Twin-Paradigm
Programmer Education: proposed New Text Book (and project)

Reiner Hartenstein, TU Kaiserslautern, Germany
http://hartenstein.de

Professor (ordinarius emeritus), TU Kaiserslautern
All academic degrees from EE department at Karlsruhe Institute of Technology (KIT)
1981: visiting professor at UC Berkeley
IEEE fellow, SDPS fellow, FPL fellow, other awards
Founder and co-founder of several international annual conference series

Viktor Prasanna called him „The father of Reconfigurable Computing“ (also pre-FPGA era) -- (Gerald Estrin: grandfather of RC)
1983: founder of German Mead–Conway VLSI design scene: the multi university „E.I.S. project“ (grant: 35 million DM)

Author & compiler writer of KARL[1]: in the 80ies the most successful HDL and trailblazer before VHDL

1977, North Holland / American Elsevier -- bestseller

Complete EDA framework[2] around KARL and ABL: 85 mio ECU grant by ESPRIT programme of the EU
format-checking functional floorplan graphic editor --- automatic test generation, testability analysis,
calculus-based term rewriting simulator --- embedded router, structured logic synthesis,
floorplan generator, structured logic synthesis,

Book Outline

• Introduction
• Instruction Stream Machines
• Data Stream Machines
• Time to Space Mapping
• Hetero Systems
• Applications
• Outlook
How Societies Chose to Fail or Succeed

Many-core failure could jeopardize both IT industry & sections of the economy depending on rapid improvement of IT. [Dave Patterson]

Several recent outages of cloud computing services.

Collapse of our computing ecosystem?

Unaffordability of von-Neumann-centric computing could jeopardize all facets of our global economy.

Motivations for a Game Change

Energy consumption of all computers worldwide may become unaffordable.

To enable mastering the impact of manycore the programmer population is not yet existing.

Fundamentally new types of skills for programmers promise to cool down the chronic “software crisis”

Energy-efficient Computing

2 different approaches:

- Low power circuit design*, SSD, & „green computers“**:
  It’s important: all sections should be supported

- The Twin Paradigm approach:
  Promising results better by orders of magnitude
  Technology established, but massive commitment required

Re-thinking basic assumptions

Instead of physical limits, fundamental misconceptions of algorithmic complexity theory limit the progress and will necessitate new breakthroughs.

Not processing is costly, but moving data and messages.

We’ve to re-think basic assumptions behind computing

Put old ideas into practice (POII)

“We need a complete re-definition of curriculum recommendations - missing several key issues.” [Reiner Hartenstein]

Wrong! I do not agree, finding out, that ...

...The biggest payoff will come from putting old ideas into practice and teaching people how to apply them properly.” [David Parnas]

Educational Deficits

Educational deficits have stalled Reconfigurable Computing (RC) as well as classical supercomputing.

Transdisciplinary segmentation: each application domain uses its own trick boxes.

Too many sophisticated very clever architectures.

We need a fundamental model with a methodology which all application domains have in common.

Transdisciplinary education & basic research needed
Most of the enabling technologies of Reconfigurable Computing have been published in the 70ies and 80ies:

This is mainly ignored by the CS community by the tunnel view of a reductionist mind set.

We need to think out of the box: R&D and education need a twin paradigm approach.

Current trends will lead to unaffordable future operation cost of our cyber infrastructure.

Power consumption by internet:

Energy cost may overtake IT equipment cost in the near future.

Power consumption by internet: x30 till 2030 if trends continue

Current trends will lead to unaffordable future operation cost of our cyber infrastructure.

... has become an industry-wide issue: incremental improvements are on track,

but „we may ultimately need revolutionary new solutions“ [Moral Simon, LBNL, Berkeley]

Energy cost may overtake IT equipment cost in the near future.

Current trends will lead to unaffordable future operation cost of our cyber infrastructure.

The ugliness of this term

“Software” stands for extremely memory-cycle-hungry instruction streams.

Patterson’s Law:

bandwidth per core grows 50% a year

has reached >1000x

from earlier talks:

The von Neumann Syndrome:

overhead piles up to code sizes of astronomic dimensions

... has become an industry-wide issue: incremental improvements are on track,

but „we may ultimately need revolutionary new solutions“ [Moral Simon, LBNL, Berkeley]

Energy cost may overtake IT equipment cost in the near future.

Power consumption by internet: x30 till 2030 if trends continue

Current trends will lead to unaffordable future operation cost of our cyber infrastructure.

Multiprocessor computers shift the burden of performance from chip designers to programmers.

... performance drops & other problems in moving single-core to multicore ...

Since People have to write code differently, we anyway need Software Education Revolution ...

... the chance to move RC from niche to mainstream

Embedded syst. & hw scene have the right background to reform the parallelism education of programmers

Missing programmer population, tools and methodology: a scenario like before the Mead & Conway revolution.
A Heliocentric CS Model needed

The Generalization of Software Engineering —
A Twin Paradigm Dual Dichotomy Approach.

FE

Lowware

Program Engineering

special

data streams

subdomain of PE

instructions

streams

A Twin Paradigm Dual Dichotomy Approach.

special

data streams

subdomain of PE

instructions

streams

A Twin Paradigm Dual Dichotomy Approach.

special

data streams

subdomain of PE

instructions

streams

A Twin Paradigm Dual Dichotomy Approach.

special

data streams

subdomain of PE

instructions

streams

A Twin Paradigm Dual Dichotomy Approach.

special

data streams

subdomain of PE

instructions

streams

A Twin Paradigm Dual Dichotomy Approach.

special

data streams

subdomain of PE

instructions

streams

A Twin Paradigm Dual Dichotomy Approach.

special

data streams

subdomain of PE

instructions

streams

A Twin Paradigm Dual Dichotomy Approach.

special

data streams

subdomain of PE

instructions

streams

A Twin Paradigm Dual Dichotomy Approach.

special

data streams

subdomain of PE

instructions

streams

A Twin Paradigm Dual Dichotomy Approach.

special

data streams

subdomain of PE

instructions

streams

A Twin Paradigm Dual Dichotomy Approach.

special

data streams

subdomain of PE

instructions

streams

A Twin Paradigm Dual Dichotomy Approach.

special

data streams

subdomain of PE

instructions

streams

A Twin Paradigm Dual Dichotomy Approach.

special

data streams

subdomain of PE

instructions

streams

A Twin Paradigm Dual Dichotomy Approach.

special

data streams

subdomain of PE

instructions

streams

A Twin Paradigm Dual Dichotomy Approach.

special

data streams

subdomain of PE

instructions

streams

A Twin Paradigm Dual Dichotomy Approach.

special

data streams

subdomain of PE

instructions

streams

A Twin Paradigm Dual Dichotomy Approach.
**Introduction**

- Instruction Stream Machines
- Data Stream Machines
- Hetero Systems
- Time to Space Mapping
- Application Examples
- Outlook

---

**The first Reconfigurable Computer**

- prototyped 1884 by Herman Hollerith
- a century before FPGA introduction
- data-stream-based
- 60 years later the von Neumann (vN) model took over
- instruction-stream-based

---

**Early LUT**

- field-programmable:
  - manually
  - or, by swapping pre-wired plug boards
- non-volatile configuration “memory”

60 years later: RAM available for configuration
But only used for von Neumann machine (at that time)

---

**The History of Mainframes**

- Reconfigurable was first
- Von Neumann had the strongest lobby
- Von-Neumanized hardware design
- Problems with hardware description languages

"I understand everything, although I do not know any electronics"
The software faculty did not support this mindset
But curriculum planners insisted in going through all abstraction levels
Every student had to learn how to design a CPU

---

**Teaching for Change: an early martyr**

- Turing is irrelevant
- "punished for blasphemy?"

The von Neumann model is the emulation of a tape machine
"The von Neumann syndrome": coined ~ a decade later

Critique of von Neumann is not new:

- Prof. C. V. Ramamoorthy, UC Berkeley, EIPS 105, San Diego, CA
- "Why Software is bad…" Peter G. Neumann 1985-2003: Inside Risks (18 years inside back cover of Comm. ACM)
- Antimachine: data counter(s) at memory unit(s)

---

**The von Neumann Paradigm Trap**

CS education got stuck in this paradigm trap which stems from technology of the 1940s
CS education’s right eye is blind, and its left eye suffers from tunnel view
We need a twin-paradigm approach

- Antimachine:
  - data counter(s) at memory unit(s)

---
Instruction Stream Machines

- A very simple von Neumann machine explained at bit level
  - Program sequencer explained, opcode, control structures, loops, simulator, simple example exercises

- Imperative language, compilation, execution on von Neumann simulator, example exercises

- Parallel processes, Hoare, MPI, etc -- extremely simple!!

- A supercomputing example (differential equations?)

- The von Neumann Syndrome

... (old stuff)

Compilation: Software

Software Engineering

Source program
Software compiler
Software code
Instruction schedule (Befehls-Fahrplan) sequential

Imperative Software Language (control part)

<table>
<thead>
<tr>
<th>Language category</th>
<th>Software Languages (instruction streams)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deterministic</td>
<td>Procedural sequencing: traceable, checkable</td>
</tr>
<tr>
<td>Operation sequence driven by:</td>
<td>Read next instruction, goto (instr. addr.), jump (to instr. addr.), instr. loop, loop nesting no parallel loops, escapes, instruction stream branching</td>
</tr>
<tr>
<td>State register</td>
<td>Program counter co-located with datapath</td>
</tr>
</tbody>
</table>

Instruction Stream Machines

- A very simple von Neumann machine explained at bit level
  - Program sequencer explained, opcode, control structures, loops, simulator, simple example exercises

- Imperative language, compilation, execution on von Neumann simulator, example exercises

- Parallel processes, Hoare, MPI, etc -- extremely simple!!

- A supercomputing example (differential equations)

- The von Neumann Syndrome
Instruction Stream Machines

- a very simple von Neumann machine explained at bit level
- program sequencer explained, opcode, control structures, loops, simulator, simple example exercises
- imperative language, compilation, execution on von Neumann simulator, example exercises
- parallel processes, Hoare, MPI, etc -- extremely simple
- a supercomputing example (differential equations)
- The von Neumann Syndrome

Instruction Stream Machines

- a very simple von Neumann machine explained at bit level
- program sequencer explained, opcode, control structures, loops, simulator, simple example exercises
- imperative language, compilation, execution on von Neumann simulator, example exercises
- parallel processes, Hoare, MPI, etc -- extremely simple
- a supercomputing example (differential equations)
- The von Neumann Syndrome
The von Neumann Syndrome

The data-stream-based anti machine approach:

- Has no von Neumann bottlenecks
- Has several von Neumann overhead phenomena

We need a new machine paradigm

A programmer does not understand function evaluation without machine mechanisms - without a program counter...

We urgently need a datastream based machine paradigm

Book Outline

- Introduction
- Instruction Stream Machines
- Data Stream Machines
- Hetero Systems
- Time to Space Mapping
- Application Examples
- Outlook

Compilation: Software vs. Configware

Software Engineering

- Source program
- Software compiler
- Software code

Configuration Engineering

- Source program
- Configuration compiler
- Configware code
- Flowware code

We have 2 choices

Routing the data by memory - cycle - hungry instruction streams thru shared memory.

Data - stream - based: placement of the execution locality ...

Pipe network generated by configware compilation ...

*) or, loading time

*) e.g. complex address computation

Software Engineering

- C, FORTRAN
- MATH LAB, ...

Configuration Engineering

- C, FORTRAN
- MATH LAB, ...

Reconfigurable Computing means ...

- For HPC run time is more precious than compile time
- Reconfigurable Computing means moving overhead from run time to compile time
- Reconfigurable Computing replaces "looping" at run time ...

**) or, loading time

**) e.g. complex address computation

We have 2 choices

Routing the data by memory - cycle - hungry instruction streams thru shared memory.

Data - stream - based: placement of the execution locality ...

Pipe network generated by configware compilation ...

*) before run time

*) or, loading time

*) e.g. complex address computation

Compilation: Software vs. Configware

Source program

- Software code
- Instruction streams

Source program

- Configware code
- Flowware code

Data streams

- Data scheduler

mapper

Configuration compiler

Software compiler

- Data

Source program

Source program

Source program

Source program

Source program

Source program
Having introduced Data streams

H. T. Kung
~1980

triggered

input data stream

Output data streams

DPA

Execution: transport-triggered

No memory wall

output data streams

Systolic array research throughout the 80ies: mainly algebra experts

The Systolic Array
(H. T. Kung paradigm)

introducing Data streams
1980

Nice time/space notation - defines: which data item at which time at which port

DPA

introducing Data streams
no instruction streams

1980

Yielding only linear uniform pipes

array
applications
Pipeline properties
mapping
scheduling
(data stream formation)

regular data dependencies
linear
resources
uniform
linear projection or algebraic synthesis

DPA

Input data stream

Output data streams

DPU operation is transport-triggered

No memory wall

No instruction streams

No message passing

Where were the supercomputing people?
Synthesis Method?

of course algebraic (linear projection)
only for applications with regular data dependencies
Algebra experts caught by their own paradigm trap
The super-systolic array: a generalization of the systolic array

Rainer Kress discarded their algebraic synthesis methods and replaced it by simulated annealing:
rDPA

1995

Legend:
backbus connect
port location marker
operator
super
Systolic array, def of datastream

Generalization of the systolic array

pipe network, organized at compile time

*) supporting non-linear pipes on free form hetero arrays

Generalization’of the systolic array

R. Kress, 1995

array port receiving or sending a data stream

What pipe network?

1DPA = rDPU array, i.e. coarse-grained
rDPU = reconfig. datapath unit (no program counter)

array port receiving or sending a data stream

more generalization

another free form pipe network example: honeycomb architecture

Generalization’of the systolic array

*) supporting non-linear pipes on free form hetero arrays - also spiral, fan, fork and join, and even more crazy routing patterns tolerated

rDPA = rDPU array

Super Pipe Networks

The key is mapping, rather than architecture

array applications pipeline properties mapping scheduling (data stream formation)
systolic array regular data dependencies only linear only uniform only linear projection or algebraic synthesis
super-systolic rDPA no restrictions

1995 (KressArray)

Data Stream Machines

- Systolic array, def of datastream - simple!! -- e.g. Matrix compute) examples intro, simulator, exercises
- introducing the datastream machine
- DMA, auto-sequencing memory, data stream machine exercises programming data streams e.g. of prev. section
- introducing a flowware language, implement a compiler example exercises running on datastream machine
- introductory examples illustrate datastream machine use
von Neumann vs. anti machine

Migration benefit by on-chip RAM

Some RC chips have hundreds of on-chip RAM blocks, orders of magnitude faster than off-chip RAM, so that the drastic code size reduction by software to configure migration can beat the memory wall. Multiple on-chip RAM blocks are the enabling technology for ultra-fast anti machine solutions.

GAGs inside ASMs generate the data streams

GAG = generic address generator
DPA = DPU array
(r)-coarse-grained
DPU: reconfig. datapath
on-chip RAM

Counters: the same micro architecture?

yes, is possible, but for data counters...
... a much better AGU methodology is available*

*) for history of AGUs see Herz et al.: Proc. ICECS 2002, Dubrovnik, Croatia

The counterpart of the von Neumann machine

data counters located at memory (not at data path)

Kress
Kung Anti Machine

Coarse-grained (r)DPA

chip RAM blocks are the enabling technology for ultra-fast anti machine solutions.

<table>
<thead>
<tr>
<th>Machine:</th>
</tr>
</thead>
<tbody>
<tr>
<td>resources</td>
</tr>
<tr>
<td>sequencer</td>
</tr>
</tbody>
</table>

without a sequencer, it's not a machine: so they have missed to define the anti machine paradigm (it's not algebraic)

Winner of the "Not My Job" Award - ADOT
L unofficial Park, AZ 85

It's not our job

Who generates the data streams?

A reductionist approach:

© 2010, Reiner Hartenstein

© 2010, Reiner Hartenstein

Behavior of the Counter

memory bank counter

Programmed by flowware

data streams generated by flowware

Programmed by software

von Neumann bottleneck instruction stream machine (von Neumann etc.)

(anti machine)

data stream machine (anti machine)

von Neumann vs. anti machine

© 2010, Reiner Hartenstein

© 2010, Reiner Hartenstein

DPU program counter

von Neumann instruction stream machine (von Neumann etc.)

ASM: auto-sequencing Memory

asMA: auto-sequencing Memory Array

© 2010, Reiner Hartenstein

© 2010, Reiner Hartenstein

2010, Kaiserslautern

2010, Kaiserslautern

2010, Kaiserslautern

2010, Kaiserslautern
Data Stream Machines

- Systolic array, def of datastream - simple !! (e.g. Matrix compute) examples intro, simulator, exercises
- introducing the datastream machine
- DMA, auto-sequencing memory, data stream machine exercises programming data streams e.g.of prev. section
- introducing a flowware language, implement a compiler example exercises running on datastream machine
- introductory examples illustrate datastream machine use

ASM Data stream generators

use data counters, no program counter

reconfigurable rDPA

implemented by distributed on-chip memory

Systolic array

super systolic pipe network [1980]

(anti-von-Neumann machine paradigm)

ASM: Auto-Sequencing Memory

GAG: 2-D Generic Data Sequence Examples

GAG Slider Operation Demo Example

GAG: 2-D Generic Data Sequence Examples

GAG Slider Operation Demo Example
MoM Scan window (MoMSW) Illustration

MoM architectural primary features:
- 2-dimensional (data) memory address space
- Multiple* vari-size reconfigurable MoMSW scan windows
- MoMSW controlled by reconfigurable GAG (generic address generators)

GAG Slider Model

GAG Generic Address Generator

CGFFT: Parallel Scan Pattern Animation
MoM-3 with 3 vari-size scan windows

Reconfigurable Generic Address Generator

CGFFT:
Generalization of the DMA

MoM architectural primary features:
- 2-dimensional (data) memory address space
- Multiple* vari-size reconfigurable MoMSW scan windows
- MoMSW controlled by reconfigurable GAG (generic address generators)

Reconfigurable Generic Address Generator

Generalization of the DMA

Acceleration factors by:
- address computation without memory cycles
- storage scheme optimization methodology, etc.
- supporting scratch optimization strategies (smart d-caching)

GAG & enabling technology published 1989, survey:
[M. Herz et al.: IEEE ICECS 2003, Dubrovnik]

Patented by TI 1995

Reconfigurable Generic Address Generator

Acceleration Mechanisms by ASM-based MoMSW

MoM Scan windows have the potential to drastically reduce traffic to/from slow off-chip memory.
- No instruction streams needed to implement scratch pad optimization strategies using fast on-chip memory.
- MoMSW Scan windows may contribute to speed-up by a factor of 10 and sometimes even much more.
- MoMSW Scan windows are the deterministic alternative ("d-caching") to (indeterministic and speculative) classical cache usage: performance can be well predicted.
- For data-stream-based computing scan windows are highly effective, whereas classical caches are entirely useless.

Significance of MoMSW Reconfigurable Scan Windows
Data Stream Machines

- Systolic array, def of datastream - simple I -- (e.g. Matrix compute) examples intro, simulator, exercises
- introducing the datastream machine
- DMA, auto-sequencing memory, data stream machine exercises programming data streams e.g. of prev. section
- introducing a flowware language, implement a compiler example exercises running on datastream machine
- introductory examples illustrate datastream machine use

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>procedural sequencing: traceable, checkpointable</td>
</tr>
</tbody>
</table>

- instruction stream languages:
  - read next instruction
  - goto (instruction address)
  - jump to (instruction address)
  - instruction loop
  - instruction loop nesting
  - instruction loop escape
  - instruction stream branching
  - no internally parallel loops

- data stream languages:
  - read next data item
  - goto (data address)
  - jump to (data address)
  - data loop
  - data loop nesting
  - data loop escape
  - data stream branching

Why these paradigms are twins

- imperative Software Languages
  - read next instruction
  - goto (instruction address)
  - jump to (instruction address)
  - instruction loop
  - instruction loop nesting
  - instruction loop escape
  - instruction stream branching
  - no: internally parallel loops

- systolic Flowware Languages
  - read next data item
  - goto (data address)
  - jump to (data address)
  - data loop
  - data loop nesting
  - data loop escape
  - data stream branching
  - yes: internally parallel loops

But there is the Asymmetry for data parallelism
The twin paradigms: different moving of data

Who moves operand to operator if not an instruction?

<table>
<thead>
<tr>
<th>#</th>
<th>moving data between</th>
<th>data transport</th>
<th>execution triggered by</th>
<th>strategy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>von Neumann CPU cores</td>
<td>via common memory</td>
<td>instruction stream</td>
<td>moving data at run time</td>
</tr>
</tbody>
</table>

Von Neumann vs. anti machine

<table>
<thead>
<tr>
<th>#</th>
<th>feature</th>
<th>von Neumann machine</th>
<th>hardwired anti machine</th>
<th>reconfigurable anti machine</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>m’ code schedules</td>
<td>instruction stream</td>
<td>data streams</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td># prog’ sources</td>
<td>1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>source 1</td>
<td>none</td>
<td>configware</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>source 2</td>
<td>software</td>
<td>flowware</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>sequenced by:</td>
<td>program counter</td>
<td>data counter</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>counter co-located with:</td>
<td>CPU</td>
<td>memory block: ASM</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>inter PU communication:</td>
<td>common memory</td>
<td>piped through</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>data meeting PU:</td>
<td>move data at run time</td>
<td>move locality of execution at compile time</td>
<td></td>
</tr>
</tbody>
</table>

Overhead avoided by Anti Machine

<table>
<thead>
<tr>
<th>#</th>
<th>feature</th>
<th>von Neumann machine</th>
<th>hardwired anti machine</th>
<th>reconfigurable anti machine</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>data address computation overhead at run time</td>
<td>instruction stream</td>
<td>overhead</td>
<td>none</td>
</tr>
<tr>
<td>12</td>
<td>data address computation overhead at run time</td>
<td>instruction stream</td>
<td>none</td>
<td>none</td>
</tr>
<tr>
<td>13</td>
<td>inter PU communication overhead at run time</td>
<td>instruction stream</td>
<td>none</td>
<td>none</td>
</tr>
<tr>
<td>14</td>
<td>instruction fetch at run time</td>
<td>instruction stream</td>
<td>none</td>
<td>none</td>
</tr>
<tr>
<td>15</td>
<td>data meet PU at run time</td>
<td>instruction stream</td>
<td>none</td>
<td>none</td>
</tr>
<tr>
<td>16</td>
<td>other overhead</td>
<td>instruction stream</td>
<td>none</td>
<td>none</td>
</tr>
</tbody>
</table>

Which language?

Newton’s 1st Law à la Gordon Bell:

Scientists do not change their direction

Computer scientists haven’t been interested in programming clusters. If putting the cluster on a chip is what excites them, fine.

It will still have to run Fortran!

New Languages for Parallelism?

Efforts to extend standards-based, serial programming languages with features to describe parallel constructs are likely to fail.

What’s more likely to succeed are languages that raise the level of abstraction in algorithm description

Nick Tredennick:

Term Rewriting Systems may raise the abstraction level up to math formulae

Mauricio Ayala-Rincón:

Classical programming languages, but with a slightly different semantics (data-procedural) are good candidates for parallel programming.

Support tools have been demonstrated by academia
What is the reason of the paradox?

**the von Neumann Syndrome**

Result from decades of tunnel view in CS R&D and education

basic mind set completely wrong

"CPU: most flexible platform"?

>1000 CPUs running in parallel: the most inflexible platform

The Law of More: drastically declining programmer productivity

However, FPGA & rDPA are very flexible

Dual paradigm: an old hat (3)

Hardware Description Languages:

"procedure call" or function call

call Module-name (parameters):

Software: time domain

Hardware description: space domain

Why a new machine paradigm ???

The anti machine as the 2nd paradigm

... a Troyan horse to introduce data-stream-based issues to the classical mind set of programmers

Programming by flowware instead of software is very easy to learn (... same language primitives)

Flowware education: no fully fledged hardware expert needed to program configware

Data Stream Machines

- Systolic array, def of datastream -simple !! -- (e.g. Matrix compute) examples intro, simulator, exercises
- introducing the datastream machine
- DMA, auto-sequencing memory, data stream machine exercises programming data streams e.g. of prev. section
- introducing a flowware language, implement a compiler example exercises running on datastream machine
- introductory examples illustrate datastream machine use
... (old stuff)

Hetero Systems

- Mastering the Hardware / Software Chasm
- The Twin Paradigm Approach
- Hetero systems examples, co-compilation, co-design, interfacing (very short intro).
- Softw 2 configw-&-hardw migration:
- Software/configware/hardware partitioning
- The MORPHEUS project

Our Contemporary Computer Machine Model

<table>
<thead>
<tr>
<th>Machine</th>
<th>resources</th>
<th>sequencer</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>property</td>
<td>property</td>
</tr>
<tr>
<td></td>
<td>programming source</td>
<td>programming source</td>
</tr>
<tr>
<td>ASIC accelerator</td>
<td>hardwired</td>
<td>hardwired</td>
</tr>
<tr>
<td>CPU</td>
<td>hardwired</td>
<td>-</td>
</tr>
<tr>
<td>RPU accelerator</td>
<td>programmable</td>
<td>Configware (configuration code)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The clash of Paradigms (a simplified view)

Computation in time
(procedural-only mindset)

CS: Software-only mindset
von-Neumann-based

You can't teach hardware
to a programmer

This is the fault of our CS curricula

von Neumann machine

Machine:
- resources
- sequencer
- software
- program counter
- algorithms

von Neumann machine:
- memory
Anti machine

**Hardwired anti machine:**
- Resources: memory, flowware, algorithms
- Memory, data counters, algorithms

**Reconfigurable anti machine:**
- Resources: memory, flowware, algorithms
- Memory, data counters, algorithms

Dual paradigm: an old hat

**Software mind set:**
- Instruction-stream-based flow chart -> control instructions (FSM: state transition)

**Mapped into a Hardware mind set:**
- Action box = Flipflop, decision box = (de)multiplexer

**Decision box turns into demultiplexer**

**Software mind set:**
- Instruction-stream-based flow chart -> control instructions (FSM: state transition)

**Decision box turns into demultiplexer**

Dual paradigm: an old hat (2)

**Hardware Description Language scene ~1970:**
- "It is so simple! Why did it take 25 years to find out?"
- Because of the reductionists' tunnel view
- Because of a lack of transdisciplinary thinking

Education: a Key Issue

**Programming in CS education is based on fine-grained strictly sequential thinking**

**Manycore programming requires mapping from computing in time, to computing in space & time**

**Decades old methods and simple models are stubbornly ignored**, even by engineers

**Resulting in embarrassing situations**
- (e. g. in talking to your CEO)

Symptom of the von Neumann Syndrome

**A high level R&D manager of a large Japanese IT industry group yielded by single-paradigm mind set**

**Executive summary? Forget it!**

**How about a microprocessor giant having >100 vice presidents?**

**If clause turns into multiplexer**
Coarse-grained Reconfigurable Array

Legend:
- backbus connect
- rout thru only
- rout thru and function (multiplexer)

KressArray Example

If X > Y then swap;
swap turned into a wiring pattern

Really so simple?

(recall this example!)

Embarrassing reaction to Ulrich Nageldinger’s talk at RAW 1996

But you can’t implement decisions!

Embarrassing:
a top level R&D manager of a global IT corp. group completely missing sense of Dichotomies

But you can’t implement decisions!

Software to Configware Migration:

Illustrating, that mono-rail education is fatal
Hetero Systems

- Mastering the Hardware / Software Chasm
- The Twin Paradigm Approach
- Hetero systems examples. Co-compilation, co-design, interfacing (very short intro)
- Softw 2 configw-& hardw migration:
  - Software/configware/hardware partitioning
  - The MORPHEUS project

Domains & what we need

we need paradigm twins

<table>
<thead>
<tr>
<th>term</th>
<th>source for programming</th>
<th>domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software</td>
<td>instruction streams</td>
<td>Time (procedural)</td>
</tr>
<tr>
<td>Configware</td>
<td>resources (structures)</td>
<td>space (structural)</td>
</tr>
<tr>
<td>Flowware</td>
<td>data streams</td>
<td>Time (procedural)</td>
</tr>
</tbody>
</table>

we need data parallelism

Our Contemporary Computer Machine Model

- ASIC
  - hardw.
  - programmed
- CPU
  - hardw.
  - programmed
- DPU
  - configware
  - programmable

The twin paradigm approach

- for legacy software and supervisory tasks
- high performance antimachine paradigm
- efficient von Neumann paradigm
- inefficient von Neumann paradigm
- only intermediate data to be stored: at distributed fast on-chip memory
- heterogenous manycore system

Why a Dichotomy?

Dichotomy = mutual allocation to two opposed domains such, that a third domain is excluded.

The dichotomy model as an educational orientation guide for dual rail education to overcome the software/configware chasm

- Dichotomy of machine paradigms (von Neumann / Anti machine): the „Twin Paradigm“ model
- Relativity Dichotomy: time domain / space domain
  - helps parallelization by time to space mapping
The biggest payoff will come from putting old ideas into practice and teaching people how to apply them properly.

Dichotomy model for dual rail education to overcome the software/configware chasm & the software/hardware chasm

1) Machine Paradigm Dichotomy (von Neumann/Datastream machine): the Twin Paradigm model

2) Relativity Dichotomy: time domain/space domain – helps parallelization by time to space mapping

Paul Dirac predicted a complete anti universe consisting of antimatter

- There are regions in the universe, which consist of antimatter ....
- .... But there are asymmetries
- when a particle hits its antiparticle, both are converted into energy: Annihilation
- We are not aware, that there is a new area in computing sciences, which consists of antimatter of computing
- Reconfigurable Computing is made from this antimatter: data-stream-based computing

- 1956: anti neutron created on Bevatron
- 1928: Paul Dirac: "there should be an anti electron having positive charge" (Nobel price 1933)
- 1932: Carl David Anderson detected this "positron" in cosmic radiation (Nobel price 1936)
- 1954: new accelerators: cyclotron, like Berkeley’s Bevatron
- 1955: Owen Chamberlin et al. create anti proton on Bevatron
- 1956: anti neutron created on Bevatron
- 1965: creation of a deuterium anti nucleus at CERN
- 1995: hydrogen anti atom created at CERN – by forcing positron and anti proton to merge by very low energy.

Anti Matter - machine paradigm:
Anti Atom

The World of Matter - machine paradigm: The Atom

Electron spinning

Position spinning
Matter & Anti-matter of Informatics: Machine and Anti Machine

CPU

Anti Machine paradigm

1936 1st electronic computer (Konrad Zuse)
1946 v. N. machine paradigm
1971 1st RISC processor (Ted Hoff)
1979 data streams: systolic array (Kung/Leiserson)
1990 anti machine paradigm published
1995 DPU: DPSS: supersystolic (Rainer Kress)

Machine paradigm: von Neumann

DPU

all ingredients available

new compilation techniques

Coherent data streams spinning around

© 2010, reiner@hartenstein.de

Matter vs. antimatter: CPU vs. DPU

there are asymmetries

DPU

without sequencer

© 2010, reiner@hartenstein.de

Computer: DPA = DPU array

Heavy anti atoms: DPA = DPU array

© 2010, reiner@hartenstein.de

Parallelism by Concurrency

Independent instruction streams difficult...

© 2010, reiner@hartenstein.de

Compilation for Dual Paradigm Multicore

Juergen Becker’s CoDe-X, 1996

C language source

automatic parallelization by loop transformations

Partitioner

placement and routing

pipeline network

compile to hybrid multicore

© 2010, reiner@hartenstein.de

Hetero Systems

- Mastering the Hardware / Software Chasm
- The Twin Paradigm Approach
- Hetero systems examples: co-compilation, co-design, interfacing (very short intro)
- Softw 2 configw-hardw migration:
  - Software/configware/hardware partitioning
  - The MORPHEUS project

© 2010, reiner@hartenstein.de
Dual Paradigm Application Development

- High level language
- Instruction-stream-based
- Data-stream-based
- Reconfigurable/sofware-accelerator

CoDe-X - C Co-Compiler

Jürgen Becker's CoDe-X, 1996

Jürgen Becker's CoDE-X - C Co-Compiler

The RAM-based domains

- Software domain
  - von Neumann Machine
  - Algorithm: variable
  - Resources: fixed
  - 1 Program Source needed

- Configware domain
  - Flowware
  - Algorithm: variable
  - Resources: variable
  - 2 Program Sources needed

Hetero Systems

- Mastering the Hardware / Software Chasm
- The Twin Paradigm Approach
- Hetero systems examples: co-compilation, co-design, interfacing (very short intro).
- Softw 2 configw & hardw migration:
  - Software/configware/hardware partitioning
  - The MORPHEUS project

Loop Transformation Examples

- Resource parameter driven
- Co-compilation
- Host: reconf.array
  - loop 1-8 trigger endloop
  - loop 9-18 trigger endloop
  - loop 1-2 trigger endloop

Dichotomy Example: Anti-Matter (vs. Matter)

Paul Dirac (1928, Nobel prize 1933):
only accelerator artefacts* (1955 – 1995)
But in CS it's really existing, it's Reconfigurable Computing

Dirac: "But there are Asymmetries" but in CS: only one asymmetry !

* except positron, found from cosmic radiation
Double Dichotomy

1) Paradigm Dichotomy
- Procedure
  - Configuration Domain
- Structure
  - Software Domain

2) Relativity Dichotomy
- Procedure
  - Configuration Domain
- Structure
  - Software Domain

3) Relativity Dichotomy
- Procedure
  - Configuration Domain
- Structure
  - Software Domain

- von Neumann Machine
- Datastream Machine

Often the space dimension is limited (e.g., because of the chip size).
Hetero Systems

- Mastering the Hardware / Software Chasm
- The Twin Paradigm Approach
- Hetero systems examples, co-compilation, co-design, interfacing (very short intro)
- Softw 2 configw-&-hardw migration:
- Software/configware/hardware partitioning
- The MORPHEUS project

... (old stuff)

Hetero Systems

- Mastering the Hardware / Software Chasm
- The Twin Paradigm Approach
- Hetero systems examples, co-compilation, co-design, interfacing (very short intro)
- Softw 2 configw-&-hardw migration:
- Software/configware/hardware partitioning
- The MORPHEUS project

Architektur statt Synchro

- Bessere Architektur statt aufwendiger Synchronisation: Halbierung der Anzahl Blöcke + Auf und Ab der Daten (Shuffle Funktion) - kein von-Neumann-Syndrom!
- Direkte Zeit-Raum-Abbildung: Zugriffs-Konflikte
- Modifikation: mit Shuffle-Funktion „Shuffle Sort“

Zeit zu Raum Abbildung

- Zeit-Domäne: Prozedur-Domäne
  - Programmschleife
    - Bubble Sort
    - Zeit-Algorithmen
  - Pipeline
    - Taktschritt, 1 CPU
  - Shuffle Sort
    - Zeit-Algorithmen

- Raum-Domäne: Struktur-Domäne
  - Pipeline
    - Taktschritt, 1 CPU
  - Shuffle Sort
    - Raum-/Zeit-Algorithmen

Transformationen seit den 70ern

Schleifentransformationen: reichhaltige Methodik publiziert

Hier ein Beispiel:

Zeit-Domäne: Prozedur-Domäne
  - Programmschleife
    - Bubble Sort
    - Zeit-Algorithmen

Raum-Domäne: Struktur-Domäne
  - Pipeline
    - Taktschritt, 1 CPU
  - Shuffle Sort
    - Raum-/Zeit-Algorithmen
time-iterative to space-iterative
Often the space dimension is limited

loop transformation methodology: 70ies and later
  e. g. example: bubble sort migration

We need to POIIP for:
  Software to Hardware Migration
  and
  Software to Configware Migration

2 simple key rules of thumb:
a) loop turns into pipeline
b) decision box turns into demultiplexer

Hetero Systems

- Mastering the Hardware / Software Chasm
- The Twin Paradigm Approach
- Hetero systems examples, co-compilation, co-design, interfacing (very short intro).
- Softw 2 configw-&-hardw migration:
  - Software/configware/hardware partitioning
  - The MORPHEUS project

... (may be)
Application Examples

- survey on speed-up-factor publications automotive, genome sequencing, image processing, what else?
- elevator example or traffic light example
- bubble sort to shuffle sort migration example
- cryptography
- bioinformatics
- genome sequencing
- information science
- automotive
- image processing
- digital signal processing
- ... your proposals ......

Book Outline

- Introduction
- Instruction Stream Machines
- Data Stream Machines
- Hetero Systems
- Time to Space Mapping
- Application Examples
- Outlook

... (old stuff)

thank you for your patience

END
extra pages for discussion:

FPGAs in Supercomputing

- Synergisms: coarse-grained parallelism through conventional parallel processing,
- and: fine-grained parallelism through direct configuration execution on the FPGAs

reconfigurable logic box: 1 Bit

© 2010, reiner@hartenstein.de

Have to re-think basic assumptions

Instead of physical limits, fundamental misconceptions of algorithmic complexity theory limit the progress and will necessitate new breakthroughs.

Not processing is costly, but moving data and messages

We've to re-think basic assumptions behind computing

Avoiding the paradigm shift?

"It is feared that domain scientists will have to learn how to design hardware. Can we avoid the need for hardware design skills and understanding?"

Tarek El-Ghazawi, panelist at SuperComputing 2006

We need a bridge strategy by developing new text books and advanced tools for training the software community

We need a bridge strategy by developing advanced tools for training the software community to think in fine grained parallelism and pipelining techniques.

PISA-MoM

DPLA: fabricated by the E.I.S. Multi University Project:

Mead- & Conway nMOS Design Rules:

256 4-by-4 reference patterns

Mead- & Conway CMOS Design Rules:

>800 4-by-4 reference patterns

vN Software: some reference patterns can be skipped, depending on earlier patterns

MoM: all reference patterns matched in a single clock cycle

Reference patterns automatically generated from Design Rules

© 2010, reiner@hartenstein.de
Acceleration Mechanisms by ASM-based MoMSW

- parallelism by multi bank memory architecture
- reconfigurable address computation - before run time
- avoiding multiple accesses to the same data.
- avoiding memory cycles for address computation
- improve parallelism by storage scheme transformations
- minimize data movement across chip boundaries

ASM: Auto-Sequencing Memory

Acceleration factor: generic address generator GAG for address computation without memory cycles

... partly explaining the RC paradox


Morphware: old stuff

structural programming (non-v-N)

1971 PROMs for small logic
1975 PLA
1978 PAL with PALASM tool
1984 first Xilinx FPGA
meanwhile mainstream ...

We need to POIIP for:

Software to Hardware Migration: and
Software to Configware Migration:

2 key rules of thumb - terrifically simple:

1) loop turns into pipeline [1979] ←
2) decision box turns into demultiplexer

(2) We need to POIIP for:

Software to Hardware Migration: and
Software to Configware Migration:

2 key rules of thumb - terrifically simple:

1) loop turns into pipeline [1979]
2) decision box turns into demultiplexer [1968]: PVOIIP
**POIIP: Loop turns into pipeline [1979]**

- (reconfigurable) DataPath Unit: rDPU
- complex loop body
- nested loops
- complex pipe network inside rDPU

**Brick Wall in the Brain**

立即*一个VIP跳起来：

“你不能实施决策！”

尴尬：全球IT公司集团的顶级R&D经理

完全缺乏“二元性”的感觉

结构与程序

*) 讨论后于奥兰多，FLA的RAW会议

**Computational Density vs. Code Size**

- SPECfp2000/Billion Transistors
- Pentium: 1996: 0.43 MOPS/MHz/Mio Transistors
- 2000: 0.06
- PowerPC 1998: 72
- 2003: 5
- SUN: 1985: 105
- 2000: 30
- 1990: 17
- alpha: 1992: 550
- 1999: 10

- Windows lines of code
  - Windows 95: 15 M
  - Windows 98: 18 M
  - Windows XP ('01): 35 M
  - Windows Vista ('07): 50 M

**ICT is at an inflection point**

- 飞速增长，商业上比相对较小的PC市场更重要

- 未来繁荣取决于网络容量，...，高效软件性能，可扩展平台

- “宽带在转折点非常显著，促使市场治理变化”

- Cowhey’s & Aronson's Law:
  - 便宜的革命：
  - 商务办公室比小得多的PC市场更重要。

**Coarse-grained vs. fine-grained**

<table>
<thead>
<tr>
<th>Device</th>
<th>Granularity</th>
<th>Path Width</th>
<th>Eff. Density</th>
<th>Flexibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>FPGA</td>
<td>fine-grained</td>
<td>~ 1 bit</td>
<td>very low</td>
<td>general purpose</td>
</tr>
<tr>
<td>DPA</td>
<td>coarse-grained</td>
<td>multi bit</td>
<td>very high</td>
<td>specialized</td>
</tr>
<tr>
<td>rDPA</td>
<td>coarse-grained</td>
<td>e.g. 32 bits</td>
<td>high</td>
<td>domain-specific</td>
</tr>
<tr>
<td>Platform FPGA</td>
<td>fine-grained &amp; embedded hw.</td>
<td>mixed</td>
<td>high</td>
<td></td>
</tr>
</tbody>
</table>

MoM
Multiple Scan Windows

MoM anti machine an Xputer architecture

example: 4x4 scan windows

memory bank

smart memory interface

MoM distributed memory

MoM anti machine an Xputer architecture

16 point CGFFT: mapped onto 2-D memory space

CGFFT: Nested and Parallel Scan Pattern

CGFFT: Parallel Scan Pattern Animation

CGFFT: Parallel Scan Pattern Animation

CGFFT: Nested and Parallel Scan Pattern

16 point CGFFT: mapped onto 2-D memory space

CGFFT: Nested and Parallel Scan Pattern

CGFFT: Parallel Scan Pattern Animation

CGFFT: Parallel Scan Pattern Animation

CGFFT: Nested and Parallel Scan Pattern

CGFFT: Parallel Scan Pattern Animation

CGFFT: Nested and Parallel Scan Pattern
The new discipline of (application-specific) distributed memory

Herz et al.: proc. IEEE ICECS 2002

GAG

GAG generic address generator Scheme

GAG Complex Sequencer Implementation

GAG: Address Stepper

GAG: Address Stepper

GAG: Address Stepper

GAG: Address Stepper

GAG: Address Stepper

GAG: Address Stepper

GAG: Address Stepper

GAG: Address Stepper

GAG: Address Stepper

GAG: Address Stepper

GAG: Address Stepper

GAG: Address Stepper

GAG: Address Stepper

GAG: Address Stepper

GAG: Address Stepper
Reiner Hartenstein, TU Kaiserslautern, Germany
http://hartenstein.de

**Generic Address Generator (GAG)**

- **Generalization of the DMA**
- **Acceleration factors by:**
  - address computation without memory cycles
  - storage scheme optimization methodology, etc.

*) Software to Xputer migration

[GAG & enabling technology published 1989, survey:
M. Herz et al. IEEE ICECS 2003, Dubrovnik]

**miscellaneous**

**Dave Patterson’s Curve ……
… is not the only problem**

- Predictions require some history
  - [Gordon Bell]
- Organizations usually behave poorer than anyone can predict
  - [Gordon Bell]
- This especially holds for research funding and politics in Europe
  - [R. H.]

**Future Trends in Microelectronics**

- Predictions require some history
  - [Gordon Bell]
- Organizations usually behave poorer than anyone can predict
  - [Gordon Bell]
- This especially holds for research funding and politics in Europe
  - [R. H.]

**The new paradigm: how the data are traveling**

- better not by instruction execution
- An old hat: transport-triggered + instruction-driven
- pipeline, or chaining
- super systolic array

**Xputer Principles**

- 1984: first FPGAs: very tiny & very expensive
- contemporary?

- reconfigurable address generators
- ASM: Auto-Programmability
- P&R: move locality of operation, not data!
HPC experts coming ...

example: N-body problem going configware ...

Simulation of Star Clusters: x10 speed-up by supercomputer-to-morphware migration (e.g., molecular biology et al.)

Reinhard Maenner, University of Mannheim

HPC pioneer since 1976 (Physics Dept Heidelberg)

Rainer Spurzem, University of Heidelberg

AGI, Astronomisches Rechen-Institut, founded 1700 in Berlin, moved 1945 to Heidelberg by August Kopff

Conclusions (1)

We massively need programmable accelerator co-processors

Established technologies are available and we can still use standard software and their tools

We need a massive Migration of Software to Configware.

The implementation wall: the programmer population's unsustainable skills mismatches

Configware skills and basic hardware knowledge are essential qualifications for programmers.

Conclusions (2)

CS education is a monster!

Fully wrong educational mainstream approaches

Yaw-dropping sclerosis of curriculum taskforces

We need a complete re-definition of CS education

We urgently need a fundamental Research and Education Revolution with connected thinking

We need „une´ Levée en Masses“