parameters

“The Secret of Success: Co-Compilation

One more argument for coarse grain
we have already seen the first day:

High level PL source

“VH” machine paradigm

Partitioner

SW compiler

Analyzer / Profiler

SW code

CW Code

Resource Parameters

“The Secret of Success: Co-Compilation

One more argument for coarse grain
we have already seen the first day:

High level PL source

“VH” machine paradigm

Partitioner

SW compiler

Analyzer / Profiler

SW code

CW Code

Resource Parameters

“The Secret of Success: Co-Compilation

One more argument for coarse grain
we have already seen the first day:

High level PL source

“VH” machine paradigm

Partitioner

SW compiler

Analyzer / Profiler

SW code

CW Code

Resource Parameters

“The Secret of Success: Co-Compilation

One more argument for coarse grain
we have already seen the first day:

High level PL source

“VH” machine paradigm

Partitioner

SW compiler

Analyzer / Profiler

SW code

CW Code

Resource Parameters

“The Secret of Success: Co-Compilation

One more argument for coarse grain
we have already seen the first day:

High level PL source

“VH” machine paradigm

Partitioner

SW compiler

Analyzer / Profiler

SW code

CW Code

Resource Parameters

“The Secret of Success: Co-Compilation

One more argument for coarse grain
we have already seen the first day:

High level PL source

“VH” machine paradigm

Partitioner

SW compiler

Analyzer / Profiler

SW code

CW Code

Resource Parameters

“The Secret of Success: Co-Compilation

One more argument for coarse grain
we have already seen the first day:

High level PL source

“VH” machine paradigm

Partitioner

SW compiler

Analyzer / Profiler

SW code

CW Code

Resource Parameters

“The Secret of Success: Co-Compilation

One more argument for coarse grain
we have already seen the first day:

High level PL source

“VH” machine paradigm

Partitioner

SW compiler

Analyzer / Profiler

SW code

CW Code

Resource Parameters

“The Secret of Success: Co-Compilation

One more argument for coarse grain
we have already seen the first day:

High level PL source

“VH” machine paradigm

Partitioner

SW compiler

Analyzer / Profiler

SW code

CW Code

Resource Parameters

“The Secret of Success: Co-Compilation

One more argument for coarse grain
we have already seen the first day:

High level PL source

“VH” machine paradigm

Partitioner

SW compiler

Analyzer / Profiler

SW code

CW Code

Resource Parameters

“The Secret of Success: Co-Compilation

One more argument for coarse grain
we have already seen the first day:

High level PL source

“VH” machine paradigm

Partitioner

SW compiler

Analyzer / Profiler

SW code

CW Code

Resource Parameters

“The Secret of Success: Co-Compilation

One more argument for coarse grain
we have already seen the first day:

High level PL source

“VH” machine paradigm

Partitioner

SW compiler

Analyzer / Profiler

SW code

CW Code

Resource Parameters

“The Secret of Success: Co-Compilation

One more argument for coarse grain
we have already seen the first day:

High level PL source

“VH” machine paradigm

Partitioner

SW compiler

Analyzer / Profiler

SW code

CW Code

Resource Parameters

“The Secret of Success: Co-Compilation

One more argument for coarse grain
we have already seen the first day:

High level PL source

“VH” machine paradigm

Partitioner

SW compiler

Analyzer / Profiler

SW code

CW Code

Resource Parameters

“The Secret of Success: Co-Compilation

One more argument for coarse grain
we have already seen the first day:

High level PL source

“VH” machine paradigm

Partitioner

SW compiler

Analyzer / Profiler

SW code

CW Code

Resource Parameters

“The Secret of Success: Co-Compilation

One more argument for coarse grain
we have already seen the first day:

High level PL source

“VH” machine paradigm

Partitioner

SW compiler

Analyzer / Profiler

SW code

CW Code

Resource Parameters

“The Secret of Success: Co-Compilation

One more argument for coarse grain
we have already seen the first day:

High level PL source

“VH” machine paradigm

Partitioner

SW compiler

Analyzer / Profiler

SW code

CW Code

Resource Parameters

“The Secret of Success: Co-Compilation

One more argument for coarse grain
we have already seen the first day:

High level PL source

“VH” machine paradigm

Partitioner

SW compiler

Analyzer / Profiler

SW code

CW Code

Resource Parameters

“The Secret of Success: Co-Compilation

One more argument for coarse grain
we have already seen the first day:

High level PL source

“VH” machine paradigm

Partitioner

SW compiler

Analyzer / Profiler

SW code

CW Code

Resource Parameters

“The Secret of Success: Co-Compilation

One more argument for coarse grain
we have already seen the first day:

High level PL source

“VH” machine paradigm

Partitioner

SW compiler

Analyzer / Profiler

SW code

CW Code

Resource Parameters

“The Secret of Success: Co-Compilation

One more argument for coarse grain
we have already seen the first day:

High level PL source

“VH” machine paradigm

Partitioner

SW compiler

Analyzer / Profiler

SW code

CW Code

Resource Parameters

“The Secret of Success: Co-Compilation

One more argument for coarse grain
we have already seen the first day:

High level PL source

“VH” machine paradigm

Partitioner

SW compiler

Analyzer / Profiler

SW code

CW Code

Resource Parameters

“The Secret of Success: Co-Compilation

One more argument for coarse grain
we have already seen the first day:

High level PL source

“VH” machine paradigm

Partitioner

SW compiler

Analyzer / Profiler

SW code

CW Code

Resource Parameters

“The Secret of Success: Co-Compilation

One more argument for coarse grain
we have already seen the first day:

High level PL source

“VH” machine paradigm

Partitioner

SW compiler

Analyzer / Profiler

SW code

CW Code

Resource Parameters

“The Secret of Success: Co- Compilation

One more argument for coarse grain
we have already seen the first day:

High level PL source

“VH” machine paradigm

Partitioner

SW compiler

Analyzer / Profiler

SW code

CW Code

Resource Parameters

“The Secret of Success: Co-Compilation

One more argument for coarse grain
we have already seen the first day:

High level PL source

“VH” machine paradigm

Partitioner

SW compiler

Analyzer / Profiler

SW code

CW Code

Resource Parameters

“The Secret of Success: Co-Compilation

One more argument for coarse grain
we have already seen the first day:

High level PL source

“VH” machine paradigm

Partitioner

SW compiler

Analyzer / Profiler

SW code

CW Code

Resource Parameters

“The Secret of Success: Co-Compilation

One more argument for coarse grain
we have already seen the first day:

High level PL source

“VH” machine paradigm

Partitioner

SW compiler

Analyzer / Profiler

SW code

CW Code

Resource Parameters

“The Secret of Success: Co-Compilation

One more argument for coarse grain
we have already seen the first day:

High level PL source

“VH” machine paradigm

Partitioner

SW compiler

Analyzer / Profiler

SW code

CW Code

Resource Parameters

“The Secret of Success: Co-Compilation

One more argument for coarse grain
we have already seen the first day:

High level PL source

“VH” machine paradigm

Partitioner

SW compiler

Analyzer / Profiler

SW code

CW Code

Resource Parameters

“The Secret of Success: Co-Compilation

One more argument for coarse grain
we have already seen the first day:
Reiner Hartenstein, University of Kaiserslautern, Germany
http://hartenstein.de
reiner@hartenstein.de
July 20, 2003
**Super Pipe Networks**

The key is **mapping**, rather than architecture.

<table>
<thead>
<tr>
<th>array</th>
<th>applications</th>
<th>pipeline properties</th>
<th>mapping</th>
<th>scheduling (data stream formation)</th>
</tr>
</thead>
<tbody>
<tr>
<td>systolic</td>
<td>regular data dependencies only</td>
<td>linear only</td>
<td>uniform only</td>
<td>linear projection or algebraic synthesis</td>
</tr>
</tbody>
</table>

KressArray [1995]

---

**distributed memory**

---

**Instruction stream-based Compilation Principles**

source text

parser

I-D memory space

link/load

library

instruction cell placement

scheduler

execution order by location

---

**Datastream-based Compilation Principles**

library

mapper

Scheduler

data stream assembly

placement & routing

---

**flowware languages**

---

**Similar Programming Language Paradigms**

<table>
<thead>
<tr>
<th>language category</th>
<th>Computer Languages</th>
<th>Xputer Languages</th>
</tr>
</thead>
<tbody>
<tr>
<td>both deterministic</td>
<td>procedural sequencing: traceable, checkpointable</td>
<td></td>
</tr>
<tr>
<td>sequencing driven by:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>real next instruction, goto (instruction addr.), jump (to instruction addr.), instruction loop, instruction loop nesting in parallel loops, instruction loop escapes, instruction stream branching</td>
<td>real next data object, goto (data addr.), jump (to data addr.), data loop, data loop nesting, parallel data loops, data loop escapes, data stream branching</td>
</tr>
</tbody>
</table>
Reiner Hartenstein, University of Kaiserslautern, Germany
http://hartenstein.de
reiner@hartenstein.de
July 20, 2003

address generators

Reiner Hartenstein: A Mead-&-Conway-like Break-through is overdue; Seminar Nº 03301, Dynamically Reconfigurable Architectures; Dagstuhl, Germany, July 20-25, 2003
Reiner Hartenstein, University of Kaiserslautern, Germany
http://hartenstein.de
reiner@hartenstein.de
July 20, 2003

GAG Complex Seuenecer Implementation

Generic Sequence Examples

GAG Slider Operation Demo Example

Storage scheme optimization: scanline unrolling
MoM anti machine architecture

"von Neumann" Computer: the wrong Machine Paradigm

Binding Time vs. Computing Domain

Reiner Hartenstein: A Mead-&-Conway-like Break-through is overdue; Seminar Nº 03301, Dynamically Reconfigurable Architectures; Dagstuhl, Germany, July 20-25, 2003

14
Reiner Hartenstein: A Mead- & Conway-like Break-through is overdue; Seminar Nº 03301, Dynamically Reconfigurable Architectures; Dagstuhl, Germany, July 20-25, 2003
Parallelism by Concurrency

independent instruction streams

Loop Transformation Examples

resource parameter driven
Co-Compilation

host: recent.array:

loop 1-8
target

loop 1-4
target

loop 1-2
target

sequential processes:

loop 1-16

body

endloop

loop 1-8

body

endloop

fork

loop 9-16

body

endloop

join

unrolling

strip mining

We introduce: Co-Compilation

Software running on Computer Machine Paradigm

High level programming language source

partitioning compiler

Software running on Xputer "Soft" Machine Paradigm

Reconfigurable Accelerators

Configware running on Reconfigurable Architecture (RA) -- instead of hardened

Co-Compilation

Xputer

"Soft" Machine Paradigm

Configware running on partitioning compiler

high level programming language source

m Processor

Reconfigurable

Accelerators

Reconfigurable Architecture (RA) -- instead of hardwired

Loop Transformation Examples

resource parameter driven
Co-Compilation

host: recent.array:

loop 1-8
target

loop 1-4
target

loop 1-2
target

sequential processes:

loop 1-16

body

endloop

loop 1-8

body

endloop

fork

loop 9-16

body

endloop

join

unrolling

strip mining

We introduce: Co-Compilation

Software running on Computer Machine Paradigm

High level programming language source

partitioning compiler

Software running on Xputer "Soft" Machine Paradigm

Reconfigurable Accelerators

Configware running on Reconfigurable Architecture (RA) -- instead of hardened

Co-Compilation

Xputer

"Soft" Machine Paradigm

Configware running on partitioning compiler

high level programming language source

m Processor

Reconfigurable

Accelerators

Reconfigurable Architecture (RA) -- instead of hardwired

Why data streams are a common model

all other details are defined here:

Configware: programming the resources
Flowware: to schedule data streams

Data streams (flowware) are derived from configware having been compiled before

Data stream execution resources: distributed memory architectures. This new discipline came just in time.

see Herz et al.: Proc. IEEE ICECS 2002

Link (via "recent talks") also here: