Toward Reconfigurable Computing via Concussive Paradigm Shifts

Predictions require some history [Gordon Bell]

Future Trends in Microelectronics … (theme of this workshop)
- You need history to predict [Gordon Bell]
- 1954: NTG founded
- 1955: Karl Steinbuch coins the term “Informatik”
- 1958: Karl Steinbuch’s Institute founded at TH Karlsruhe
- 1969: GI foundation completed (Gesellschaft für Informatik)
- 1969: 1st German University program f. Informatik started at TH Karlsruhe

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

The European Paradigm Shift Paradox
- When major funding agencies and their prominent advisors exclude an area, it becomes mainstream (e.g. elsewhere)
  - the EU commission rejected all consortia applications on funds for research in Reconfigurable Computing [2003]
  - DFG (German National Science Foundation) opens a program on Reconfigurable Computing (June 2003 at Stuttgart-Untertürkheim - initiative by Jürgen Teich)

Paradigm Shift to the Computer Age
- Ken Olson, 1977: “There is no reason anyone would want a computer in their home”
- Heinz Nixdorf: “We won’t switch from Mercedes to Bicycle”

Paradigm Shift a la Mead & Conway
- Lobbying for funding Mead-å-Conway-type multi university projects [1980 - 1983]: declined by all European national governments
- German technology professor (German R&T minister’s chief advisor) 1981: “Carver Mead is a Charlatan” by the new minister of R&T: Heinz Riesenhuber
- German Federal elections in 1983: “die große Wende” by the new minister of R&T: Heinz Riesenhuber
- Similar programs fall in France and some other European countries
VLSI Design: no more „EE only”

Beginning „design sciences” as its own discipline

EDA industry: soon first start-ups

Mead & Conway: the hot topic everywhere

Paradigm shifts and German side effects

- NTG (German communication technology society) in 1986: renames itself** ITG (Information Technique Society)
  
- German Univ. department names: EE turning IT, later EIT

- Settlement commission founded [1987], results:
  
- GI Informatik-Spektrum, June 1991

- More CS people start working on VLSI and EDA

- The GI (Gesellschaft für Informatik) reorganizes* (1986 – 1987) and founds a SIG on VLSI architecture and other VLSI/EDA-related SIGs

*) Professors Krückeberg (GI president), Grass, Hartenstein later EIT

**) [NTZ 39 / 9, 1986]

Beethoven a mathematician?

no, only a journalist's effort to make title appear to be more crisp

Math expanding into biology?

EDA shifting to CS mentality

- The impact of Makimoto’s Wave
- Reconfigurable Computing needed
- Datastream-based Computing
- CS curricula update is overdue
CS notations for EDA

- Microprogramming to replace FSM design
- Hardware languages replace EE-type schematics
- EDA Software and ist interfacing languages
- Newer system level languages like systemC etc.
- Hierarchical organization of designs, EDA, et. al.

EDA industry shifts into CS mentality

- patches instead of engineering
- innovation stalled many years ago
- netlist-based: do not care about efficiency, ...
- ... do not care about transistor density

Improving RTL-only design cost model

EDA shifting to CS mentality
- The Impact of Makimoto’s Wave
- Reconfigurable Computing needed for SoC
- Reconfigurable Computing needed for µP
- Datastream-based Computing
- CS curricula update is overdue

Semiconductor Paradigm Shifts

- “Mainstream Silicon Application is switching every 10 Years”
- “The Programmable System-on-a-Chip is the next wave”

terminology

soft hardware?
morphware
IT ages

- Mainframe age
- Computer age (PC age)
- Morphware age

Migration of programming to the structural domain
The structural domain has become RAM-based
The next fundamental revolution after introduction of the microprocessor

Reconfigurable Computing: a second programming domain

Reconfigurable Computing: not that new – but shocking the fundamentals of CS curricula

Throughput vs. Flexibility

Wide variety of speed-up factors

PACT XPP: Reference Module: XPU128 Co-Processor

Platform | Application example | Speed-up factor | Method
---|---|---|---
PACT Xtreme 4-by-4 array (2003) | 16 tap FIR filter | x16 MOPS/mW | Straight forward
MoM anti machine with DPLA* (1983) | grid-based DRC** | > x1000 (computation time) | Multiple aspects

© 2003, reiner@hartenstein.de
http://hartenstein.de
Reconfigurable Computing needed for SoC

- EDA shifting to CS mentality
- The impact of Makimoto’s Wave
- Reconfigurable Computing needed for SoC
- Reconfigurable Computing needed for µP
- Datastream-based Computing
- CS curricula update is overdue

Embedded System Design Crisis

the 2nd design crisis

Kaiserslautern University of Technology

Mask & NRE cost

[ST microelectronics]

Foundries: Adoption Rate By Process

[Nick Tredennick]

Ubiquitous Embedded Systems

Embedded System Engineering (ESE) requires:

- Hardware (HW) / ESoftware (ESW) co-design
- Configware (CW) / ESW co-design
- HW / CW / ESW co-design

ESW and CW become main vehicle
to product differentiation
ESE and CW become the main focus in system design

SoC System level Design

Embedded SW (ESW) and CW

(Performance and) Flexibility are key issues

new design automation from high level descriptions

HW-(E)SW codesign onto highly programmable platforms (SoC)

formal verification for (E)SW and CW
Reconfigurable Computing needed for μP

- EDA shifting to CS mentality
- The impact of Makimoto’s Wave
- Reconfigurable Computing needed for SoC
- Reconfigurable Computing needed for μP
- Datastream-based Computing
- CS curricula update is overdue

“Pollack’s Law” (simplified)

growth factor
area efficiency
performance

MPU designs more complex

- new kinds of concurrency are becoming important
- chip-level multiprocessing + simultaneous multithreading
- many bugs relate to concurrency issues
- greatly complicates the verification process

Steroids for the aging microprocessor:

The Impact of Reconfigurable Computing

PC replaced by PS

PC replaced by PS (personal supercomputer)
• EDA shifting to CS mentality
• The impact of Makimoto’s Wave
• Reconfigurable Computing needed for SoC
• Reconfigurable Computing needed for µP
• Datastream-based Computing
• CS curricula update is overdue

Paradigm Shifts: Nick Tredennick’s view

why 2 program sources?

Instruction-stream
reconfigurable computing:
algorithms variable
resources fixed

Datastream-based computing:
algorithms variable
resources variable

Software
Configware
Flowware

Flowware heading toward mainstream

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

Flowware: mostly not yet modelled that way; most flowware is hidden by its indirection:

Instruction-stream-based implementation

Computing paradigms and methodologies

1946: machine paradigm (von Neumann)
1980: data streams (Kung, Leiserson)
1989: anti-machine paradigm
1990: rDPU (Robey)
1994: anti-machine high level programming language
1995: super systolic rDPA
1996+: SCCC (LANL), SCORE, ASPCR, Bee (UCB), ...
1997+: discipline of distributed memory architecture
1997: configure / software partitioning compiler
Programming Language Paradigms

<table>
<thead>
<tr>
<th>Language Category</th>
<th>Computer Languages</th>
<th>Languages f. Anti Machines</th>
</tr>
</thead>
<tbody>
<tr>
<td>Both deterministic</td>
<td>Both sequential</td>
<td>Both sequential</td>
</tr>
<tr>
<td>Operation sequence driven by:</td>
<td>Read next instruction, go to base addr., count loop nesting, parallel loops, escapes, data stream branching, program counter</td>
<td>Read next data item, go to data addr., count loop nesting, parallel loops, escapes, data stream branching, data counter</td>
</tr>
<tr>
<td>State register</td>
<td>Instruction counter</td>
<td>Data counter(s)</td>
</tr>
<tr>
<td>Instruction fetch</td>
<td>Memory cycle overhead</td>
<td>Data stream overhead</td>
</tr>
<tr>
<td>Language features</td>
<td>Control flow + data manipulation</td>
<td>Data stream only (no data manipulation)</td>
</tr>
</tbody>
</table>

Machine paradigms

- von Neumann: instruction stream machine
- Data-stream machine: (r)DPU or rDPU
- Instruction stream (anti machine)
- Configuration (Configware)
- Distributed memory architecture (DPA)


CS curricula update is overdue

- EDA shifting to CS mentality
- The impact of Makimoto's Wave
- Reconfigurable Computing needed for SoC
- Reconfigurable Computing needed for µP
- Datastream-based Computing
- CS curricula update is overdue

Lacking Sense of Direction?

"we are o.k.!" (no new direction intended)

SoC means Embedded Systems

- The real labor market: 10 times more programmers will write embedded applications than computer software by 2010

What's the problem?

- Crossing the Hardware / Software Chasm (Mike Butts)

It's the gap between procedural and structural mind set

Traditional CS: programming is (control-)procedural, instruction-stream-based - sources: software

The typical programmer has problems to understand function evaluation without machine mechanisms...

... e.g. by signals rippling through a network of transistors.
Conclusion: all knowledge needed is 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
- languages & (co-)compilation techniques
- morphware vendors like PACT ....
- literature from last 30 years

let's update CS curricula to qualify our students for the real labor market
GI and ITG should cooperate

thank you for your patience

for discussion:

Escape from specific silicon:

FPGA -> general purpose
   -> heavy area penalty -> 1% ......
numerically intensive applications -> coarse grain
   slightly application area specific
   benefit -> power and area efficiency like full custom
KressArray principles

- take systolic array principles
- replace classical synthesis by simulated annealing
- yields the super systolic array
- a generalization of the systolic array
- no more restricted to regular data dependencies
- now reconfigurability makes sense

Super Pipe Networks

The key is mapping, rather than architecture

<table>
<thead>
<tr>
<th>array</th>
<th>applications</th>
<th>pipeline properties</th>
<th>mapping</th>
<th>scheduling</th>
</tr>
</thead>
<tbody>
<tr>
<td>systolic array</td>
<td>regular data dependencies only</td>
<td>linear only</td>
<td>uniform only</td>
<td>linear projection or algebraic synthesis</td>
</tr>
<tr>
<td>super-systolic rDPA</td>
<td>no restrictions</td>
<td>simulated annealing or P&amp;R algorithm</td>
<td>(e.g. force-directed) scheduling algorithm</td>
<td></td>
</tr>
</tbody>
</table>

*) KressArray [1995]

KressArray Xplorer

http://kressarray.de

KressArray Family generic Fabrics: a few examples

Examples of level
Interconnect: entire cell - no separate routing areas!
Compilation for (r)DPA of anti machine

- high level source program
  - parameters
  - wrapper
  - mapware
  - expression
  - tree
  - configware
  - code
  - generators
  - scheduler

systolic array: no routing

placement
routing

The Secret of Success: Co-Compilation

- “VN” machine paradigm
- Partitioner
- Analyzer / Profiler
- SW compiler
- CW Code
- SW code
- resource parameters
- supporting different platforms

High level PL source

Loop Transformation Examples

- sequential processes:
  - loop body
  - endloop

- resource parameter driven
  - Co-Compilation

- host:
  - recent array
  - loop 1-8
  - trigger
  - endloop

- fork
  - loop 1-8
  - trigger
  - endloop

- join
  - loop 9-16
  - trigger
  - endloop

- strip mining

Significance of Address Generators

- Address generators have the potential to reduce computation time significantly.
- In a grid-based design rule check a speed-up of more than 2000 has been achieved, compared to a VAX-11/750.
- Dedicated address generators contributed a factor of 10 - avoiding memory cycles for address computation overhead.

Why a dichotomy of machine paradigms?

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

The anti machine has no von Neumann bottleneck

>> final remarks <<

- Embedded System Design Crisis
- Computing Crisis
- CS for Embedded Systems?
- Flowware-based Computing
- Enabling Architectural Resources
- New Machine Paradigm
- final remarks
Acceleration Mechanisms

- parallelism by multi bank memory architecture
- auxiliary hardware for address calculation
- address calculation before run time
- avoiding multiple accesses to the same data
- avoiding memory cycles for address computation
- improve parallelism by storage scheme transformations
- improve parallelism by memory architecture transformations
- alleviate interconnect overhead (delay, power and area)

Reconfigurable Computing: a second programming domain

Migration of programming to the structural domain
The structural domain has become RAM-based
The opportunity to introduce the structural domain to programmers...
...to bridge the gap by clever abstraction mechanisms using a simple new machine paradigm

Hardware and Software as Alternatives

Brain Usage: both Hemispheres

The Dominance of the Submarine Model ...

...indicates that our CS education system produces zillions of mentally disabled persons...
...completely disabled to cope with solutions other than software only

It's time to attack the software faculty dictatorship. Get involved!
Algorithmic cleverness

Very high throughput on low power slow FPGAs may be obtained only by algorithmic cleverness - not yet taught by CS & CSE at Universities - an urgent educational problem.

Conclusions

- the anti machine is the way to go for massive parallelism, also data-intensive applications
- reconfigurable anti machine for high performance with short product life cycles, unstable standards
- reconfigurable for low cost low volume production
- new infrastructures for the sparepart problem
- Giga FPGAs highly promising - only by a new design flow: configware could repeat the success of software industry

Summary of the Anti Machine Paradigm

- anti language primitives are almost the same (slightly extended)
- anti machine execution potential is dramatically more powerful
- provides drastically more flexibility
- not always replacing von Neumann

Datastream-based Compilation Principles

Conventional processors use the sequential model:
Each operation takes one clock cycle.
Multiple operations are computed consecutively.
A New Parallel Processor Paradigm

Multiple computations are configured as code sections onto a two dimensional array.

Parallel Processor Model

Multiple code sections are computed sequentially.

Dataflow Performance

Traditional Microprocessor

- ALU
- Register
- One word
- One operation per cycle
- Basic machine operations performed on single words

XPP Architecture

- Configuration
- Memory and cache
- Array of ALUs
- Stream of words
- Many operations per cycle
- Complex functions performed on data streams

Dataflow Performance

Process level

- Loop Level (data-stream-based, pipe nets, etc.)
- Instruction Level (VLIW etc.)
- RT Level (special architectures etc.)
- Logic Level (FPGAs)

mentally handicapped

The brain hurts on paradigm shift?

no, it can’t ...

There are more Levels of Parallelism

ignore by typical CS people & ignored by CS curricula
old CS lab course philosophy:
given an application: implement it by a program

new CS freshman lab course environment:
Given an application:

a) implement it by writing a program
b) implement it as a morphware prototype
c) Partition it into P and Q
   c.1) implement P by software
   c.2) implement Q by morphware
   c.3) implement P/Q communication interface
Crusty Computing Sciences

more and more efforts yield only marginal improvements
areas fade away
datflow machines dead
shrinking supercomputing conferences
98.5% VNU-only
this monopoly is the problem

VLSI Design: no more „EE only“

Beginning „design sciences“ as its own discipline
EDA industry: soon first start-ups

Lynn Conway: the hot topic everywhere
Mead & Conway: the hot topic everywhere