We Need to Reinvent Computing to Avoid its Future Unaffordable Electricity Consumption

We Need to Reinvent Computing to Avoid its Future Unaffordable Electricity Consumption

We Need to Reinvent Computing to Avoid its Future Unaffordable Electricity Consumption

We Need to Reinvent Computing to Avoid its Future Unaffordable Electricity Consumption

better googling with „NSF center for HPRC“

www.chrec.org: the 1st from 48,400 hits

Not only in CHREC the term HPRC means Heterogeneous Systems including both:

• Instruction Stream Parallelism (on manycore etc.)
• Data Stream Parallelism (on accelerators; on FPGAs)

Qualified Programmer Population not existing

Outline

• Coming unaffordable electricity bill
• Massively saving energy by FPGAs
• What’s the problem with FPGAs?
• The parallelism crisis
• We have to re-invent computing
• Conclusions

Outline

• Coming unaffordable electricity bill
• Massively saving energy by FPGAs
• What’s the problem with FPGAs?
• The parallelism crisis
• We have to re-invent computing
• Conclusions
Beyond peak oil

... has become an industry-wide issue: incremental improvements are on track, "we will ultimately need revolutionary new solutions" [Horst Simon, LBNL, Berkeley]

Power Consumption of Computers

Google going to sell electricity

The possibility of computer equipment power consumption spiraling out of control could have serious consequences for the overall affordability of computing. [L. A. Barroso, Google]

Supercomputers are Scientific Instruments

In my opinion, the largest supercomputers at any time, including the first exaflops, should not be thought of as computers. They are strategic scientific instruments that happen to be built from computer technology. Their usage patterns and scientific impact are closer to major research facilities such as CERN, ITER, or Hubble. [Andrew Jones, vice president Numerical Algorithms Group]

no reason to solve the power problem?
Overall affordability of computing

Exa-scale: \(10^{18}\) computations/second): exp. by 2018:
Power estimations (for a single supercomputer):
250 MW = 10 GW (twice NY City w. 16 million people)
[several sources]

Why Computers are important

- Blackout-free electricity generation and distribution,
- Extremo-yield agriculture,
- Safe and rapid evacuation in response to natural or man-made disasters,
- Perpetual life assistants for busy, senior/disabled people,
- Location-independent access to world-class medicine,
- Near-zero automotive traffic fatalities, minimal injuries, and significantly reduced traffic congestion and delays,
- Reduce testing and integration time and costs of complex CPS systems (e.g. avionics) by 1 to 2 orders of magnitude,
- Energy-aware buildings and cities,
- Physical critical infrastructure that calls for preventive maintenance,
- Self-correcting cyber-physical systems for “one-off” applications,
- Disaster Response: Large-Scale Emergency Evacuation,
- Assistive Devices.

Business Information Systems ...
... without computers

Lufthansa anno 1960

Outline

- Coming unaffordable electricity bill
- Massively saving energy by FPGAs
- What’s the problem with FPGAs?
- The parallelism crisis
- We have to re-invent computing
- Conclusions

Potential of RC

Reconfigurable Computing offers an overwhelming reduction of electricity consumption

as well as massive speed-up factors ...

... both by up to several orders of magnitude.

Only Reconfigurable Computing can avoid, that running our infrastructures becomes unaffordable in the future.
**Keynote, FPGA forum: 2-3 March 2010, Trondheim, Norway**

**Table 2: Speed-up by Multiple Scan Windows:**

<table>
<thead>
<tr>
<th>Application</th>
<th>Speed-up factor</th>
<th>Savings</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNA and Protein</td>
<td>8723</td>
<td>779</td>
</tr>
<tr>
<td>sequencing</td>
<td></td>
<td>22</td>
</tr>
<tr>
<td>DES breaking</td>
<td>28514</td>
<td>3439</td>
</tr>
<tr>
<td></td>
<td></td>
<td>96</td>
</tr>
</tbody>
</table>

*RC*: Demonstrating the intensive Impact

<table>
<thead>
<tr>
<th>Application</th>
<th>Speed-up factor</th>
<th>Power</th>
<th>Cost</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNA and Protein</td>
<td>8723</td>
<td>253</td>
<td>22</td>
<td></td>
</tr>
<tr>
<td>sequencing</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DES breaking</td>
<td>28514</td>
<td>1116</td>
<td>96</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Power save factors obtained**

- Energy saving factors: ~10% of speedup
- Low Power Circuit Design: PowerOpt™ (ChipVision Design Systems) reduces power consumption by up to 40%

**RC**: Demonstrating the intensive Impact

- SGI® RASC™ Module (Version 1)
  - Xilinx Virtex II 6000 FPGA
  - Dual QDR DRAM
  - Rack mountable
  - Seamless direct attach to server’s shared memory fabric
  - Datasheet (PDF 145K)

- SGI® RASC™ RC100 Blade
  - Dual Virtex X LX200 FPGA
  - 8GB QDR DRAM
  - Blade or rack-mountable form factor
  - Seamless direct attach to server’s shared memory fabric
  - Datasheet (PDF 137K)

**The Reconfigurability Paradox**

- Lower clock speed
- Wiring overhead
- Reconfigurability overhead
- Routing congestion
reiner@hartenstein.de
10 March 2014

Keynote, FPGA forum; 2-3 March 2010, Trondheim, Norway

The von Neumann Syndrome

Massive Overhead Phenomena

Critique of the von Neumann Model

>40 years Software Crisis

von Neumann overhead vs. Reconfigurable Computing

© 2010, reiner@hartenstein.de

© 2010, reiner@hartenstein.de

© 2010, reiner@hartenstein.de

© 2010, reiner@hartenstein.de
Outline

• Coming unaffordable electricity bill
• Massively saving energy by FPGAs
• What’s the problem with FPGAs?
• The parallelism crisis
• We have to re-invent computing
• Conclusions

FPGA to ASIC design start ratio

Most ASIC design in the world has stopped

3% ASIC

97% FPGA

Revenue down and up

Tools, IP and support

“pure” FPGA abandoned

The programmable logic industry abandoned “pure” FPGAs - a big field of programmable fabric surrounded by IO.

Instead of FPGA fabric in custom SoC designs, we get custom SoC in FPGAs; devices with narrower application focus

Magic mixture of FPGA fabric and hard logic on the same die

Super-flexible ASSP-like devices for optimized hard-core design mixed with FPGA flexibility.

... to capture huge segments of standard parts / ASSP market.

Fab Line Cost

40 billion US$ [IT Business Strategies, Inc.]

Revenue down and up

Tools, IP and support

Xilinx and Altera started FPGA synthesis development

A commercially-viable design flow for FPGA fabric requires years of development and customer experience

Synthesis and place-and-route are P-complete: monstrously complex software - no “magic bullet”.

To develop a robust synthesis and place-and-route tool suite requires years of testing, and fine-tuning
FPGA synthesis advantages

Advantages of FPGA companies:

1) focus on optimizing their tool only for their FPGAs
2) their FPGA synthesis teams influence future FPGA architectures (what EDA synthesis teams could not)
3) earliest possible access and most detailed information about their company’s FPGA architectures
4) regularly access and benchmark to EDA companies’ tools: a known, measurable target to work against.

Will EDA companies survive?

Routing delays: the dominant timing factor, not the logic
FPGA firms interfaced synthesis to routing delay estimations
Feedback from a huge variety of design projects world-wide
Only FPGA firms’ synthesis and place-and-route tools survive?

If EDA abandons FPGA synthesis, smaller FPGA firms are in deep trouble.

Performance Growth by Multicore?

“Multicore shifts the burden of Performance from Chip Designer to Software Developers.”

Performance from Chip Designer & massive programmer productivity problems
much slower than Moore’s law
von-Neumann-only parallelism

vN passes into history

• Suddenly, All Computing Is Parallel: Seizing Opportunity Amid the Clamor - [Michael Wrinn]
• The proud era of von Neumann architecture passes into history - [Michael Wrinn]
• Foundational change will disrupt traditional habits throughout the discipline - [Michael Wrinn]

Outline

• Coming unaffordable electricity bill
• Massively saving energy by FPGAs
• What’s the problem with FPGAs?
• The parallelism crisis
• We have to re-invent computing
• Conclusions

Multimedia in the Multicore Era

Multimedia Performance Needs
application performance needs up to:
Audio 800 MIPS
Graphics 11 GOPS
Video 160 GOPS
Digital TV 900 GOPS

[Courtesy E. Sanchez]

• GSM
• GPRS
• EDGE
• UMTS
• next standard

© 2010, reiner@hartenstein.de
http://hartenstein.de

© 2010, reiner@hartenstein.de
http://hartenstein.de

© 2010, reiner@hartenstein.de
http://hartenstein.de

© 2010, reiner@hartenstein.de
http://hartenstein.de

© 2010, reiner@hartenstein.de
http://hartenstein.de

Keynote, FPGA forum: 2-3 March 2010, Trondheim, Norway
The growing core counts are racing ahead of programming paradigms and programmer productivity

The parallel programming problem has been addressed, in HPC, for at least 25 years. The result: only a small number of specialized developers write parallel code. Multicore becoming ubiquitous, there is some hope that the “if you build it, they will come” [T. Mattson, M. Wrinn]

Also see the list „dead supercomputer society”

The vast majority of HPC or supercomputing applications originally written for single processor with direct access to main memory.

But the first petascale supercomputers employ more than 100,000 processor cores each, and distributed memory. They hope, that dozens of applications are inherently parallel enough to be laboriously decomposed, sliced and diced for mapping onto HPC

Large applications only modestly scalable. >50% apps don’t scale beyond 8 cores, only 6% can exploit >128 PE cores, a tiny fraction 100,000 or more cores.

Some programming languages

Some languages for parallelism

Language wars are religious wars

Humans are quickly overwhelmed by concurrency and find it much more difficult to reason about concurrent than sequential code. Even careful people miss possible interleavings among even simple collections of partially ordered operations.” [Sutter and Larus]

Concurrency in software is difficult because of the abstractions having been chosen.

Adding just new features to existing languages ?

Threads make programs absurdly incomprehensible caused by the wildly nondeterministic nature [E. A. Lee]

Object-oriented limits the visibility of data

Concurrency models can operate at component architecture level rather than programming languages [E. A. Lee]

We need a new Textbook


We've to re-invent computing before writing it

Manycore Programming DUMMIES

having an impact like Mead & Conway

We've to re-write anyway

We've to re-write software anyway (because of manycore)

We've to re-write into configware (part of the software because of the power wall)

For both we have to learn locality awareness

We have to re-invent programmer education

We need a tool flow to support a twin-paradigm approach and locality awareness

We've to re-write anyway (because of manycore)

We've to re-write into configware (part of the software because of the power wall)

For both we have to learn locality awareness

We have to re-invent programmer education

We need a tool flow to support a twin-paradigm approach and locality awareness

A Clean Terminology, please

<table>
<thead>
<tr>
<th>program source</th>
<th>result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software</td>
<td>instruction streams</td>
</tr>
<tr>
<td>Flowware</td>
<td>data streams</td>
</tr>
<tr>
<td>Configware</td>
<td>datapath structures configured</td>
</tr>
</tbody>
</table>

Cray-XD1 Architecture features

The Cray-XD1 allows the Opteron µP to access the FPGA internal registers, internal and external memory.

provides several transfer modes between µP and the FPGA

The µP can read from / write to the FPGA local memory space (i.e., internal registers, internal BRAMs, and external memory).

The FPGA can read from / write to the µP local memory space.

The most bandwidth efficient transfer mode: write-only mode (producer initiates the transfer): burst (for large amount of data) or non-burst.

However, the use of HLL can disable some of these features.

What the programmer should know
What is best for locality awareness?

Some more hardware description languages

+ New Machine Model for FPGAs

Imperative Languages Twins

Too many HDLs

Table 1. Surveyed HLLs

<table>
<thead>
<tr>
<th>Term</th>
<th>Controlled by</th>
<th>Execution Triggered by</th>
<th>Paradigm</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>program counter (at ALU)</td>
<td>instruction fetch</td>
<td>instruction stream</td>
</tr>
<tr>
<td>DPU</td>
<td>data counter(s) (at memory)</td>
<td>data arrival*</td>
<td>data-stream</td>
</tr>
</tbody>
</table>
**A Clean Terminology, please**

<table>
<thead>
<tr>
<th>program source</th>
<th>result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software</td>
<td>instruction streams</td>
</tr>
<tr>
<td>Flowware</td>
<td>data streams</td>
</tr>
<tr>
<td>Configware</td>
<td>datapath structures configured</td>
</tr>
</tbody>
</table>

**Outline**

- Coming unaffordable electricity bill
- Massively saving energy by FPGAs
- What's the problem with FPGAs?
- The parallelism crisis
- We have to re-invent computing
- Conclusions
**key issues**

- The massive power consumption of computers.
- Hetero systems needed (vN + non-v-N accelerators).
- Accelerators: hardwired + RC + (soon?) ANN
- Productivity programmer population missing.
- Productivity programming methodology missing
- 2 scenes: eRC (embedded) + HPRC (supercomputing)

---

**Potential of RC**

- Reconfigurable Computing offers an overwhelming reduction of electricity consumption
- as well as massive speed-up factors ...
- ... both by up to several orders of magnitude.
- Only Reconfigurable Computing can avoid, that running our infrastructures becomes unaffordable in the future.
- We have to re-invent computing as soon as possible

---

**We need „une‘ Levée en Masses.**

---

**Taxonomy of Twin Paradigm Programming Flows (HPRC)**

- "...Adoption of VHDL was one of the biggest mistakes in the history of design automation, costing users and EDA vendors to waste hundreds of millions of dollars..." -- Joe Costella, Cadence Design Systems, 1995
- "The [n]'off of EDA" [P.N.]

---

**Locality awareness is essential for flowware**

- Software: by addresses, read from instruction
- Flowware: by wire (configured before run time)

---

**END**
**Reinvent? (final remark)**

rediscovery and revival of old ideas
rearrange and teach them properly
avoid traditional tunnel views
to obtain new perspectives
to reach promising new horizons

---

**time to space mapping**

**Time domain:**
- procedure domain
  - program loop
  - Bubble Sort
  - time algorithm

**Space domain:**
- structure domain
  - pipeline
  - space algorithm

- time algorithm
  - space/time algorithm

**Architrecture instead of synchro**

---

**Transformations since the 70ies**

**Time domain:**
- procedure domain
  - program loop
  - n x k time steps, CPU

**Space domain:**
- structure domain
  - Pipeline
  - time steps, n CPUs

---

**Outline**

- Never run out of energy?
- Energy consumption: unaffordable soon?
- The many-core crisis
- Rescue by Reconfigurable Computing?
- We need to Reinvent Computing
- Conclusions
Development with VHDL is expensive
FPGAs Achilles' Heel is their long development time
Low level HDLs (VHDL/Verilog) are still dominant
Unlike software, FPGAs do not offer forward/backward compatibility
FPGAs: low technology maturity; small user base, compared to software
[ quoted from P. Miller, Univ. of Arizona]
Complicated IP Core Scene
Variety of Tool Flows
FPGA eASIC ASIC
Survey urgently needed: Manpower required!
TSMC creates a 'soft' IP core collaboration program to improve soft IP readiness for EDA and IP suppliers including Atanix, Mentor, Cadence, Chip's and Metra. Integration Technologies, Integrated, MIPS Technologies, Sona, Synopsys, and several others.

Productivity vs. Efficiency
HDLs: zero ease of use !!!
[ Adoption of VHDL was one of the biggest mistakes in the history of design automation, causing new and ELA's vendors to waste hundreds of millions of dollars. ]
[ quoted from K. Benkrid, Caltech Design Systems, 1995 ]

The proof of EDA [K. N.]
how to hide the ugliness from the user [Herman Schmit]
Fig. 3. Productivity graph of HDLs

Understanding Complex Hetero Systems
We must change how programmers think
Internode Communications reduces Computational Efficiency
Understanding streams through complex fabrics needed
Efficient Distribution of Tasks being memory limited
Focusing on memory mapping issues and transfer modes to detect overhead and bottlenecks
Layers of Abstraction and Automatic Parallelization hide
critical sources of, and limits to efficient parallel execution
essential: awareness of locality,

Processor inside FPGA vs. FPGA inside Processor: EPP
Xilinx: Extensible Processing Platform™
- totally changed concept
device more like heterogeneous SoCs:
significant benefits for HPC applications:
not hardware centric:
FPGAs became software-centric:
- EDUCATION !!!

Structured ASICS
confiware on ROM
Structured ASICS like eASIC are based mostly on FPGA-like architecture with special configuration mechanism to program at mask level: not re-programmable (more performance for less cost).

RTL Programming to ASIC / ASSP
Platform-Language of Silicon: attractive for IP providers
Application Tools-Path to ASICs/ASSPs, across FPGAs
Specific Standard RTL is inherently parallel, mapped application are automatically optimally parallelized by CAD tools.
now ESL (Electronic System Level) bridging HDL and ANSI C/C++
at industrial level)

Battle between FPGAs vs MPSoC:
it is RTL vs Software programming.
Keynote, FPGA forum; 2-3 March 2010, Trondheim, Norway

15
The Language and Tool Disaster

- Software people don’t speak VHDL
- Hardware people don’t MPI nor OpenMP
- Bad quality application development tools
- 86% designers hate their tools [FCCM’98]
- Comprehensibility barrier between procedural and structural mind set

Productivity Semantic Gap

- Productivity = f (time-to-solution) = f (development time, execution time)
- Productivity Problem – Semantic Gap

Some Acceleration Mechanisms

- Accelerate tasks by streaming
  - MISD structured computation: streaming computations across a long array before storing results in memory
  - Can achieve 100x in improved use of memory
  - parallelism by multi bank memory architecture
  - auxiliary hardware for address calculation
  - address calculation before run time
  - avoiding multiple accesses to the same data
  - optimizing address computation by storage scheme transformations
  - optimization by memory architecture transformations