Part 1

Nick Tredennick's Paradigm Shifts:

Early historic machines
  - Resources: fixed
  - Algorithm: fixed

very early machines:
  - Not really programable

The Hollerith Machine
  - Prototyped 1884 by Herman Hollerith
  - A century before FPGA introduction
  - Data-stream-based

The first Reconfigurable Computer

Hi tec: From plug board to FPGA
Shift back to the von Neumann machine*

-60 years later the von Neumann (vN) model took over

"Mainframe"

* ENIAC: US Army just for computing ballistics tables (early 40ies)

IT ages(1)

mainframe age

IBM

1957
1967
1977
1987
1997
2007

The von Neumann Paradigm

education stubbornly adhered to the absolute monopoly of a model of computation which stems from technology of the 1940s.

- [Burks, Goldstein, von Neumann; 1946]
- RAM (memory cells have addresses ....)
- Program counter (auto-increment, jump, goto, branch)
- Datapath Unit with ALU etc.,
- I/O unit, ....

The von Neumann machine (2)

from tape to RAM

Von Neumann model: emulation of tape
RAM definition: cells have addresses
Program Sequencer def.: Program Counter, increment of jump
Turing Machine!

The von Neumann machine (3)

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>yes</td>
<td>instruction fetch</td>
<td>instruction-stream-based</td>
</tr>
<tr>
<td>DPU</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RAM memory</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Program Source: Software world of Software -Engineering
CPU-centric model

from Kopernicus back to a flat world

Nick Tredennick's Paradigm Shifts:

Early historic machines
resources: fixed
algorithm: fixed

Von Neumann
software
resources: fixed
algorithm: variable
1 programming source needed

Compilation: Software

Software Engineering

source program
software compiler
software code

instruction schedule (Befehls-Fahrplan)
sequential

Paradigm Shifts:
Nick Tredennick's view (1)

algorithms variable
resources fixed

Instruction-stream-based computing:
algorithms: variable
resources: fixed

The Mainframe Age

only a single programming source needed

Instruction schedule (Befehls-Fahrplan)
sequential

4 key issues
(climate change faster than predicted: by carbon emission, primarily from power plants?)
growing computer operating energy cost - and growing number of power plants needed
the manycore programming crisis (end of the free ride on the Gordon Moore curve)
technologically stalled Moore's Law*

wasting energy

growing computer operating energy cost - and growing number of power plants needed
the manycore programming crisis (end of the free ride on the Gordon Moore curve)
technologically stalled Moore's Law*
• 6 – 8 key issues
  • 14 - 16 mini tutorial
  • 21 - 26 vN Syndrome
  • 27 - 30 manycore > RC
  • 31 - 34 speed-up
  • 35 continue beyond Moore

computers’ electricity bill

Energy Cost of Computing

Apropos Google: Nelson Mattos
Climate protection policy

Reconfigurable Computing: much more efficient than any other climate protection effort

factor 10 ? 50% -> 5%
or even less?

Side effect: saving the affordability of our computing infrastructure

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

Introduction

The Manycore Crisis & the von Neumann Syndrome

The Impact of Reconfigurable Computing

Two Dichotomies & Old Ideas

Conclusions

Reconfigurable Computing: much more efficient than any other climate protection effort

Side effect: saving the affordability of our computing infrastructure

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

The Manycore Crisis & the von Neumann Syndrome

The Impact of Reconfigurable Computing

Two Dichotomies & Old Ideas

Conclusions

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

The von Neumann Syndrome

More power for creating foam than to accelerate the vessel?

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

The spirit of the Mainframe Age

For decades, we’ve trained programmers to think sequentially, breaking complex parallelism down into atomic instruction steps…

… finally tending to code sizes of astronomic dimensions

Even in “hardware” courses (unloved child of CS scenes) we often teach von Neumann machine design – deepening this tunnel view

1951: Hardware Design going von Neumann (Microprogramming)
Massive Overhead Phenomena

All this overhead piling up to code sizes of astronomic dimensions

overhead (list not complete)

von Neumann Syndrome (S.V. Raman, R. K. Narayan)

A terrifying number of processes running in parallel, creating sequential processing bottlenecks and losses in data locality

von Neumann Syndrome

Dave Patterson's Law - "Performance" Gap:

growth 50% / year

50% / year

ends in 2005

μProc 60%/yr.

<parallelism>

Monstrous Steam Engines of Computing

Crossbar weight: 220 t, 3000 km of thick cable, 5120 Processors, 5000 pins each

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

reiner@hartenstein.de

19 February 2009
We are in a Computing Crisis

<table>
<thead>
<tr>
<th>platform example</th>
<th>hardw cost $ / Gflops</th>
<th>cost factor</th>
<th>energy W / Gflops</th>
<th>energy factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>MDgrape-3*</td>
<td>15</td>
<td>1</td>
<td>0.2</td>
<td>1</td>
</tr>
<tr>
<td>Pentium 4</td>
<td>400</td>
<td>27</td>
<td>14</td>
<td>70</td>
</tr>
<tr>
<td>Earth Simulator</td>
<td>8000</td>
<td>533</td>
<td>128</td>
<td>640</td>
</tr>
</tbody>
</table>

von Neumann parallelism

The sprinkler head has only a single whole: the von Neumann bottleneck

The watering pot model

Several overhead phenomena

The instruction-stream-based parallel von Neumann approach: per CPU has several von Neumann overhead phenomena

Explosion of overhead by von Neumann parallelism

Reconfigurable Computing means ...

• For HPC run time is more precious than compiletime
• Reconfigurable Computing means moving overhead from run time to compile time
• Reconfigurable Computing replaces "looping" at run time* ...
• ... by configuration before run time

*) e.g. complex address computation

© 2007, reiner@hartenstein.de [R.H.]
Data meeting the Processing Unit (PU) 
... explaining the vN syndrome

路由数据由内存-周期-饥饿的指令流
经由共享内存

the computer age

Benchmarked Computational Density
CPU clock speed ≠ performance: processor's silicon is mostly cache

IT ages (3)

accelerators needed

accelerators needed

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

mainframe age

computer age (PC age)

morphware age

IBM.

Microsoft

von Neumann does not support morphware

1957
1967
1977
1987
1997
2007

© 2007, reiner@hartenstein.de [R.H.]

http://hartenstein.de

IT ages: PC replaced by PS

mainframe age

computer age (PC age)

morphware age

IBM.

Intel

Microsoft

co-compiler

µProc.

rDPA

PC replaced by PS (personal supercomputer)

© 2007, reiner@hartenstein.de [R.H.]

http://hartenstein.de

Outline

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

© 2007, reiner@hartenstein.de [R.H.]

http://hartenstein.de

The clash of paradigms

the basic mind set is
instruction-stream-based

procedural

hammer

the software / hardware chasm

kind of data-stream-based mind set

structural

the software / hardware chasm

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

we need a datastream based machine paradigm

© 2007, reiner@hartenstein.de [R.H.]

http://hartenstein.de

Here is the common model

configware age:

CPU

reconfigurable

accelerator

hardware

accelerator

© 2007, reiner@hartenstein.de [R.H.]

http://hartenstein.de
FPGAs in Supercomputing

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

FPGAs

Number of design starts

ASIC cost

FPGA with island architecture

- 6 - 8 key issues
- 14 – 16 mini tutorial
- 21 - 26 vN Syndrome
- 27 - 30 manycore > RC
- 31 - 34 speed-up
- 35 continue beyond Moore
**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

Paradigm Shifts: Nick Tredennick’s view (1)

The Mainframe Age

- Instruction stream-based computing:
  - Algorithms variable
  - Resources fixed

Software

- Only a single programming source needed

Software Engineering

Paradigm Shifts: Nick Tredennick’s view (2)

Reconfigurable Computing

- Data stream-based reconfigurable computing:
  - 2 programming sources needed

Configware Engineering

- Configware accelerators
- Flowware

Paradigm Shifts: Nick Tredennick’s view (3)

The Morphware Age

- Instruction stream-based computing:
  - Algorithms variable
  - Resources fixed

Software

- 3 programmable sources

Software Engineering

Compilation: Software vs. Configware

- C, FORTRAN, MATLAB

Co-Compilation

Software / Configware Co-Compiler

- Software code
- Configure code
- Flowware code
Code Destinations

- C, FORTRAN, MATHLAB
- Software / Configware Co-Compiler
  - software compiler
  - mapper
  - configware compiler
  - data scheduler
  - configuration
  - Data Downward Code

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

Semiconductor Revolutions
- Makimoto’s Wave
  - “Mainstream Silicon Application is switching every 10 Years”
  - “The Programmable System-on-a-Chip is the next wave”

Impact of Makimoto’s wave
- Software Industry’s Secret of Success
  - Procedural personalization via RAM-based Machine Paradigm
  - Coarse grain RAs

Impact of Data-stream-based...
- Repeat Success Story by new Machine Paradigm!
  - Embedded Hardware/Configware Industry
  - Qualified people are not available

Repeat Success Story by new Machine Paradigm!
- Qualified people are not available
- RAM-based before run time
- structural personalization
## Success Factors

<table>
<thead>
<tr>
<th>Property</th>
<th>Instruction stream based</th>
<th>Data stream based</th>
<th>Reconfigurable</th>
<th>Hardwired</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM-based</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Machine paradigm</td>
<td>yes</td>
<td>available</td>
<td>available</td>
<td>available</td>
</tr>
<tr>
<td>Compatibility</td>
<td>yes</td>
<td>feasible**</td>
<td>feasible**</td>
<td>feasible**</td>
</tr>
<tr>
<td>Scalability</td>
<td>yes</td>
<td>good</td>
<td>good**</td>
<td>good**</td>
</tr>
<tr>
<td>Code relocatability</td>
<td>yes</td>
<td>good**</td>
<td>good**</td>
<td>good**</td>
</tr>
</tbody>
</table>

- For configware industry is missing: - FPGA compatibility - Fully suitable FPGA - Reconfigurable configuration code

- rDPUs and rDPAs do much better than FGAs

---

## Machine Paradigms

### Machine category

<table>
<thead>
<tr>
<th>Computer (the Machine: &quot;von Neumann&quot;)</th>
<th>The Anti Machine</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Driven by:</strong></td>
<td></td>
</tr>
<tr>
<td>Instruction streams</td>
<td>Data streams</td>
</tr>
<tr>
<td>Instruction sequencing</td>
<td>Sequencing data streams</td>
</tr>
<tr>
<td>Engine principles</td>
<td></td>
</tr>
<tr>
<td>Single program counter</td>
<td>Multiple data counter(s)</td>
</tr>
<tr>
<td>Communication path set-up (&quot;instruction fetch&quot;)</td>
<td></td>
</tr>
<tr>
<td>At run time</td>
<td>At load time</td>
</tr>
<tr>
<td>Data resource</td>
<td></td>
</tr>
<tr>
<td>DPU (e.g. single ALU)</td>
<td>DPU or DPA (DPU array etc.)</td>
</tr>
<tr>
<td>Operation</td>
<td></td>
</tr>
<tr>
<td>Sequential</td>
<td>Parallel pipe network etc.</td>
</tr>
</tbody>
</table>

---

## The Impact of Makimoto's Paradigm Shifts

**Configware Industry**

- Software Industry's Secret of Success
- Repetitive Success Story by new Machine Paradigm

- Procedural personalization via RAM-based Machine Paradigm
- Structural personalization before run time

---

## The Digital Divide of Parallelism

### Many application areas:

- Most processors idling
- Less watts per MIPS by orders of magnitude
- Parallelism also instruction-level (ILP) and below

---

## Modes of Operation

### Legend:

- **E ph**: Execution phase
- **C ph**: Configuration phase
- Simple, static reconfigurability

<table>
<thead>
<tr>
<th>Configuration phase</th>
<th>Mode of Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution phase</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

## >> FPGAs <<

- Dynamically reconfigurable
- Higher throughput
- Lower power consumption
illustrating dynamically reconfigurable partially reconfigurable

Swapping and scheduling of relocatable configware code macros is managed by a configware operating system.

Configware OS
- Fundamentally different from software OS
- Reconfigurable Computing at Microsoft

IT ages

mainframe age computer age (PC age) morphware age


- Fine Grain morphware platforms
  - already mainstream: reconfigurable logic just logic design on a strange platform

- Coarse Grain platforms:
  - Reconfigurable Computing:
    - not that new – but shocking the fundamentals of CS curricula
    - an order of magnitude more MIPS/mW than fine grain

speed-up till 3 orders of magnitude

• 6 – 8 key issues
  - 14 – 16 mini tutorial
  - 21 – 26 vN Syndrome
  - 27 – 30 manycore > RC
  - 31 – 34 speed-up
  - 35 continue beyond Moore

• 6 – 8 key issues
  - 14 – 16 mini tutorial
  - 21 – 26 vN Syndrome
  - 27 – 30 manycore > RC
  - 31 – 34 speed-up
  - 35 continue beyond Moore
Flowware instead of software

Flowware provides a (data-)procedural abstraction from the (data-stream-based) structural domain.

... a Trojan horse to introduce the structural domain to the procedural-only mind set of programmers.

The digital divide

Flowware education: no fully fledged hardware expert needed to program embedded systems.

control-procedural vs. data-procedural

The structural domain is primarily data-stream-based:

Flowware ... mostly not yet modelled that way: most flowware is hidden by its indirect instruction-stream-based implementation.

Flowware converts „procedural vs. structural” into „control-procedural vs. data-procedural”...

Flowware defines:

... which data item at which time at which port

Flowware has been pioneered by systolic arrays...

... the hobby of mathematicians throughout the 80ies

---

Generating the Data Streams

Who generates the data streams?

Mathematicians: it’s not our job.

(Flowware is not algebraic)

---

ASM Data stream generators

ASM: Auto-Sequencing Memory

---

Reiner Hartenstein, TU Kaiserslautern, Germany

http://hartenstein.de

reiner@hartenstein.de

19 February 2009
**Legend:**
- backbus connection
- not used
- operator
- and routing port
- location marker

### rDPA with 10 ports for flowware

**SNN filter on KressArray**

Array size: 10 x 16 = 160 rDPUs

Legend: backbus connection

### Mapping algorithms efficiently onto rDPA

**SNN filter on KressArray**

Array size: 10 x 16 = 160 rDPUs

Legend: backbus connection

### Super Pipe Networks

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

- 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></td>
<td>Regular data dependencies</td>
<td>Linear only</td>
<td>Linear projection or algebraic synthesis</td>
</tr>
<tr>
<td>Systolic</td>
<td></td>
<td>Only</td>
<td>Uniform only</td>
<td>(e.g., force-directed) scheduling algorithm</td>
</tr>
<tr>
<td>Supersystolic</td>
<td></td>
<td>Only</td>
<td>No restrictions</td>
<td></td>
</tr>
</tbody>
</table>

*KressArray [1995]*

### Supersystolic Array Principles

- Take systolic array principles
- Replace classical synthesis by simulated annealing
- Yields the supersystolic array
- Generalization of the systolic array
- No more restricted to regular data dependencies
- Now reconfigurability makes sense: use morphware

### Computing paradigms and methodologies

1946: Machine paradigm (von Neumann)
1980: Data streams (Kung, Leiserson)
1989: Anti-machine paradigm
1990: rDPU (Rabaey)
1994: Anti-machine high level programming language
1995: Super systolic array (rDPA)
1996+: SCCC (LANL), SCORE, ASPR, Bee (UCB), ...
1997: Discipline of distributed memory architecture
1997: Configware / software partitioning compiler

### Flowware heading toward mainstream

**Flowware:** managing data streams
**Software:** managing instruction streams

- Data-stream-based computing is heading for mainstream
- 1997: SCCC (LANL) Streams-C Configurable Computing
- SCORE (UCB) Stream Computations Organized for Reconfigurable Execution
- ASPR (UCB) Adopting Software Pipelining for Reconfigurable Computing
- 2000 Bee (UCB), ...
- Most stream-based multimedia systems, etc.
- Many other areas...
Flowware Languages

specialized:
- **Brook**: for modern graphics hardware
- **Streams-C**: defines 1-D streams; generates VHDL
- **DSP-C**: allows to describe key features of DSPs

general purpose:
- **MoPL**: fully supporting the anti machine paradigm - the counterpart of the von Neumann paradigm

---

Programming Language Paradigms

<table>
<thead>
<tr>
<th>Language Category</th>
<th>Von Neumann Languages</th>
<th>Anti Machine Languages</th>
</tr>
</thead>
<tbody>
<tr>
<td>both deterministic</td>
<td>procedural sequencing, traceable, checkable, no parallel loops, escapes, instruction stream branching</td>
<td>procedural sequencing, traceable, checkable, data counters</td>
</tr>
<tr>
<td>operation sequence driven by:</td>
<td>read next instruction, goto (instr. addr.), jump (to instr. addr.), instr. loop, loop nesting</td>
<td>read next data item, goto (data addr.), jump (to data addr.), data loop, data nesting, parallel loops, escapes, data stream branching</td>
</tr>
<tr>
<td>state register program counter</td>
<td>data counter</td>
<td>memory cycle overhead, memory cycle overhead avoided</td>
</tr>
<tr>
<td>Instruction fetch</td>
<td>memory cycle overhead, memory cycle overhead avoided</td>
<td>instruction only, no restrictions</td>
</tr>
<tr>
<td>Parallel memory bank access</td>
<td>interleaving only, no restrictions</td>
<td>interleaving only, no restrictions</td>
</tr>
<tr>
<td>Language features</td>
<td>control flow + data manipulation, (no data manipulation)</td>
<td>data streams only, (no data manipulation)</td>
</tr>
</tbody>
</table>

---

The Secret of Success: Co-Compilation

**High level PL source**

```
SW compiler
```

```
Analyzer / Profiler
```

```
anti machine paradigm
```

```
“vN” machine paradigm
```

```
Resource Parameters
```

**SW Code**

```
Partitioner
```

```
anti machine paradigm
```

```
Supporting different platforms
```

**CW Code**

---

RC

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

IEEE Computer Society EAB November 2005 at Philadelphia
Reiner Hartenstein, TU Kaiserslautern, Germany
http://hartenstein.de

Reconfigurable Computing

Introduction:
1. What means Configware?
2. The impact of Reconfigurable Computing (RC)

what means Configware

Software 2 Configware Migration

Software 2 Hardware Migration

many-core
Feature Implementation Choices

Reconfigurable Logic (RLL) Unit
- Time to Market
- Higher Area
- Lower Area
- Multiple instructions can be mapped to one RLL

Reconfigurable Array of PEs

Two Specialization Examples

Programming Languages

PACT XPP: Reference Module: XPU128 Co-Processor

Full 32 or 24 Bit Design
- 2 Configuration Hierarchies
- Evaluation Board available, and
- XDS Development Tool with Simulator
Discussion at RAW 2000

“But you can’t implement decisions!”

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

19 February 2009

Reiner Hartenstein, TU Kaiserslautern, Germany
http://hartenstein.de
We need 2 machine paradigms

1957
1967
1977
1987
1997
2007

- hardwired
- procedural programming
- structural programming

- algorithm: fixed
- algorithm: variable
- resources: fixed
- resources: variable

- von Neumann machine paradigm
- anti machine paradigm

vN: unbalanced
vN bottleneck
data stream machine:
- bad message: caches do not help
- good message: no vN bottleneck
- caches not needed

rDPA
µProc.

Why a dichotomy of machine paradigms?

The anti machine has no von Neumann bottleneck

Final Remarks

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

All enabling technologies are available

- literature from last 30 years
- languages & (co-)compilation techniques
- anti machine and all its architectural resources
- parallel memory IP cores and generators
- morphware vendors like PACT....
- anything else needed

What's the problem?
The Digital Divide of CS

It's the gap between procedural (instruction-stream-based) and structural (datastream-based) mind set

The von Neumann paradigm's monololy is the main problem
The typical programmer has problems to understand function evaluation without machine mechanisms....
The anti machine as the 2nd paradigm is the key to cope with this problem - for successful curricular innovation

We need long-term managerial commitment on education

>> thank you
Reiner Hartenstein, TU Kaiserslautern, Germany

http://hartenstein.de

---

*** Re-configurable Hardware ?? ***

Terminology has been highly confusing
„Re-configurable Hardware“ ??
this „Hardware“ is not hard !

it’s Morphware

We need a concise terminology:
a consensus is on the way

---

Terminology: DPU versus CPU ...

- DPU: data path unit
- DPA: DPU array
- GA: gate array
- rDPU: reconfigurable DPU
- rDPA: reconfigurable DPA
- rGA: reconfigurable GA

DPU is no CPU: there is nothing central - like in a DPA

---

Digital System Platforms clearly distinguished

<table>
<thead>
<tr>
<th>Platform</th>
<th>Program Source Running on It</th>
<th>Machine Paradigm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware</td>
<td>Fine grain</td>
<td>Configure</td>
</tr>
<tr>
<td>morphware</td>
<td>Course grain</td>
<td></td>
</tr>
<tr>
<td>rDPU, rDPA</td>
<td>Data stream processor</td>
<td>Flowware &amp; configure</td>
</tr>
<tr>
<td>rGA (FPGA)</td>
<td>Instruction stream processor</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Von Neumann machine</td>
</tr>
</tbody>
</table>

---
Glossary

DPU: data path unit
rDPU: reconfigurable DPU
DPA: data path array (DPU array)
rDPA: reconfigurable DPA
ISP: instruction set processor
rAMP: reconfigurable AMP
AMP: data stream processor
rAMP: reconfigurable AMP

Categories of morphware:
- Morphware use
- Granularity (path width)
- Reconfigurable blocks

Configware / Flowware Compilation

- High level source program
- Wrapper
- Intermediate
- Mapper
- Scheduler
- Data flow sequencer
- Address generator
- Data streams
- rDPA
- r. Data Path Array

Time to Market

- A Fundamental Paradigm Shift in Silicon Application
- Revenue
- Product
- Product
- Time to market
- Update 1
- Update 2
- [Tom Kean]

The spirit of the Mainframe Age

- For decades, we’ve trained programmers to think sequentially, breaking complex parallelism down into atomic instruction steps...
- ...finally tending to **code sizes of astronomic dimensions**
- Even in “hardware” courses (unloved child of CS scenes) we often teach von Neumann machine design - deepening this tunnel view
- 1951: Hardware Design going von Neumann (Microprogramming)

Outline

- von Neumann overhead hits the memory wall
- The manycore programming crisis
- Reconfigurable Computing is the solution
- We need a twin paradigm approach
- Conclusions

IEEE Computer Society EAB November 2005 at Philadelphia
von Neumann: array of massive overhead phenomena

<table>
<thead>
<tr>
<th>overhead</th>
<th>von Neumann machine</th>
</tr>
</thead>
<tbody>
<tr>
<td>instruction fetch</td>
<td>instruction stream</td>
</tr>
<tr>
<td>state address computation</td>
<td>instruction stream</td>
</tr>
<tr>
<td>data address computation</td>
<td>instruction stream</td>
</tr>
<tr>
<td>data meet PU</td>
<td>instruction stream</td>
</tr>
<tr>
<td>i/o - i/o from off-chip RAM</td>
<td>instruction stream</td>
</tr>
<tr>
<td>multi-threading overhead</td>
<td>instruction stream</td>
</tr>
<tr>
<td>... other overhead</td>
<td>instruction stream</td>
</tr>
</tbody>
</table>

... piling up to code sizes of astronomic dimensions

von Neumann overhead: just one example

<table>
<thead>
<tr>
<th>overhead</th>
<th>von Neumann machine</th>
</tr>
</thead>
<tbody>
<tr>
<td>instruction fetch</td>
<td>instruction stream</td>
</tr>
<tr>
<td>state address computation</td>
<td>instruction stream</td>
</tr>
<tr>
<td>data address computation</td>
<td>instruction stream</td>
</tr>
<tr>
<td>data meet PU</td>
<td>instruction stream</td>
</tr>
<tr>
<td>i/o - i/o from off-chip RAM</td>
<td>instruction stream</td>
</tr>
<tr>
<td>multi-threading overhead</td>
<td>instruction stream</td>
</tr>
<tr>
<td>... other overhead</td>
<td>instruction stream</td>
</tr>
</tbody>
</table>

... piling up to code sizes of astronomic dimensions

Backus, 1978: Can programming be liberated from the von Neumann style?

Arvind et al., 1983: A critique of Multiprocessing the von Neumann Style

Dave Patterson's Law - "Performance" Gap: ends in 2005

2005: ~1000

CPU clock speed ≠ performance: processor's silicon is mostly cache

better compare off-chip vs. fast on-chip memory

Benchmarked Computational Density

- von Neumann overhead hits the memory wall
- The manycore programming crisis
- Reconfigurable Computing is the solution
- We need a twin paradigm approach
- Conclusions
The Manycore future

- we are embarking on a new computing age -- the age of massive parallelism [Burton Smith]
- everyone will have multiple parallel computers [B.S.]
- Even mobile devices will exploit multicore processors, also to extend battery life [B.S.]
- multiple von Neumann CPUs on the same µprocessor chip lead to exploding (vN) instruction stream overhead [R.H.]

Several overhead phenomena

The instruction-stream-based parallel von Neumann approach:

- per CPU instruction fetch
- has several von Neumann overhead phenomena

Explosion of overhead by von Neumann parallelism

- instruction fetch
- state address computation
- data address computation
- data meet PU
- i/o to/from off-chip RAM
- other overhead

Rewriting Applications

- more processors mean rewriting applications
- we need to map an application onto different size manycore configurations
- most applications are not readily mappable onto a regular array.

- Mapping is much less problematic with Reconfigurable Computing

Disruptive Development

- Computer industry is probably going to be disrupted by some very fundamental changes. [Ian Baron]

- We must reinvent computing. [Burton J. Smith]

- A parallel (vN) programming model for manycore machines will not emerge for five to 10 years. [experts from Microsoft Corp]

- I don’t agree: we have a model.
- Reconfigurable Computing: Technology is Ready, Users are Not
- It’s mainly an education problem
There is something fundamentally wrong with the von Neumann approach.

The reason of this paradox?

The spirit from the Mainframe Age is still alive.

Reconfigurable Computing is the solution.

We need a twin paradigm approach.

Conclusions

RC based parallelism beyond von Neumann Parallelism

The Reconfigurable Computing Paradox

- Bad FPGA technology: reconfigurability overhead, wiring overhead, routing congestion, slow clock speed
- Up to 4 orders of magnitude speedup + tremendously slashing the electricity bill by migration to FPGA
- The reason of this paradox?
- There is something fundamentally wrong in using the von Neumann paradigm
- The spirit from the Mainframe Age is collapsing under the von Neumann syndrome

Recondigurable Computing it’s data-stream-based RC

We need an approach like this:

it’s data-stream-based RC

We need an approach like this:

it’s data-stream-based RC

The Reconfigurable Computing Paradox beyond von Neumann Parallelism

The instruction-stream-based von Neumann approach:

we need an approach like this:

we need an approach like this:

von Neumann overhead vs. Reconfigurable Computing

von Neumann overhead vs. Reconfigurable Computing

- Message passing overhead
- Instruction fetch
- Data address computation
- Inter PU communication

von Neumann overhead vs. Reconfigurable Computing

von Neumann overhead vs. Reconfigurable Computing

- Message passing overhead
- Instruction fetch
- Data address computation
- Inter PU communication

von Neumann overhead vs. Reconfigurable Computing

von Neumann overhead vs. Reconfigurable Computing

- Message passing overhead
- Instruction fetch
- Data address computation
- Inter PU communication

von Neumann overhead vs. Reconfigurable Computing

von Neumann overhead vs. Reconfigurable Computing

- Message passing overhead
- Instruction fetch
- Data address computation
- Inter PU communication
Reconfigurable Computing means ...

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

**) or, loading time  *) e. g. complex address computation

Reconfigurable Computing is the solution used for routing only

ASM

reiner@hartenstein.de

compiled by Nageldinger's KressArray Xplorer (Juergen Becker's CoDe

http://hartenstein.de

2007,

reiner@hartenstein.de

For HPC run time is more

RAM

Conclusions

von Neumann overhead hits the memory wall
The manycore programming crisis
Reconfigurable Computing is the solution
We need a twin paradigm approach

... explaining the RC advantage

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 configure compilation

**) or, loading time  *) before run time

Legend: backbus connect

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 be the memory wall
multiple on-chip RAM blocks are the enabling technology for ultra-fast anti machine solutions

GAGs inside

ASM: generate the data streams

GAG = generic address generator
rDPA = rDPU array, i. e. coarse-grained
rDPU = recoll. datapath itself (no program counter)

Data meeting the Processing Unit (PU)

We have 2 choices

by Software

by Configware

algorithm

(data)

http://hartenstein.de

2007

reiner@hartenstein.de

165

What pipe network?

array port receiving or sending a data stream

pipe network, organized at compile time

depending on connect fabrics

Generalization of the systolic array

[R. Koss, 1989]

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

array size: 10 x 16 = 160 such rDPUs

compiled by Nageldinger's KressArray Xplorer (Juergen Becker's CoDe-X inside)

Reconfigurable Computing means ...

10 x 16 = 160 such rDPUs

array size: 10 x 16 = 160 such rDPUs

by configuration before run time

run time to compile time

means moving overhead from

software

note: kind of software perspective, but without instruction streams + pipelining

32 bits wide

array size: 10 x 16 = 160 such rDPUs

compiled by Nageldinger's KressArray Xplorer (Juergen Becker's CoDe-X inside)

32 bits wide

array size: 10 x 16 = 160 such rDPUs

compiled by Nageldinger's KressArray Xplorer (Juergen Becker's CoDe-X inside)

IEEE Computer Society EAB November 2005 at Philadelphia

28
But we need a dual paradigm approach to run legacy software together with configurables.

Reconfigurable Computing Technology is ready. Users are not?

We need a twin paradigm education

procedural structural

We need new courses

Conclusions
An Open Question

- Coarse-grained arrays: technology ready\(^*\), users not ready
  \(^*\) offered by startups (PACT Corp. and others)

- Much closer to programmer’s mind set:
  really much closer than FPGAs\(^**\)
  \(^**\) “FPGAs? Do we need to learn hardware design?”

- Which effect is delaying the break-through?

please, reply to:

\[ \text{reiner@hartenstein.de} \]

\[ \text{http://hartenstein.de} \]

---

Disruptive Development

The way the industry has grown up writing software - the languages we chose, the model of synchronization and orchestration, do not lead toward uncovering parallelism for allowing large-scale composition of big systems. [Ann Barron]

---

Dual paradigm mind set: an old hat

Software mind set:
- instruction-stream-based:
  flow chart -> control instructions

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

\[ \text{1972: C. G. Bell et al: The Description and Use of Register-Transfer Modules (RTMs), IEEE Trans-C21/5, May 1972} \]
The von Neumann Paradigm

- Program counter (auto-increment, jump, goto, branch)
- Datapath Unit with ALU etc.,
- I/O unit, ....

Von Neumann CPU

Program Source: Software world of Software Engineering

Compilation: Software

Early historic machines

resources: fixed
algorithm: fixed

Von Neumann

CPU

resources: fixed
algorithm: variable

1 programming source needed

Software Engineering
source program

software code

instruction schedule (Befehls-Fahrplan)
sequential
Monstrous Steam Engines of Computing
Crossbar weight: 220 t, 3000 km of thick cable, 5120 Processors, 5000 pins each
larger than a battleship

We are in a Computing Crisis
platform example hardw cost $ / Gflops cost factor energy W / Gflops energy factor
MGrope-3* (domain-specific 2004) 15 1 0.2 1
Pentium 4 400 27 14 70
Earth Simulator (supercomputer 2003) 8000 533 128 640

Energy consumption
• Google's annual electricity bill: $50,000,000
• Amsterdam's electricity: 25% into server farms
• NY city server farms: 1/4 km² building floor area
• Predicted f. USA in 2020: 30-50% of the entire national electricity consumption goes into cyber infrastructure
• petaFlop supercomputer (by 2012 ?): extreme power consumption

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

von Neumann is not the common model
software instruction-stream-based data-stream-based
hardware CPU co-processors

The clash of paradigms
the basic mind set is instruction-stream-based procedural
kind of data-stream-based mind set hardware guy
a programmer does not understand function evaluation without machine mechanisms - without a program counter ... we need a datastream based machine paradigm
Here is the common model

- CPU
- reconfigurable accelerator
- hardwired accelerator

configware age:

DPU

FPGA in Supercomputing

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

- reconfigurable logic box: 1 Bit

FPGAs in Supercomputing

- DataPath Units
  • 32 Bit, 64 Bit

Legend:

Execution phase

Configuration phase

simple, static reconfigurability

Modes of Operation

- configware code loaded from external flash memory, e.g. after power-on (≈ milisecond)

Swapping and scheduling of relocatable configware code macros is managed by a configware operating system

Gliederung

- The von Neumann Paradigm
- Accelerators and FPGAs
- The Reconfigurable Computing Paradox
- The new Paradigm
- Coarse-grained
- Bridging the Paradigm Chasm
- Conclusions
Deficiencies of reconfigurable fabrics (FPGA) (fine-grained)

1996: DeHon's Law

Density: overhead:
FPGA physical overhead
FPGA logical overhead
Cost per gate

Immense area inefficiency
Power guzzler
Slow clock

Areas of success, from high-end systems on earth to mission-critical systems in space.

Reason of the Paradox?

von Neumann chickens?

We must first understand the nature of the paradigm

What is the reason of the paradox?

The von Neumann Syndrome

Moore's law not applicable to all aspects of VLSI

Resulting from decades of tunnel view in arch. R&D and education

Basic mind set completely wrong

'CPU: most flexible platform'?

But >1000 CPUs running in parallel are the most inflexible platform:

The Law of Moore's drastically declining programmer productivity

But FPGA & rDPA are very flexible

An accidentally discovered side effect

- Software to FPGA migration of an oil and gas application:
  - 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, January 29, 2007]

$52 Price of delivery in February 2007

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