Part 2

Outline

• The von Neumann Paradigm
• Accelerators and FPGAs
• The Reconfigurable Computing Paradox
• The new Paradigm
  • Coarse-grained
  • Bridging the Paradigm Chasm
• Conclusions

Nick Tredennick's Paradigm Shifts

Reconfigurable Computing

Nick Tredennick's Paradigm Shifts

Reconfigurable Computing

Configware Compilation

Configware Engineering

Nick Tredennick's Paradigm Shifts

Reconfigurable Computing

resource: variable
algorithm: variable
2 programming sources needed

Nick Tredennick's Paradigm Shifts

Reconfigurable Computing

Configware Compilation

Configware Engineering

Nick Tredennick's Paradigm Shifts

Reconfigurable Computing

Configware Compilation

Configware Engineering

Nick Tredennick's Paradigm Shifts

Reconfigurable Computing

Configware Compilation

Configware Engineering
Data meeting the Processing Unit (PU) ... partly explaining the RC paradox

We have 2 choices
- by Software
- by Configware

pipe network generated by configware compilation

We need a new machine paradigm

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

we need a data stream based machine paradigm

Who generates the Data Streams?

Mathematicians: it's not our job

of course algebraic (linear projection)
only for applications with regular data dependencies
Mathematicians caught by their own paradigm trap

reductionist approach

(it's not our job)

Synthesis Method?

The super-systolic array: a generalization of the systolic array

1995
Rainer Kress discarded their algebraic synthesis methods and replaced it by simulated annealing: iTDA

without a sequencer ...
... it's not a machine

Mathematicians have missed to invent the new machine paradigm
The counterpart of the von Neumann machine

Kress /Kung Anti-Machine

ASM: Auto-Sequencing Memory

Data counters: located at memory (not at data path)

Data counters instead of a program counter

Generic Address Generator (GAG)

Generalization of the DMA

Acceleration factors by:

- Address computation without memory cycles
- Storage scheme optimization methodology, etc.


Apropos HiPEAC: Software / Configware Co-Compilation

Automatic parallelization by loop transformations

CoDe-X, 1996 & language source

Partitioner

SW compiler

CW compiler

CPU

Software / Configware compile

CPU

 accelerators

郭路-106 Reconfigurable Array

Coarse-grained Reconfigurable Array

SNN filter on (supersystolic) KressArray (mainly a pipe network)

No CPU

rDPU reconfigurable Data Path Unit, 32 bits wide

array size: 10 x 16 rDPUs

compiled by Nageldinger's KressArray Aprober

Outline

- The von Neumann Paradigm
- Accelerators and FPGAs
- The Reconfigurable Computing Paradox
- The new Paradigm
- Coarse-grained
- Bridging the Paradigm Chasm
- Conclusions
Symptom of the Software/Configware Chasm

A high level R&D manager of a large Japanese IT industry group.

Executive summary?

Forget it!

How about a microprocessor giant having >100 vice presidents?

Much less deficiencies by coarse-grained

1980  2000  2010

10^6  10^9

0  10  20

Transistors  Microchips

Hartenstein's Law

[1996: ISIS, Austin, TX]

Very compact configuration codes very fast reconfiguration

Reconfigurable Data Path Unit, e.g. 32 bits wide

No CPU

Note: software perspective without instruction streams

Symptom of the Software/Configware Chasm

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

Here is the common model

Microprocessor (also multi core)

Simple FPGA (fine-grained)

Platform FPGA (domain-specific core assortment, embedded in FPGA fabrics)

Coarse-grained reconfigurable array

Reconfigurable instruction set processor

Outline

• The von Neumann Paradigm
• Accelerators and FPGAs
• The Reconfigurable Computing Paradox
• The new Paradigm
• Coarse-grained
• Bridging the Paradigm Chasm
• Conclusions

Hybrid Multi Core

Dual mode multi core

Graphik aus anderen ppt

Copyright © 2007, reiner@hartenstein.de [R.H.]

http://hartenstein.de

IEEE Computer Society EAB November 2005 at Philadelphia
The von Neumann syndrome

Accelerators and FPGAs

The Reconfigurable Computing Paradox

We need a twin paradigm approach

The Gordon Moore gap

The Reconfigurable Computing Paradox

The Pervasiveness of FPGAs

The Reconfigurable Computing Paradox

The von Neumann syndrome

We need a twin paradigm approach

The Reconfigurable Computing Paradox
"simple" FPGAs are only the beginning

- Less discrepancy for platform FPGAs and coarse-grained reconfigurable arrays

Multi Core: Just more CPUs?

Complexity and clock frequency of single-core microprocessors come to an end

Just more CPUs on chip may lead to the dead roads known from supercomputing

Multi-core microprocessor chips emerging: soon 32 cores on an AMD chip, and 80 on an Intel

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
Multi-threading is not the silver bullet
We should not expect the solution from CS

The progress of the joint task force on CS curriculum recommendations is extremely disillusioning

Mastery is feasible only by mutual efforts, like the cooperation between EE and CS as known from the Mead & Conway revolution

For RC the motivations are similarly high: growing cost and looming shortage of energy, and:

The personal supercomputer: a far-ranging massive push of innovation in all areas of science and economy:

by Reconfigurable Computing

CPU vs. DPU

<table>
<thead>
<tr>
<th>memory wall issues</th>
<th>CPU</th>
<th>DPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>instruction fetch overhead</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>address compute overhead</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>synchronization overhead</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>internal data transport</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>caches (also a power issue)</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>

much more saved by coarse-grain

<table>
<thead>
<tr>
<th>platform example</th>
<th>energy W / Gflops</th>
<th>energy factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>MDgrope-3*</td>
<td>0.2</td>
<td>1</td>
</tr>
<tr>
<td>Pentium 4</td>
<td>14</td>
<td>70</td>
</tr>
<tr>
<td>Earth Simulator</td>
<td>128</td>
<td>640</td>
</tr>
</tbody>
</table>

*) feasible also with rDPA

Dead Supercomputer Society

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Illustrating the von Neumann paradigm trap

The data-stream-based approach

The instruction-stream-based approach

von Neumann

<table>
<thead>
<tr>
<th>program counter</th>
<th>execution triggered by</th>
<th>instruction-stream-based</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Terminology

<table>
<thead>
<tr>
<th>term</th>
<th>RAM counter</th>
<th>program counter</th>
<th>instruction fetch</th>
<th>instruction-stream-based</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

© 2007, reiner@hartenstein.de [RH]

IEEE Computer Society EAB November 2005 at Philadelphia
Outline

- The Pervasiveness of FPGAs
- The Reconfigurable Computing Paradox
- The Gordon Moore gap
- The von Neumann syndrome
- The Anti Machine
- We need a twin paradigm approach
- Conclusions

simple FPGAs
coarse-grained arrays
saving energy

Deficiencies of reconfigurable fabrics (FPGA) (fine-grained)

- Transistor density: overhead: physical overhead: reconfiguring overhead:
- Power consumption: overhead: physical overhead: reconfiguring overhead:
- Floorplanning: overhead: physical overhead: reconfiguring overhead:

general purpose: simple FPGA

area inefficiency
deficiency factor: >10,000
power guzzler
slow clock

Dual Paradigm Application Development

Platform-FPGAs, being a predefined mixture of configurable logic cells and powerful, fixed resources.

An alternative: coarse-grained Reconfigurability

- Much more area-efficient
- Much less reconfigurability overhead
- Much higher computational density than CPU (no cache...)

Reconfigurable Computing Paradox

Software perspective, but no program counter

ASM: Auto-Sequencing Memory

- Generalization of the DMA
- Acceleration factor: generic address generator GAG for address computation without memory cycles
- ... partly explaining the RC paradox

Dual Paradigm Application Development

- Software perspective, but no program counter
- More coarse grained arrays
- Much more area-efficient
- Much less reconfigurability overhead
- Much higher computational density than CPU (no cache...)

How many types of application domains?

The Reconfigurable Computing Paradox

DeHon's Law

Simple FPGAs
coarse-grained arrays
saving energy

Platform-FPGA (DSP)

courtesy Xilinx Corp.

Outline

- The Pervasiveness of FPGAs
- The Reconfigurable Computing Paradox
- The Gordon Moore gap
- The von Neumann syndrome
- The Anti Machine
- We need a twin paradigm approach
- Conclusions

simple FPGAs
coarse-grained arrays
saving energy

Deficiencies of reconfigurable fabrics (FPGA) (fine-grained)

- Transistor density: overhead: physical overhead: reconfiguring overhead:
- Power consumption: overhead: physical overhead: reconfiguring overhead:
- Floorplanning: overhead: physical overhead: reconfiguring overhead:

general purpose: simple FPGA

area inefficiency
deficiency factor: >10,000
power guzzler
slow clock

Dual Paradigm Application Development

Platform-FPGAs, being a predefined mixture of configurable logic cells and powerful, fixed resources.
Less discrepancy for platform FPGAs and coarse-grained reconfigurable arrays

- An accidentally discovered side effect

- Software to FPGA migration of an oil and gas application:
  - only a speed-up factor of 17
  - Electricity bill down to <10%
  - Hardware cost down to <10%

- All other publications reporting speed-up did not report energy consumption. - This will change.
What’s Really Going On With Oil Prices?

[BusinessWeek, Jan. 29, 2007]

$52 Price in Feb 2007
[NY Mercantile Exch.: Jan. 17]

$200 Minimum oil price in 2010, in a bet by investment banker Matthew Simmons

Energy as a strategic issue

• Google’s ann. electricity bill: 50,000,000 $
• Amsterdam: 25% goes into server farms
• NY city server farms: 1/4 km$ floor area
• Predicted for the USA in the year 2020: 30-50% of the entire national electricity consumption goes into cyber infrastructure

[Mark P. Mills]

Energy: an important motivation

platform example | Energy: W / Gflops | energy factor
--- | --- | ---
MDgrape-3* (domain specific 2004) | 0.2 | 1
Pentium 4 | 14 | 70
Earth Simulator (supercomputer 2003) | 128 | 640

*) feasible also on reconfigurable platforms

Understanding the Paradox?

“If you were plowing a field, which would you rather use? Two strong oxen or 1024 chickens?” — Seymour Cray

von Neumann chickens? | We must first understand the nature of the paradigm

Outline

• The Pervasiveness of FPGAs
• The Reconfigurable Computing Paradox
• The Gordon Moore gap
• The von Neumann syndrome
• The Anti Machine
• We need a twin paradigm approach
• Conclusions in & the multicore crisis

What is the reason of the paradox?

The Gordon Moore curve does not indicate performance

Moore’s law not applicable to all aspects of VLSI

The peak clock frequency does not indicate performance

astronomic code size causes massive overhead, due to von Neumann syndrome

Outline

• The Pervasiveness of FPGAs
• The Reconfigurable Computing Paradox
• The Gordon Moore gap
• The von Neumann syndrome
• The Anti Machine
• We need a twin paradigm approach
• Conclusions

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

"A leap too far for the existing HPC community"
panelist Allan J. Cantle

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


The von Neumann Syndrome

The instruction-stream-based von Neumann approach:

• has no von Neumann bottle-necks
• has several von Neumann overhead phenomena


The Law of more

• 1000 processors running in parallel means 1000 instruction streams with all their overhead phenomena

Reiner Hartenstein, TU Kaiserslautern, Germany
http://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

Refusing the paradigm shift leads to ...

• an array of overhead phenomena
• waste of researcher capacity on "speculative" methods - the newest: "transactional memory"
• multithreading* is not the silver bullet
• highly disappointing computational density
• the multicore programming crisis
• massive programmer productivity decline
• massive software engineering problems

* is indeterministic

blind on one eye ...

• Most “computer scientists” have mainly ignored the RC break-through
• Curriculum recommendations miss to hit most of the IT job market
• instruction-stream-based only: blind on the other eye....
• ... reductionist tunnel view ...

Von Neumann CPU

<table>
<thead>
<tr>
<th>term</th>
<th>program counter</th>
<th>execution triggered by</th>
<th>paradigm</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>DPU</td>
<td>instruction fetch</td>
<td>instruction-stream-based</td>
</tr>
</tbody>
</table>

Program Source: Software - World of Software - Engineering

Data-stream-based

• in contrast to von Neumann, which is instruction-stream-based, the anti machine is data-stream-based (no instruction fetch at run time)
• Sequencing by one or multiple data counters (each located with an ASM*)
• The history of data streams ......

*) ASM = auto-sequencing memory block
Having introduced Data streams

\[ ~1980 \]

DPA
(pipe network)
no memory wall
execution transport-triggered

input data stream

time

<table>
<thead>
<tr>
<th>port #</th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
</tr>
<tr>
<td>port #</td>
</tr>
</tbody>
</table>

output data streams

\[ \sim 1995 \]

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

of course algebraic (linear projection)
only for applications with regular data dependencies
algebra: the paradigm trap

Who generates the Data Streams?

Mathematicians: it’s not our job

(\textit{it’s not algebraic})

Who generates the Data Streams?

Mathematicians: it’s not our job

(\textit{it’s not algebraic})

The counterpart of the von Neumann machine

crDPA

data counters: located at memory (\textit{not} at data path)

data counters instead of a program counter

von Neumann \textbf{is not} the common model

\begin{itemize}
  \item instruction-stream-based machine
  \item microprocessor age
\end{itemize}
Contrasting machine models

<table>
<thead>
<tr>
<th>term</th>
<th>memory</th>
<th>program counter</th>
<th>execution triggered by</th>
<th>paradigm</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>RAM</td>
<td>instruction</td>
<td>instruction-stream-based</td>
<td></td>
</tr>
<tr>
<td></td>
<td>counter</td>
<td>fetch</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*) "transport-triggered"**) does not have a program counter - no instruction fetch at run time

Early historic machines

von Neumann

- resources: fixed
- algorithm: variable
- 1 programming source needed

Von Neumann

- software
- CPU
- resources: fixed
- algorithm: variable
- 1 programming source needed

Compilation: Software Co-Compilation

- automatic parallelization by loop transformations
- old stuff!

Software / Configware Co-Compilation

- CoDeX, 1996
- C language source
- Partitioner
- SW compiler
- CW compiler
- CPU
- reconfigurable accelerator
- configware age:
- software/configware co-compiler
- CPU
- reconfigurable accelerator
- configware age:
- hardware

Nick Tredennick’s Paradigm Shifts:

- (slowly preparing to use both eyes for a dual paradigm point of view)
- Early historic machines
- Von Neumann
- 1 programming source needed
- configuration
- resources: variable
- algorithm: variable
- 2 programming sources needed

Compilation: Software (von Neumann model)

- Software Engineering
- source program
- software compiler
- software code
- instruction schedule (Befehls-Fahrplan) sequential

Software / Configware Co-Compilation

- automatic parallelization by loop transformations
- old stuff!

Software / Configware Co-Compilation

- CoDeX, 1996
- C language source
- Partitioner
- SW compiler
- CW compiler
- CPU
- reconfigurable accelerator
- configware age:
- software/configware co-compiler
- CPU
- reconfigurable accelerator
- configware age:
- hardware

Nick Tredennick’s Paradigm Shifts:

- (slowly preparing to use both eyes for a dual paradigm point of view)
- Early historic machines
- Von Neumann
- 1 programming source needed
- configuration
- resources: variable
- algorithm: variable
- 2 programming sources needed

Compilation: Software (von Neumann model)

- Software Engineering
- source program
- software compiler
- software code
- instruction schedule (Befehls-Fahrplan) sequential
Hardwired vs. reconfigurable anti machine

- The reconfigurable anti machine needs 2 programming sources:
  - Configware,
  - Flowware
- The hardwired anti machine needs only 1 programming source:
  - Flowware only

Data meeting the Processing Unit (PU)

... partly explaining the RC paradox

We have 2 choices
- routing the data by memory-cycle-hungry instruction streams thru shared memory
- placement of the execution locality ...
- pipe network generated by configware compilations

The Configware Age

- Attempts to avoid the paradigm shift will again create a disaster
- Mainframe age and microprocessor (only) age are history
- We are living in the configure age right now!

Similarity of Programming Language Paradigms

- Instruction stream Languages
  - procedural sequencing, traceable, checkpointable
  - instruction stream branching
- Data stream Languages
  - parallel memory bank access
  - interleave only
  - no restrictions
  - data stream branching

Dave Patterson’s Curve ....

... is not the only problem
The wrong mind set

NYT Aufsatz – Dr. Maric fragen
Was steht in diesem Aufsatz ???
Not general purpose or domain-specific
but reconfigured interconnect instead,
compiled from the particular application
The only enabling technology is coming along with
Reconfigurable Computing and its platform principles

Configware and Flowware ...
... are the sources for programming morphware.

Software is the source for programming traditional
hardwired processors (instruction-stream-driven:
von Neumann machine paradigm and its derivatives)

For Configware and Flowware we prefer the
anti machine paradigm – counterpart of von Neumann

Reconfigurable Computing

Using coarse grain morphware platforms leads to
Reconfigurable Computing, which is really Computing,
whereas physical use of fine grain morphware (FPGAs etc.) means kind of Logic Design on a strange platform.
Importance of binding time

not all switching is done by Configware

Configuration: like a kind of pre-packed frozen-in „super instruction fetch”

---

Not a „dataflow machine”

The anti-machine is not a "dataflow machine"

The so-called "dataflow machine" is driven by arbitration, but not by data streams*

*Their compilers accepting "dataflow language" did not generate data-stream-driven code.

"dataflow machine" operation is indeterministic

anti machine operation is deterministic

anti machine operation is data-stream-driven: execution is transport-triggered by handshake

---

Speed-up Enablers

Hier eine Liste

DRC 4 orders of magnitude

Address computation overhead

Translate into super-systolic rather than into instruction streams

Determine interconnect fabrics by compilation, but not before fabrication

Determine memory architecture by compilation, but not before fabrication

---

IT ages: PC replaced by PS

PC replaced by PS (personal supercomputer)

---

entire system on a single chip

all you need on board

- Xilinx Virtex-II Pro FPGA Architecture
- PowerPC 405 RISC CPU (PPC405) cores
- FPGA Fabric-based on Virtex-II Architecture

*Source: Herbert K. von, M. Econ.
Transdisciplinary Education? in CS: even worse

Lacking *intra*disciplinary cohesion between the mind sets of:

- Theoreticians
- Hardware People:
  - Computer architects
  - Embedded systems des.
- Software People

von Neumann paradigm trap: teaching all students how to design a processor, although most graduates go into embedded systems.

Compilation: Software vs. Configware

Software Engineering

- source program
- software compiler
- software code

Configware Engineering

- C, FORTRAN, MATLAB
- automatic SW/CW partitioner
- mapper
- data
- scheduler

Compilation: Software vs. Configware

Nick Tredennick’s Paradigm Shifts explain the differences

Software Engineering

- CPU: resources: fixed
- software: algorithm: variable
- 1 programming source needed

Configware Engineering

- configure: resources: variable
- flowware: algorithm: variable
- 2 programming sources needed

Co-Compilation

C, FORTRAN, MATLAB

Co-Compiler for Hardwired Kress/Kung Machine

The first archetype machine model

Software Industry

- Software Industry’s Secret of Success
- simple basic
- Machine Paradigm
- instruction-stream-based mind set

Software / Flowware Co-Compiler

- automatic SW/CW partitioner
- compile or assemble
- main frame
- CPU
- „von Neumann”
- RAM-based
The 2nd archetype machine model

Configware Industry

Configware Industry's Secret of Success

Structural personalization

Data-stream-based mind set

Kress-Kung

Reconfigurable accelerator

Simple basic Machine Paradigm

Personalization: RAM-based

The 2nd archetype machine model

Coarse grain is about computing, not logic

SNN filter on KressArray (mainly a pipe network)

No CPU

Reconfigurable data path unit, e.g. 32 bits wide

Compare it to software solution on CPU

S = R + (if C then A else B endif);

Simple conservative CPU example

Read instruction
Read operand
Add & store
Total

C = 1

1

100

100

100

5

100

C = 0

1

100

100

100

5

100

Why the speed-up? What's the difference?

Moving the locality of operation into the route of the data stream by P&R

instead of moving data by instruction streams
The wrong mind set ....

S = R + (if C then A else B endif);
not knowing this solution:
symptom of the hardware / software chasm
and the configuration / software chasm
We need Reconfigurable Computing Education

pre FPGA era: Why DPLA* was so good

Large arrays of canonical boolean expressions
PLA layout – similar to RAM / ROM layout:
Close to Moore because of small overhead (writing, programmability, routing)
Mid’80ies: first very tiny FPGAs available

GAG Generic Address Generator to avoid address computation overhead

Legend: backbus connect

rout thru only
not used
backbus connect

reiner@hartenstein.de
19 February 2009
http://hartenstein.de

(anti-von-Neumann machine paradigm)

ASM: Auto-Sequencing Memory
GAG: Generic Address Generator
RAM: Random Access Memory
data counter

GAG & enabling technology: published 1989 [by TU-KL],
Survey paper: [M. Herz et al.**: IEEE ICECS 2003, Dubrovnik]
patented by TI** 1995

Juergen Becker's CoDe-X, 1996

The effective integration density of FPGAs is behind the Gordon Moore curve by more than 4 orders of magnitude
• wiring overhead
• reconfigurability overhead
• routing congestion
• Low clock frequency
• Power-hungry
• Going worse for larger FPGAs

Remember: speed-ups up to 4 orders of magnitude

© 2007, reiner@hartenstein.de [R.H.]
http://hartenstein.de

TU Kaiserslautern

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

IEEE Computer Society EAB November 2005 at Philadelphia
21
An Example: FPGAs in Oil and Gas .... (1)

Application migration [from supercomputer] has resulted in a 17-to-1 increase in performance

For this example speed-up is not my key issue (Jürgen Becker’s tutorial showed much higher speed-ups - going up to a factor of 6000)

For this oil and gas example a side effect is much more interesting than the speed-up

© 2007, reiner@hartenstein.de

An Example: FPGAs in Oil and Gas .... (2)

Application migration [from supercomputer] has resulted in a 17-to-1 increase in performance

Saves more than $10,000 in electricity bills per year (7¢ / kWh) - .... per 64-processor 19” rack

did you know …

… 25% of Amsterdam’s electric energy consumption goes into server farms ?

… a quarter square-kilometer of office floor space within New York City is occupied by server farms ?

© 2007, reiner@hartenstein.de

Oil and Gas as a strategic issue

Low power design: not only to keep the chips cool

You know the amount of Google’s electricity bill?

It should be investigated, how far the migrational achievements obtained for computationally intensive applications, can also be utilized for servers

Recently the US senate ordered a study on the energy consumption of servers

© 2007, reiner@hartenstein.de

Reconfigurability per se is not the key

It’s the paradigm coming along with it

Note: no instruction fetch at run time !

Data streams instead of instruction streams

Enabling technology for data sequencers brings further performance improvements

A non-reconfigurable example is the BEE project (Bob Broderson et al., UC Berkeley)

© 2007, reiner@hartenstein.de

Petaflops by GRAPE

<table>
<thead>
<tr>
<th>W/Gflops</th>
<th>$/Gflops</th>
</tr>
</thead>
<tbody>
<tr>
<td>Earth Simulator</td>
<td>128</td>
</tr>
<tr>
<td>MDGRAPE-3</td>
<td>0.2</td>
</tr>
<tr>
<td>factor</td>
<td>640</td>
</tr>
</tbody>
</table>

GRAvity PipE: special purpose computer for astrophysical N-body simulations, and, Molecular Dynamics Simulations (coordinates by Makimoto’s wave)

MDGRAPE-3 (aka Protein Explorer): Petaflops-GRAPE (Univ. of Tokyo & Genomic Sciences Center at RIKEN institute)

© 2007, reiner@hartenstein.de

History of Machine Models

<table>
<thead>
<tr>
<th>mainframe age</th>
<th>computer age (PC age)</th>
</tr>
</thead>
<tbody>
<tr>
<td>compile + main frame</td>
<td>compile + Proc. + accel.</td>
</tr>
<tr>
<td>(coordinates by Makimoto’s wave)</td>
<td></td>
</tr>
</tbody>
</table>

MDGRAPE-2 PCI board [1997] 4 chips for N-body simulation converts a PC to 64 GFlops

several Gordon Bell awards

© 2007, reiner@hartenstein.de
History of Machine Models

1957
- Mainframe age
- Compile mainframe
- Procedural mind set: instruction-based
- (coordinates by Makimoto wave)

1967
- Mainframe age
- Compile mainframe
- Procedural mind set: instruction-based

1977
- Mainframe age
- Compile mainframe
- Procedural mind set: instruction-based
- By hardware guys
- Structural mind set: stream-based

1987
- Mainframe age
- Compile mainframe
- Structural mind set: stream-based
- By hardware guys
- Design

1997
- Mainframe age
- Compile mainframe
- Structural mind set: stream-based
- By hardware guys
- Design
- Accelerators
- Microprocessors

2007
- Mainframe age
- Compile mainframe
- Structural mind set: stream-based
- By hardware guys
- Design
- Accelerators
- Microprocessors

Computer age (PC age)

- Compile
- Structural mind set: datastream-based
- By hardware guys
- Design

Reconfigurable Computing (RC)

- Reconfigurable Hardware
  - Functionality (a design)
  - Reconfiguration
  - New functionality (a new design)
- It's RAM-based
- High Flexibility
- High Performance

Reconfiguration

Microprocessors

- Highest Flexibility
- Performance?

ASICS

- Highest Performance
- Lowest Flexibility

Reconfigurable Computing

- It's RAM-based
- High Flexibility
- High Performance

What means Computing?

- The implementation of algorithms...
- Classical computing (software):
  - Instruction-stream-based
- Reconfigurable Computing (RC):
  - Mostly by migration from time to time and space
  - The symbiosis of vN and RC

But for most CS & CE students...

- It's logic design, not computing
- It's hardware design on a strange platform
- This problem is ignored by our CS/CE curricula
- It should be solved by education already at undergraduate level or even earlier
Program Counter
Data Counter(s)

Software / Configware Co-Compiler

C, FORTRAN, MATHLAB

Hardwired anti machine

software
compiler

data
flowware

Makimoto’s Wave
morphware: fastest growing sector of the semiconductor market

von Neumann Syndrome

this term has been coined by “RAM” (C.V. Ramamoorthy, emeritus, UC Berkeley)
>> Flowware <<

- Introduction
- Reconfigurable Computing
- Flowware
- Datastream-based Computing
- The Anti Machine Paradigm
- Final Remarks

http://www.uni-kl.de

© 2007, reiner@hartenstein.de [R.H.]

>> Datastream-based Computing <<

- Introduction
- Reconfigurable Computing
- Flowware
- Datastream-based Computing
- The Anti Machine Paradigm
- Final Remarks

http://www.uni-kl.de

© 2007, reiner@hartenstein.de [R.H.]

Software vs Flowware and Configware

Programming source for instruction-stream-based computing (von Neumann etc.):

Software

The programming source for data-stream-based computing operations (the anti machine paradigm):

Flowware

Programming sources for Reconfigurable Computing (morphware):

Flowware and Configware

Sources for Embedded Systems: Flowware, Configware and Software

http://hartenstein.de

© 2007, reiner@hartenstein.de [R.H.]

The dichotomy of models

- Note for von Neumann: state register is with the CPU
- Note for the anti machine: state register is with memory bank / state registers are within memory banks

http://hartenstein.de

© 2007, reiner@hartenstein.de [R.H.]

Machine paradigms

1) the new discipline came just in time:
see Herz et al.: Proc. IEEE ICECS, 2002
Also see books by Francky Catthoor et al.

http://hartenstein.de

© 2007, reiner@hartenstein.de [R.H.]

Configuration / Flowware Co-Compilation

rDPA
r. Data Path Array

asM

high level source

wrapper

intermediate

mapper

configware

Scheduler

data streams

flowware

address generator

flowware

data streams

http://hartenstein.de

© 2007, reiner@hartenstein.de [R.H.]
von Neumann Syndrome

this term has been coined by "RAM" (C.V. Ramamoorthy, emeritus, UC Berkeley)

>> Outline <<

1. Reconfigurable Computing
2. Tremendous speed-up factors
3. The manycore crisis
4. The von Neumann syndrome
5. Barriers: the education dilemma
6. Twin paradigm dichotomies
7. Conclusions

5. Embarrassing revelations of the hardw./softw. chasm
   ... the configware/software chasm

The collision of paradigms

brick wall in the head only? here? 

"you can always teach programming to a hardware guy"
"you can never teach hardware to a programmer"

[Luigi Dadda]

Caused by our curricula

The CS education dilemma

Nick Tredennick's Perspective

von Neumann Machine:

Software Engineering

CPU

resources: fixed
(also data moved by instruction streams)

Algorithm: variable

1 Program Source needed

Anti Machine:

Configware Engineering

Configware Flowware

resources: variable
(also data streams)

Algorithm: variable

2 Program Sources needed
Nick Tredennick’s Perspective and the Relativity Dichotomy

- Anti Machine:
  - resources: variable
  - algorithm: variable
  - space domain
  - (structure)
- Flowware:
  - time domain
  - (data streams)

Hardware (accelerators) and the Relativity Dichotomy

- Hardware:
  - resources: fixed
  - algorithm: fixed
  - space domain
  - (structure)
- Flowware:
  - time domain
  - (data streams)
  - directly by data streams (anti machine)
  - indirectly by instruction streams for moving data (von Neumann)

Software

- Software Code
  - (instruction-procedural)
  - automatic SW / CW partitioner
- Software Compiler

Software Source

- Software Code
  - (instruction-procedural)
- Software Compiler

Hardware

- Hardware
  - data streams:
    - variable
    - time domain
  - Software
    - (data streams)
    - (structural: space domain)

Configware

- Configware Code
  - (structural: space domain)
- Configware Compiler
  - data scheduler
  - Placement & Routing

Placement & Routing

- mapper

Software to Configware Migration
Coarse-grained rekonfigurable Array

SNN Filter on supersystolic Array: mainly a Pipe Network)

no CPU

reconfigurable Data Path Unit, 32 Bits wide

Array size: 10 x 16 rDPUs
Generated by Nageldinger’s KnapsArray Xplorer (Jürgen Becker’s CoDeX inside)

© 2007, reiner@hartenstein.de (R.H.)

Tu Kaiserslautern

Lehre ohne “connected thinking” zu vermeiden

What is a Dichotomy?

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

The dichotomy model as a didactic orientation guide to overcome the software/configware chasm

Dichotomy of paradigms (von Neumann /Anti machine): the „Twin Paradigm“

Embarrassing: top level R&D manager of a global IT corp. group
We immediately see the brick wall in his brain:
not familiar with very simple old CS wisdom:
completely missing sense of Dichotomies

*) RAW workshop, late 90ies at Orlando, Florida
http://hartenstein.de

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

reiner@hartenstein.de
19 February 2009
(Matter and) Anti-Matter (dichotomy example)

Paul Dirac (1928, Nobel prize 1933): “There are regions in the universe, which consist of antimatter ….

But in CS Antimatter is existing

Reconfigurable Computing is the antimatter of traditional computing

Paul Dirac: “But there are Asymmetries”

1955–1995: Artefacts, synthetized by accelerators

But there is an Asymmetry:

1884 1st data-stream-based computer (Hollerith)
1936 1st instruction stream computer (Konrad Zuse)
1946 von Neumann machine paradigm defined
1971 1st microprocessor (Ted Hoff)
1979 “data streams” def. (systolic array: Kung / Leiserson)
1990 anti machine paradigm published
1995 rDPA / DPSS (supersystolic array: Rainer Kress)

But there is an Asymmetry:

1884 1st data-stream-based computer (Hollerith)
1936 1st instruction stream computer (Konrad Zuse)
1946 von Neumann machine paradigm defined
1971 1st microprocessor (Ted Hoff)
1979 “data streams” def. (systolic array: Kung / Leiserson)
1990 anti machine paradigm published

The Dichotomy: Matter and Antimatter

Reiner Hartenstein, TU Kaiserslautern, Germany

http://hartenstein.de
**Education Revolution: Reconfigurable Computing Revolution**

Application level

Programmer

von-Neumann-Paradigm

(instruction based)

Enrümplung

Anti Machine

Paradigm

(instruction stream)

Enrümplung

The new Mead & Conway?

---

**Who generates the data streams?**

Without a Sequencer it's not a Machine!

---

**Duality of procedural Languages**

*Imperative* Software Languages

read next instruction

goto (instruction address)

jump to (instruction address)

instruction loop

instruction loop nesting

instruction loop escape

instruction stream branching

not: internally parallel loops

**Systolic** Flowware Languages

read next data item

goto (data address)

jump to (data address)

instruction loop

instruction loop nesting

instruction loop escape

instruction stream branching

yes: internally parallel loops

But there is an Asymmetry

---

**Double Dichotomy**

Paradigm Dichotomy

von Neumann

instruction stream

(software-domain)

Anti Machine

data stream

(Flowware-Domain)

Relativity Dichotomy

time

Procedure

(software-domain)

space

Structure

(Configware-Domain)
**Nick Tredennick’s Perspective and the Relativity Dichotomy**

- **Configware**
- **Flowware**
- Anti Machine: resources: variable
- algorithm: variable
- space domain
- (data streams)
- time domain
- (data streams)

**Bus vs. pipe network**

- **Software-Domain:** Bus-based:
  - Memory
  - CPU
  - memory bus = von Neumann bottleneck
- **Configware-Domain:** Pipeline:
  - very simple basic rule
  - Software to Configware Migration: loop turns into pipeline
  - Pipeline: no Memory cycles

**Relativity Dichotomy**

- **time domain:**
  - procedure domain
  - (Machine Dichotomy)
- **space domain:**
  - structure domain
- **time**
- **space**
- 2 phases:
  - 1) programming instruction streams
  - 2) run time
- von Neumann Machine

**Acceleration factors: by what?**

- **n** time steps, 1 CPU
- a time to space mapping
- 1 time step, **n** DPUs
- Software to Configware Migration: loop turns into pipeline
- we all need to become familiar with this.

**time-iterative to space-iterative**

- Often the time dimension (k) is limited (e.g. because of the chip size)
- Elements of a useful methodology of loop transformations have been published since the 70ies.
- The biggest payoff will come from putting old ideas into practice and teaching people to apply them properly.
- [David Parnas]

**time to space mapping**

- **time domain:**
  - procedure domain
- **space domain:**
  - structure domain
- **time**
- **space**
- program loop
  - time steps, 1 CPU
- Pipeline
  - time step, **n** DPUs
  - Bubble Sort
  - n x k time steps, 1 conditional swap unit
- Shuffle Sort
  - n x k time steps, **n** conditional swap units

**Anti Machine:**

- resources: variable
- algorithm: variable
- space domain
- (data streams)
- time domain
- (data streams)
Better Architecture instead of complex synchronisation: half the number of Blocks + up and down of data (shuffle function) – von Neumann syndrome!

Example

Transformations since the 70ies