Preface:) it's old stuff!

- 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

* this term has been coined by C. V. Ramamoorthy

---

Arthur Schopenhauer

- Arthur Schopenhauer: "Approximately every 30 years, we declare the scientific, literary and artistic spirit of the age bankrupt. In time, the accumulation of errors collapses under the absurdity of its own weight."
- Reiner H.: "Mesmerized by the Gordon Moore Curve, we in computer science slowed down our learning curve. Finally, after 60 years, we are witnessing the spirit from the Mainframe Age collapsing under its own monstrosity – meanwhile laid open by Reconfigurable Computing."

---

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

---

Configware: more compute power than by Software (very cautious estimation)

80% of all (micro)processors are embedded
25% of embedded µProc. are accelerated by FPGA(s)
Every 5th µProc is accelerated by FPGA(s)
average acceleration factor \(>5\)

Conclusion: most compute power comes from Configware

very pessimistic estimation
FPGAs as accelerators found everywhere

Reconfigurable Computing
- Pattern recognition
- Signal processing
- Video, vision
- Radar, sonar
- Wireless communication
- Cryptography
- Music
- Fuzzy logic
- Video
- HD video
- Aerospace
- Automotive
- Multimedia
- Manufacturing
- Image processing

Reconfigurable Scientific Computing
- Artificial intelligence
- Environmental mechanics
- Mathematics
- Petroleum
- Vector
- DNA
- Genetic
- Weather
- Chemistry
- Molecular
- Oil and gas
- Astrophysics
- Fluid dynamics
- Neural network
- Crash simulation

© 2007, reiner@hartenstein.de

Pervasiveness of RC

http://hartenstein.de/pervasiveness.html

One click only per keyword on this list:
- Shows number of hits by google

Mirror: http://www.fpl.uni-kl.de/RCeducation08/pervasiveness.html

© 2007, reiner@hartenstein.de

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

© 2007, reiner@hartenstein.de

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

Density: overhead
- FPGA routing
- Physical area
- FPGA overhead
- Reconfigurability overhead
- Immobile area
- Inefficiency
- Deficiency factor: >10,000
- Power guzzler
- Slow clock

© 2007, reiner@hartenstein.de

Software-to-Configur (FPGA) Migration:
Some published speed-up factors [2003 – 2005]

© 2007, reiner@hartenstein.de

Accelerator SIEMENS Bruchsal

- Speed-up: x 3 000
- 16 FPGAs
- MAC means Multiply and Accumulate
- Tera means 10^12 or 1 000 000 000 000 (1 trillion)
- 1.5 TeraMAC/s
- I/O bandwidth: 50 GByte/s
- Manufactured by SIEMENS at Bruchsal

© 2007, reiner@hartenstein.de

Oil and gas [2005]

“simple” FPGAs are only the beginning

An accidentally discovered side effect

What’s Really Going On With Oil Prices?

Energy as a strategic issue
Energy: an important motivation

<table>
<thead>
<tr>
<th>Platform Example</th>
<th>Energy: $W / Gflops$</th>
<th>Energy Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>MDgrape-3* (domain-specific 2004)</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 (supercomputer 2003)</td>
<td>128</td>
<td>640</td>
</tr>
</tbody>
</table>

*) 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

Executive Summary doesn’t help

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

Moore’s law not applicable to all aspects of VLSI

What is the reason of the paradox?

The Gordon Moore curve does not indicate performance
The peak clock frequency does not indicate performance

Monstrous Steam Engines of Computing

power measured in tens of megawatts,
floor space measured in tens of thousands of square feet

Rapid Decline of Computational Density

primary design goal: avoiding a paradigm shift

dramatic demo of the von Neumann Syndrome

IBM: down by 100 in 6 yrs
HP: down by 20 in 6 yrs
DEC alphas: down by 100 in 6 yrs

Monstrous Steam Engines of Computing

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

larger than a battleship
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"

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:

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

The Law of more

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
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 ...

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

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</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Program Source: Software - World of Software -Engineering

Von Neumann CPU

- 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

Data-stream-based

Having introduced Data streams

~1980

(pipe network) DPA

no memory wall

execution transport-triggered

input data stream

output data streams

1995

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

Synthesis Method?

of course algebraic (linear projection)

only for applications with regular data dependencies

algebra: the paradigm trap

The super-systolic array: a generalization of the systolic array
Who generates the Data Streams?

Mathematicians: it's not our job

~Systolic~

© 2007, reiner@hartenstein.de

Without a sequencer...

(ıt's not our job)

Mathematicians have missed to invent the new machine paradigm... the anti machine

© 2007, reiner@hartenstein.de

The counterpart of the von Neumann machine

vone Neumann is not the common model

© 2007, reiner@hartenstein.de

Here is the common model

Software / Configware Co-Compilation

© 2007, reiner@hartenstein.de
Contrasting machine models

<table>
<thead>
<tr>
<th>Term</th>
<th>CPU program counter</th>
<th>execution triggered by</th>
<th>paradigm</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM</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>
</tbody>
</table>

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

Early historic machines

Von Neumann

resources: fixed
algorithm: fixed

CPU
software

Nick Tredennick’s Paradigm Shifts

Compilation: Software
(von Neumann model)

Nick Tredennick’s Paradigm Shifts

Reconfigurable Computing

Hardwired vs. reconfigurable anti machine

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

Configware Compilation
### Von Neumann vs. anti machine

<table>
<thead>
<tr>
<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>
</tr>
<tr>
<td>#2</td>
<td># prog’ sources</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>#3</td>
<td>source 1</td>
<td>none</td>
<td>configware</td>
</tr>
<tr>
<td>#4</td>
<td>source 2</td>
<td>software</td>
<td>flowware</td>
</tr>
<tr>
<td>#5</td>
<td>sequenced by:</td>
<td>program counter</td>
<td>data counter</td>
</tr>
<tr>
<td>#6</td>
<td>counter co-located with:</td>
<td>PU (data path) CPU memory block</td>
<td>ASM</td>
</tr>
<tr>
<td>#7</td>
<td>inter PU communication:</td>
<td>common memory</td>
<td>piped through</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>
</tr>
</tbody>
</table>

### Overhead avoided by anti machine

<table>
<thead>
<tr>
<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>state address computation overhead at run time</td>
<td>instruction stream</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>
</tr>
<tr>
<td>#13</td>
<td>inter PU communication overhead at run time</td>
<td>instruction stream</td>
<td>none</td>
</tr>
<tr>
<td>#14</td>
<td>instruction fetch at run time</td>
<td>instruction stream</td>
<td>none</td>
</tr>
<tr>
<td>#15</td>
<td>data meet PU at run time</td>
<td>instruction stream</td>
<td>none</td>
</tr>
<tr>
<td>#16</td>
<td>synchronization overhead</td>
<td>instruction stream</td>
<td>none</td>
</tr>
</tbody>
</table>

### Data meeting the Processing Unit (PU)

**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**

**by Software**

**by Configware**

...partly explaining the RC paradox

### 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

### 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
- Register Transfer Modules (DEC: mid 1970ies)
- similar concept: Case Western Reserve Univ.

**An old hat: we just need to accept it**

- Hardware Description Languages;
  - "procedure call" or function call
  - call Module-name (parameters);
  - Software: time domain

- Hardware description: space domain
We need a twin paradigm approach

We do not need a paradigm shift
We must adopt the second paradigm

- We need a duality of 2 cultures:
  - a kind of transdisciplinary approach
  - 1) the instruction-stream-based mind set
    - computing in time (procedural semantics)
  - and 2) the data-stream-based mind set
    - computing in space (structural semantics)

Why the two paradigms are twins

- Both paradigms have the same syntax rules
- Their sequencers use the same circuity
- Their semantics is only slightly different
- But there is an external asymmetry:
  - The location of the counter (with the CPU or with memory)
  - The number of counters: single (program counter), multiple (data counters)

Similarity of Programming Language Paradigms

<table>
<thead>
<tr>
<th>language category</th>
<th>instruction stream Languages</th>
<th>data stream Languages</th>
</tr>
</thead>
<tbody>
<tr>
<td>both deterministic</td>
<td>procedural sequencing</td>
<td>traceable, checkpointable</td>
</tr>
<tr>
<td>operation sequence driven by</td>
<td>goto (in instr. addr.), jump (to instr. addr.), instr. loop, loop nesting, no parallel loops, escape, instruction stream branching</td>
<td>goto (data addr.), jump (to data addr.), data loop, loop nesting, parallel loops, escape, data stream branching</td>
</tr>
<tr>
<td>state register</td>
<td>program counter</td>
<td>data counter(s)</td>
</tr>
<tr>
<td>addressing</td>
<td>massive memory cycle overhead</td>
<td>overhead avoided</td>
</tr>
<tr>
<td>parallel memory bank access</td>
<td>interleaving only</td>
<td>no restrictions</td>
</tr>
</tbody>
</table>

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

Conclusions

- De facto performance of von Neumann computing systems is dramatically behind the expectations from the Gordon Moore curve
- Massive von Neumann parallelism causes a progressive decline of programmer productivity
- Trouble stems from a refused paradigm shift
- Reconfigurable Computing provides improvement by orders of magnitude
- We need a twin paradigm education
- Upgrading CS curriculum recommendations is overdue

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
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 configware age right now!

FPGA experts needed

- Inserat kopieren: FPGA expert saught
- Akute Mangelware

thank you for your patience

Much less deficiencies by coarse-grained

END
Much less deficiencies by coarse-grained area efficiency close to Moore’s configuration code compact. Hartenstein’s Law (1996: ISIS, Austin, TX) (chip abbrev)

Coarse-grained KressArray Family generic Fabrics: a few examples
- Examples of 2nd Level Interconnect: routed through desired function
- Examples of 3rd Level Interconnect: rDPU cell - no separate routing areas!

Coarse-grained Reconfigurable Array
- SNN filter on (supersystolic) KressArray (mainly a pipe network)
- no CPU
- reconfigurable Data Path Unit, 32 bits wide
- array size: 10 x 16 rDPUs
- compiled by Nageldinger’s KressArray Xplorer with Juergen Becker’s Code-X inside

Legend:
- backbus connect
- array size: 10 x 16 rDPUs
- 232 bits wide
- select nearest neighbour (NN) Interconnect: an example
- select mode, number, width of NNports
- without instruction streams: pipelining
- select nearest neighbour (NN) Interconnect: an example
- select mode, number, width of NNports

Note: software perspective