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

Data-stream-based Computing

The Anti Universe of Computing

- Paul Dirac predicted a complete anti universe consisting of antimatter
- “There are regions in the universe, which consist of antimatter …
  “ … But there are asymmetries”
- when a particle hits its antiparticle, both are converted into energy: Annihilation
- We are not aware, that there is a new area in computing sciences, which consists of antimatter of computing
- Reconfigurable Computing is made from this antimatter: data-stream-based computing
anti particles

- 1928: Paul Dirac: “there should be an anti electron having positive charge” (Nobel prize 1933)
- 1932: Carl David Anderson detected this “positron” in cosmic radiation (Nobel prize 1936)
- 1954: new accelerators: cyclotron, like Berkeley’s Bevatron
- 1956: anti neutron created on Bevatron
- 1928: Paul Dirac: “there should be an anti electron having positive charge” (Nobel price 1933)
- 1932: Owen Chamberlain et al. create anti proton on Bevatron
- 1955: Owen Chamberlain et al. create anti proton on Bevatron
- 1955: hydrogen anti atom created at CERN
- 1995: hydrogen anti atom created at CERN

Matter & Antimatter: Atom and Anti Atom

The World of Matter - machine paradigm: the Atom

Electron spinning
Positron spinning

Matter & Antimatter of Informatics: Machine and Anti Machine

Anti Machine paradigm

Machine paradigm: “von Neumann”

independent instruction streams difficult ...

Parallelism by Concurrency
• Machine vs. Anti Machine
• Terminology
• Morphware Platforms
• coarse-grained Platforms
• Dual Machine Paradigms
• Data Sequencing
• Final Remarks

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.

Terminology: Digital System Platforms

<table>
<thead>
<tr>
<th>platform</th>
<th>source running on it</th>
<th>machine paradigm</th>
</tr>
</thead>
<tbody>
<tr>
<td>hardware</td>
<td>not running on it</td>
<td>none</td>
</tr>
<tr>
<td>morphware coarse grain</td>
<td>rGA (FPGA)</td>
<td>configware</td>
</tr>
<tr>
<td></td>
<td>rCPU, rDPA</td>
<td></td>
</tr>
<tr>
<td>data stream processor (hardwired)</td>
<td>Flowware &amp; configware</td>
<td>anti machine</td>
</tr>
<tr>
<td>instruction stream processor</td>
<td>software</td>
<td>von Neumann machine</td>
</tr>
</tbody>
</table>

Importance of binding time

not all switching is done by Configware

Configuration: like for a pipe network a kind of pre-packed frozen-in „super instruction fetch“
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“ ...

data streams*: not new

1980: data streams (Kung, Leiserson: systolic arrays)
1989: data-stream-based Xputer architecture
1990: rDPU (Rabaey)
1994: Flowware Language MoPL (Becker et al.)
1995: super systolic array (rDPA) + DPSS tool (Kress)
1996: Streams-C language, SCCC (Los Alamos), SCORE, ASPRC, Bee (UC Berkeley), DSP-C, Brook, ...
1996+: Streams-C language, SCCC (Los Alamos), SCORE, ASPRC, Bee (UC Berkeley), DSP-C, Brook, ...

1996: configure / software partitioning compiler (Becker)

URLs: Software vs Flowware and Configware

http://data-streams.org/
http://anti-machine.org/
http://kressarray.de/

HPC going configware

International Conference on Field-Programmable Logic and Applications (FPL)

http://ifpl.org

Aug. 20 – Sept 1, 2004, Antwerp, Belgium

Morphware Platforms: Drawbacks

• Machine vs. Anti Machine
• Terminology
• Morphware Platforms
• coarse-grained Platforms
• Dual Machine Paradigms
• Data Sequencing
• Final Remarks

Fine-grain Morphware: Drawbacks

• FPGA Architectures
  - SRAM-based Look-up Tables (LUTs)
  - Problems:
    - Routing; reduces Performance
    - Bad Ratio: active / passive Elements

Configurable Logic Block (CLB)
Reconfigurability Overhead

area used by application
partly for configuration code storage
resources needed for reconfigurability
"hidden RAM" not shown

Throughput vs. Efficiency

Wiring by abutment: 32 Bit example
Instruction set processors

One more argument for coarse grain

If coarse grain cells are full custom and mesh-connected, and 2nd level interconnect resources layouted over the cells the array is almost as area-efficient as hardwired.

It's a Paradigm Shift!

• Using FPGAs (fine grain reconfigurable) just mainly has been classical Logic Synthesis on a "strange hardware" platform
• Coarse Grain Reconfigurable Arrays (rDPAs) (Reconfigurable Computing), however, mean a really fundamental Paradigm Shift
• This is still ignored by CS and EE Curricula and almost all R&D scenes

Mega-rGAs

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
>> coarse-grained Platforms <<

- Machine vs. Anti Machine
- Terminology
- Morphware Platforms
- coarse-grained Platforms
- Dual Machine Paradigms
- Data Sequencing
- Final Remarks

Generalized Stream-based Computing System

The same mapper for both:
Reconfigurable, or hardwired

Kress DPSS (1995)

DPU architectures expression tree

Mapper

Scheduler

Configware

Compiler

flowware history:

1980: data streams
(Kung, Leiderson)

1995: super systolic rDPA (Kress)

1996+: SCCC (LANL), SCORE, ASPRC, Bee (UCB)

Standard and courseware available on request

flowware history:

DPA

input data streams

output data streams

Supersystolic Array Principles

- take systolic array principles
- replace classical synthesis by simulated annealing
- yields the supersystolic array
- a 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, ASPRC, Bee (UCB), ...
1997: discipline of distributed memory architecture
1997: configware / software partitioning compiler

**Super Systolic**

The key is mapping, rather than architecture

- **Array**
  - **Applications**: shape, resource
  - **Pipeline Properties**: linear only, uniform only
  - **Mapping**: linear projection or algebraic synthesis
  - **Scheduling**: decomposing or PAR algorithm, (e.g. force-directed) scheduling algorithm

**Programming Language Paradigms**

<table>
<thead>
<tr>
<th>Language Features</th>
<th>Von Neumann Languages</th>
<th>Anti-Machine Languages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Counter</td>
<td>procedural sequencing: goto next instruction, jump (to instruction addr.), instruction, loop, read next data item, instruction stream branching, program counter</td>
<td>instruction, loop, read next data object, goto (data addr.), jump (to data addr.), instruction stream branching, data counter(s)</td>
</tr>
<tr>
<td>Control Flow</td>
<td>control flow only, data manipulation only</td>
<td>control flow only, data manipulation only</td>
</tr>
<tr>
<td>Memory Cycle</td>
<td>memory cycle overhead, overhead avoided</td>
<td>memory cycle overhead, overhead avoided</td>
</tr>
<tr>
<td>Instruction Fetch</td>
<td>instruction fetch parity memory, data access</td>
<td>instruction fetch parity memory, data access</td>
</tr>
</tbody>
</table>

**Similar Programming Language Paradigms**

<table>
<thead>
<tr>
<th>Language Category</th>
<th>Computer Languages</th>
<th>XSPYR Languages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Both Deterministic</td>
<td>procedural sequencing, traceable, checkpointable</td>
<td>procedural sequencing, traceable, checkpointable</td>
</tr>
<tr>
<td>Sequencing Driven By:</td>
<td>read next instruction, goto (instruction addr.), jump (to instruction addr.), instruction loop, read next data object, goto (data addr.), jump (to data addr.), instruction stream branching, data counter(s)</td>
<td>read next data object, goto (data addr.), jump (to data addr.), instruction loop, read next data object, goto (data addr.), jump (to data addr.), instruction stream branching, data counter(s)</td>
</tr>
</tbody>
</table>

**Machine Paradigms**

- **Driven by**: Instruction streams vs. data streams (no “dataflow”)
- **Engine Principles**: Instruction sequencing vs. sequencing data streams
- **State Register**: Single program counter vs. multiple data counter(s)
- **Communication Path Setup**: Instruction, parallel loops
- **Data Paths**: Single, parallel data loop, parallel pipe network etc.
- **Resource**: DPU (e.g. single ALU) vs. DPU or DPA (DPA array) etc.

---

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, ASPRC, Bee (UCB), ...
1997: discipline of distributed memory architecture
1997: configware / software partitioning compiler
The new discipline of (application-specific) distributed memory

Herz et al.: proc. IEEE ICECS 2002

Efficient Memory Communication should be directly supported by the Mapper Tools

Synthesizable Distributed Memory

Commercial rDPA example:
PACT XPP - XPU128

http://pactcorp.com

http://kressarray.de
**XPP64A: Platform Development Board**
- SDR Board in Debug Phase
- XPP64A Chips from STMicro Fab
- Assembly & Test / Available March 2003

**Dataflow Performance**
- Traditional Microprocessor
  - Instruction Memory and cache
  - ALU, Register
  - ADD, One word
  - MULT, One operation per cycle
  - Shift, Stream of words
- XPP Architecture
  - Configuration Memory and cache
  - Array of ALUs, Buffer
  - Complex Functions, Stream of data streams

**>> Data Sequencing <<**
- Machine vs. Anti Machine
- Terminology
- Morphware Platforms
- coarse-grained Platforms
- Dual Machine Paradigms
- Data Sequencing
- Final Remarks

**>> Dual Machine Paradigms <<**
- Machine vs. Anti Machine
- Terminology
- Morphware Platforms
- coarse-grained Platforms
- Dual Machine Paradigms
- Data Sequencing
- Final Remarks

**3rd machine model became mainstream**
- Mainframe age
- Computer age (PC age)
- Morphware age
- Programmable
- µProc, rDPA

**benefit from RAM-based & 2nd paradigm**
- RAM-based platform needed for:
  - flexibility, programmability
  - avoiding the need of specific silicon
- simple 2nd machine paradigm needed as a common model:
  - to avoid the need of circuit expertise
  - needed to educate zillions of programmers
- what programming source language?
McKinsey Curve: dynamics of R&D disciplines

EDA industry paradigm, switching every 7 years

McKinsey Curve: dynamics of R&D disciplines

EDA Industry Revolutions

EDA industry paradigm, switching every 7 years

How to achieve acceptance

No hardware description languages

Tools usable by users not being hardware designers

Courses tailored for students not being hardware-savvy

EDA tools based on term rewriting [Arvind] [Mauricio Ayala]

Your name here: your proposals

configware compiler

symbiosis of machine models
symbiosis of machine models

source \textit{program}

partitioning compiler

\textit{µ}Proc. - \textit{rDPA}

---

Software / Configware Co-Compilation

Juergen Becker's GoDe-X, 1996

High level PL source

\begin{align*}
\text{Partitioner} & \quad \text{SW} \quad \text{Compiler} \\
\text{Analyzer} & \quad \text{Profile} \\
\text{CW Code} & \quad \text{Supporting different platforms}
\end{align*}

---

Loop Transformation Examples

\begin{itemize}
  \item sequential processes:
  \begin{align*}
  \text{loop } 1-16 \\
  \text{loop } 1-8 \\
  \text{fork} \\
  \text{unrolling} \\
  \text{strip mining}
  \end{align*}

  \item resource parameter driven Co-Compilation:
  \begin{itemize}
    \item host:
    \begin{align*}
    \text{loop } 1-8 \\
    \text{trigger} \\
    \text{endloop}
    \end{align*}
    \\
    \begin{align*}
    \text{recent.array:}
    \text{loop } 1-16 \\
    \text{trigger} \\
    \text{endloop}
    \end{align*}
  \end{itemize}
\end{itemize}

---

History of Loop Transformations

Loop Unrolling, Loop Fusion, Strip Mining

70ies - 80ies: at Process Level:
- Sequential to Parallel Processes, incl. Vectorization

1995/97 [Karin Schmidt / Jürgen Becker]: down to Datapath Level:
- (Parameter-driven) Time to Time/Space Partitioning
e. g.: Transformation from Sequential Process to Super-systolic

2000 [Michael Herz]: optimized RA to Memory Communication Bandwidth:
- Multi-dimensional Loop Unrolling / Storage Scheme Optimization
  supporting burst-mode & parallel Memory Banks

---

Data Sequencing

- Machine vs. Anti Machine
- Terminology
- Morphware Platforms
- coarse-grained Platforms
- Dual Machine Paradigms
- Data Sequencing
- Final Remarks

---

application-specific distributed memory*

- Application-specific memory: rapidly growing markets:
  - IP cores
  - Module generators
  - EDA environments
- Optimization of memory bandwidth for application-specific distributed memory
- Power and area optimization as a further benefit
- Key issues of address generators will be discussed

*) see books by Francky Catthoor et al.
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.

Smart Address Generators

1983 The Structured Memory Access (SMA) Machine
1984 The GAG (generic address generator)
1989 Application-specific Address Generator (ASA6)
1990 The slider method: GAG of the MoM-2 machine
1991 The AGU
1994 The GAG of the MoM-3 machine
1997 The Texas Instruments TMS320C54x DSP
1997 Intersil HSP45240 Address Sequencer
1999 Adopt (IMEC)

Adopt (from IMEC)

- cMMU synthesis environment:
  - application-specific ACUs for array index reference
  - ACU as a counter modified by multi-level logic filter
  - ACU with ASUs from a Cathedral-3 library
  - distributed ACU alleviates interconnect overhead (delay, power, area)
  - nested loop minimization by algebraic transformations
  - AE splitting/clustering
  - AE multiplexing to obtain interleaved ASs
  - other features

For more details on Adopt see paper in proceedings CD-ROM.

Distributed Memory

SA: scrambling and descrambling the data?
Just in time: a new research area:
Application-specific distributed memory:
  e. g. book by F. Catthoor et al...
Data address generators - 20 years research:

Generic Sequence Examples

- atomic scan
- linear scan
- video scan
- 90° rotated video scan
- 45° rotated (mirror X-scan)
- sheared video scan
- non-rectangular video scan
- zigzag video scan
- spiral scan
- feed-back driven scans
- perfect shuffle

GAU generative address unit Scheme

Limit Slider
Address Stepper
Base Slider

All 3 are copies of the same BSU stepper circuit.
GAG: Address Stepper

GAG = Generic Address Generator

Limit Stepper
Address Stepper
Base Stepper

GAG Slider Model

Limit Stepper
Address Stepper
Base Stepper

GAG: Address Stepper

speedup by MoM

MoM architecture:
2-D memory space, adj. scan window
grid-based design
rule check example
speed-up: >1000
complex boolean expressions in
1 clock cycle
address computation overhead: 94%
Antimachine: MoM architecture

Handle Position Generator
Scan Window Generator
memory accesses

Size adjustable at run time
square or rectangular shape
location's individual access mode: R, W, R/W, no-op
by no-op placements any wild window shape
avoid multiple read/multiple write for overlapping successive scan window positions

Linear Filter Application

Scanline unrolling

90° Rotation of Scan Pattern

Parallelized Merged Buffer Linear Filter Application
with example image of x=22 by y=11 pixel
Multiple Scan Windows

MoM anti machine
an Xputer architecture

MoM distributed memory

each window is a single memory bank

smart memory interface

counter

example: 4x4 scan windows

16 point CGFFT: mapped onto 2-D memory space

MoM distributed memory

counter

distributed memory

counter

example: 4x4 scan windows

CGFFT: Nested and Parallel Scan Pattern

CGFFT: Parallel Scan Pattern Animation

MAC

in_i, coeff

out_i

in_i+1, coeff

out_i+1

empty

inner loop

outer loop

pattern

HLScan
is 3 steps

[2, 0]

SP1
is 7 steps

[0, 2]

SP23
is 7 steps

[0, 1]

3 in parallel

outer loop

scan pattern

16 steps

16 MAC units

parallel

1 MAC unit

parallel

inner loop

compound scan patterns

CGFFT: Parallel Scan Pattern Animation

CGFFT: Nested and Parallel Scan Pattern

HLScan
is 2 steps

[2, 0]

SP1
is 7 steps

[0, 2]

go to

SP23
is 7 steps

[0, 1]

3 in parallel

inner loop

compound scan patterns
• Machine vs. Anti Machine
• Terminology
• Morphware Platforms
• coarse-grained Platforms
• Dual Machine Paradigms
• Final Remarks

Jürgen Becker

Dissertation
Jürgen Becker: Professor at Univ. Karlsruhe
• (configware / software co-compilation)
• Resource-parameter-driven retargettable
• ... Automatically partitioning Co-compiler
• Profiler-driven optimization
• Accepts HLL „ALE-X“ (extended C subset)
• (subset: pointers not supported)

http://www.uni-kl.de

© 2004, reiner@hartenstein.de
http://hartenstein.de

Rainer Kress

Dissertation
Rainer Kress: infineon technologies, Munich
• ... on mapping applications onto his* KessArray
• DPSS datapath synthesis system
• Including a data scheduler
• (data stream scheduler)
• Generalization of the Systolic Array
• (KressArray is a super systolic array)
• 32 bit design via Eurochip support

Ulrich Nageldinger

Dissertation
Ulrich Nageldinger: infineon technologies, Munich
• Coarse-grained Reconfigurable Architectures
• Design Space Exploration; Dissertation, 2001

http://hartenstein.de/keynotes.html

© 2004, reiner@hartenstein.de
http://hartenstein.de

Michael Herz

Dissertation
Michael Herz: Agilent, Sindelfingen, Germany
• High Performance Memory Communication
• Architectures for Coarse-grained Reconfigurable
• Computing Systems; Dissertation 2001

More Presentations / Literature

http://hartenstein.de/keynotes.html
http://hartenstein.de/publications.html

© 2004, reiner@hartenstein.de
http://hartenstein.de
Antimatter Search?

in EE & CS we do not need to search

END