Coarse Grain Reconfigurable Architectures

>>> Overview

- Introduction
- Coarse Grain Architectures
- Programming Coarse Grain RAs
- Fundamental Issues
- Conclusions

>>> Introduction

- Introduction
- Coarse Grain Architectures
- Programming Coarse Grain RAs
- Fundamental Issues
- Conclusions

R. Hartenstein et al. (invited embedded tutorial): Coarse Grain Reconfigurable Architectures; Asian and South Pacific Design Automation Conf. 2001 (ASP-DAC'01), Yokohama, Japan, Jan 30 - Feb. 2, 2001
Coarse Grain Architectures

Using FPGAs (fine grain reconfigurable) is just mainly classical Logic Synthesis on a "strange hardware" platform. Coarse grain reconfigurable arrays (reconfigurable computing) however, just mean a really fundamental paradigm shift. This is still ignored by CS and EE curricula and almost all R&D scenes.

Why Coarse Grain instead of FPGA?

- Increased efficiency
- Higher density
- Physical reconfigurability
- Reduced reconfigurability overhead by up to ~ 1000
- Higher efficiency
- Many more benefits

- Coarse grain architectures, such as Reconfigurable Interconnect eXperiment (RIEX), dramatically smaller
- More benefits
- Much faster loading
- Much faster reconfiguration
- A lot of more benefits

It's a Paradigm Shift!
MATRIX Interconnect Fabrics
Communication Resources are often the bottleneck

BFUs
its neighbours

Informatik (CS&E Dept.)

CHESS Array w. embedded RAM (hp)

BSU
its neighbours

Informatik (CS&E Dept.)

Matrix Interconnect Fabrics

BFUs
its neighbours

Informatik (CS&E Dept.)

13

More Research Projects

Garp (UC Berkeley)
RaPiD (U. Washington)
REMARC (Stanford)
DReAM (TU Darmstadt)
... and others
Asia / Pacific: also see embedded tutorials by Prof. Amano
(ASP_DAC'99, FPL-2000)

Informatik (CS&E Dept.)

15

http://www.fpl.uni-kl.de

XPU128 Block Diagram

XPU family:
PACT corp.
CALISTO:
Silicon Spice
CS2000 family:
Chameleon Systems
MECA family:
Malleable
flexible array:
MorphICs
FIPSOC:
SIDSA

Informatik (CS&E Dept.)

16

http://www.fpl.uni-kl.de

KressArray (1995) mapping of SNN filter
member of generic KressArray family

KressArray

Legend:
backbus connect
memory interface
operator and routing
port location marker

Legend:
backbus connect
memory interface
operator and routing
port location marker

Informatik (CS&E Dept.)

17

http://kressarray.de

R. Hartenstein et al. (invited embedded tutorial): Coarse Grain Reconfigurable Architectures; Asian and South Pacific Design Automation Conf. 2001 (ASP-DAC'01), Yokohama, Japan, Jan 30 - Feb. 2, 2001
It is indispensable to operate within the Convergence Area of Compilers, Co-Compilers, Architecture and full-custom-style VLSI Design (array cells).

- It is a must, that Products come with many others Tensilica

Dynamically Reconfigurable Architectures

Coarse Grain Architectures

- Brief Overview on Architectures
- Dynamically Reconfigurable Architectures
- RA Architecture Generators
- Memory Bandwidth Problems

Super Pipe Networks

The key is mapping, rather than architecture

array applications pipeline properties mapping scheduling (data stream formation)

shape resources

linear only uniform only linear projection or algebraic synthesis

simulated annealing or P&R algorithm (e.g. force-directed) scheduling algorithm

KressArray (ASP-DAC'1995)

Informatik (CS&E Dept.) 19 http://www.fpl.uni-kl.de

Dimensions of Reconfigurability

Class of Processors product vendor

ASIP Tensilica Tensilica

MECA family Malleable

CALISTO SiliconSpice

Network Processors

Configuration time design time

fabrication time compile time run time

ASIP

PipeRench Architecture (CMU 1998)

alternating data/instruction stream

highly dynamic reconfiguration

Informatik (CS&E Dept.) 24 http://www.fpl.uni-kl.de

R. Hartenstein et al. (invited embedded tutorial): Coarse Grain Reconfigurable Architectures; Asian and South Pacific Design Automation Conf. 2001 (ASP-DAC’01), Yokohama, Japan, Jan 30 - Feb. 2, 2001
Reiner Hartenstein, Informatik Department, TU Kaiserslautern
http://hartenstein.de

R. Hartenstein et al. (invited embedded tutorial): Coarse Grain Reconfigurable Architectures; Asian and South Pacific Design Automation Conf. 2001 (ASP-DAC'01), Yokohama, Japan, Jan 30 - Feb. 2, 2001
Memory Bandwidth Problems

Coarse Grain Architectures
- Brief Overview on Architectures
- Dynamically Reconfigurable Architectures
- RA Architecture Generators
- Memory Bandwidth Problems

RAs: Cache does not help

- super pipe networks, no parallel computers!
- the memory bandwidth problem is often more dramatic than for microprocessors
- classical caches do not help, since instruction sequencing is not used
- interleaving is not practicable, since based on sequential instruction streams
- the problem: throughput of parallel data streams, not instruction streams

Synthesizable Memory Communication

Efficient Memory Communication should be directly supported by the Mapper Tools

Legend:
- sequencers
- memory ports
- application
- not used
- DPU used to implement the application
- DPU used to implement the data sequencer
- memory port
- "St" = stepper operator
- "<" = comparator
- "SWG" = scan window generator
- "SM" = stream mapper
- "cSM" = SM with control input
- "DD" = data distributor

Example by Design Space Xplorer

Reconfigurable parallel Memory Communication Architecture

- Introduction
- Coarse Grain Architectures
- Programming Coarse Grain RAs
- Fundamental Issues
- Conclusions
FPGA-Style Mapping for coarse grain reconfigurable arrays

<table>
<thead>
<tr>
<th>Developer</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPSS</td>
<td>Compiler</td>
</tr>
<tr>
<td>CHESS</td>
<td>Mapper</td>
</tr>
<tr>
<td>RaPiD</td>
<td>Scheduler</td>
</tr>
<tr>
<td>Colt</td>
<td></td>
</tr>
</tbody>
</table>

specifies and assembles the data streams from/to array

Kress Array DPSS (Datapath Synthesis System)

Co-Compilation

Software running on Computer
- hardwired Machine Paradigm
- high level programming language source
- Configware, running on Xputer
- "Soft" Machine Paradigm

Host
- Reconfigurable Accelerators
- Reconfigurable Architecture (RA)

History of Loop Transformations

Instruction Code vs. Reconfiguration Code

Pathfinder greedy algorithm

Algorithm
Simulated Annealing
Genetic Algorithm
Routing
Placement
Mapping

Loop Transformation Examples

resource parameter driven Co-Compilation

Jürgen Becker’s Co-DE-X Co-Compiler

R. Hartenstein et al. (invited embedded tutorial): Coarse Grain Reconfigurable Architectures; Asian and South Pacific Design Automation Conf. 2001 (ASP-DAC’01), Yokohama, Japan, Jan 30 - Feb. 2, 2001
Fundamental Issues

- Introduction
- Coarse Grain Architectures
- Programming Coarse Grain RAs
- Fundamental Issues
- Conclusions

Changing Models of Computing

Software (procedural)
- downloading
- instruction sequencer
- instruction code
- RAM
- instruction stream branching
- I/O
- instruction fetch
- hardwired accelerator (ASICS)

Software (reconfigurable)
- downloading
- instruction sequencer
- instruction code
- RAM
- instruction stream branching
- I/O
- instruction fetch
- hardwired accelerator (ASICS)

Fundamental Issues

Coarse Grain Architectures
- memory cycle overhead
- operation
- massive memory
- program counter
- overhead avoided
- control flow
- data sequencing
- procedural sequencing: deterministic
- procedural sequencing: traceable, checkpointable
- Instruction sequencing

Machine Paradigms

<table>
<thead>
<tr>
<th>machine category</th>
<th>Computer (&quot;v. Neumann&quot;)</th>
<th>Xputer ([8]) (no transputer!)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Machine paradigm</td>
<td>procedural sequencing: deterministic</td>
<td>procedural sequencing: stochastic, checkpointable</td>
</tr>
<tr>
<td>driven by:</td>
<td>control flow</td>
<td>read next instruction, data stream(s)</td>
</tr>
<tr>
<td>RA support</td>
<td>no</td>
<td>read next data item, data stream(s)</td>
</tr>
<tr>
<td>engine principles</td>
<td>Instruction sequencing</td>
<td>read next instruction, data stream(s)</td>
</tr>
<tr>
<td>state register</td>
<td>program counter (multiple data counter(s))</td>
<td>data counter(s)</td>
</tr>
<tr>
<td>communication</td>
<td>at run time</td>
<td>memory cycle overhead</td>
</tr>
<tr>
<td>path set-up</td>
<td>at load time</td>
<td>overhead avoided</td>
</tr>
<tr>
<td>data</td>
<td>single ALU</td>
<td>instruction fetch</td>
</tr>
<tr>
<td>resource</td>
<td>array of ALUs</td>
<td>overhead avoided</td>
</tr>
<tr>
<td>path</td>
<td>sequential</td>
<td>parallel</td>
</tr>
<tr>
<td>operation</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Programming Languages

<table>
<thead>
<tr>
<th>language category</th>
<th>Computer Languages</th>
<th>Xputer Languages</th>
</tr>
</thead>
<tbody>
<tr>
<td>operation sequences driven by:</td>
<td>read next instruction, data stream(s)</td>
<td>read next data item, data stream(s)</td>
</tr>
<tr>
<td>read next instruction, data stream(s)</td>
<td>read next data item, data stream(s)</td>
<td></td>
</tr>
<tr>
<td>jump (to inner, outer), branch</td>
<td>data counter(s)</td>
<td></td>
</tr>
<tr>
<td>data loop, loop nesting,</td>
<td>instruction fetch</td>
<td></td>
</tr>
<tr>
<td>parallel loops, parallel data loops, data stream branching</td>
<td></td>
<td></td>
</tr>
<tr>
<td>no restrictions</td>
<td>memory cycle overhead</td>
<td></td>
</tr>
<tr>
<td>overhead avoided</td>
<td>overhead avoided</td>
<td></td>
</tr>
<tr>
<td>instruction fetch</td>
<td>memory cycle overhead</td>
<td></td>
</tr>
<tr>
<td>overhead avoided</td>
<td>instruction fetch</td>
<td></td>
</tr>
<tr>
<td>no restrictions</td>
<td>overhead avoided</td>
<td></td>
</tr>
<tr>
<td>instruction fetch</td>
<td>memory cycle overhead</td>
<td></td>
</tr>
<tr>
<td>overhead avoided</td>
<td>instruction fetch</td>
<td></td>
</tr>
<tr>
<td>no restrictions</td>
<td>instruction fetch</td>
<td></td>
</tr>
</tbody>
</table>

Dr. Makimoto: keynote at FPL 2000

R. Hartenstein et al. (invited embedded tutorial): Coarse Grain Reconfigurable Architectures; Asian and South Pacific Design Automation Conf. 2001 (ASP-DAC'01), Yokohama, Japan, Jan 30 - Feb. 2, 2001

http://hartenstein.de
Conclusions

- Introduction
- Coarse Grain Architectures
- Programming Coarse Grain RAs
- Fundamental Issues
- Conclusions

How’s next Wave?

Co-Compilation opens new Horizon

- Computing in Space: "von Neumann" paradigm
- Data Sequencer Methodology
- Parallelizing Loop Transformation Methods
- Compilers supporting Soft Machines
- Migration by retiming

A Decade of Research in Reconfigurable Computing

- By numerous Research Projects throughout the 90ies
- A rich Supply of attractive Ingredients is available
- The time has come to spread the Awareness and to include the Achievements in CS&E Curricula
- Dear colleague, the Reconfigurable Computing Scene is waiting for you: Get involved!

Fundamental Ideas available

- Data Sequencer Methodology
- Data-procedural Languages (Duality w. v. N.)
- Supporting memory bandwidth optimization
- Soft Data Path Synthesis Algorithms
- Compilers supporting Soft Machines
- SW / CW Partitioning Co-Compilers

Thank you for your attention

Reiner Hartenstein, Informatik Department, TU Kaiserslautern
http://hartenstein.de

R. Hartenstein et al. (invited embedded tutorial): Coarse Grain Reconfigurable Architectures; Asian and South Pacific Design Automation Conf. 2001 (ASP-DAC'01), Yokohama, Japan, Jan 30 - Feb. 2, 2001
ready for Soft Machine Paradigm

• DD-driven* (instead of control-flow-driven)
• i. e. data-procedural (not control-procedural)
• deterministic (no "data flow machine")
• supporting soft data paths
• supporting data path arrays (i. e. KressArray)
• supporting multiple I/O data streams

*data-dependency-driven: data sequencing instead of instruction sequencing

The Language Paradigm Shift

• hard to understand from Frog’s Perspective
  - known HW/SW communication barriers
• Orientation only from a Bird’s Perspective:
  - Understanding the Paradigm Shift
  - Smelling the Application Trends
  - Predicting the Significance to EDA in general

The Language Paradigm Shift

• hard to understand from Frog’s Perspective
  - known HW/SW communication barriers
• Orientation only from a Bird’s Perspective:
  - Understanding the Paradigm Shift
  - Smelling the Application Trends
  - Predicting the Significance to EDA in general

The Language Paradigm Shift

• hard to understand from Frog’s Perspective
  - known HW/SW communication barriers
• Orientation only from a Bird’s Perspective:
  - Understanding the Paradigm Shift
  - Smelling the Application Trends
  - Predicting the Significance to EDA in general

END

backup for discussion

New Paradigm

Terminology

<table>
<thead>
<tr>
<th>Paradigm</th>
<th>Platforms</th>
<th>Programming source</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;von Neumann&quot;</td>
<td>Hardware</td>
<td>Software</td>
</tr>
<tr>
<td>Xputer</td>
<td>Coarse-grain Flexware</td>
<td>high level Configware</td>
</tr>
<tr>
<td>RL (FPGA etc.)</td>
<td>fine grain Flexware</td>
<td>netlist level Configware</td>
</tr>
</tbody>
</table>
Coarse Grain Reconfigurable Arrays vs. Parallel Processes

Parallelism at Datapath Level

binding time: (setup of communication channels)

run time
loading time
compile time

MoPL + GAG

binding time (setup of communication channels)

Reconfigurable Computing

The KressArray is a generalization of the systolic array: super systolic array.

MoPL + GAG

Generic Sequence Examples

Slider Operation Demo Example

R. Hartenstein et al. (invited embedded tutorial): Coarse Grain Reconfigurable Architectures; Asian and South Pacific Design Automation Conf. 2001 (ASP-DAC'01), Yokohama, Japan, Jan 30 - Feb. 2, 2001
R. Hartenstein et al. (invited embedded tutorial): Coarse Grain Reconfigurable Architectures; Asian and South Pacific Design Automation Conf. 2001 (ASP-DAC'01), Yokohama, Japan, Jan 30 - Feb. 2, 2001

GAG = Generic Address Generator

- **Limit Stepper**
- **Address Stepper**
- **Base Stepper**

**GAG Scheme**

- \( B_0 \)
- \( \Delta A \)
- \( L \)
- \( \text{limit} \)

**GAG: Address Stepper**

- \( B_0 \)
- \( \Delta A \)
- \( L \)
- \( \text{limit} \)

**MoPL: JPEG Zigzag Scan Pattern**

- **JPEG zigzag scan pattern**

**Fundamental Ideas available**

- Data Sequencer Methodology
- Data-procedural Languages (Duality w. v. N.)
- ... supporting memory bandwidth optimization
- Soft Data Path Synthesis Algorithms
- Parallelizing Loop Transformation Methods
- Compilers supporting Soft Machines
- SW / CW Partitioning Co-Compilers

**ready for Soft Machine Paradigm**

- DD-driven* (instead of control-flow-driven)
- i. e. data-procedural (not control-procedural)
- deterministic (no "data flow machine")
- supporting soft data paths
- supporting data path arrays (i. e. KressArray)
- supporting multiple I/O data streams

*data-dependency-driven: data sequencing instead of instruction sequencing

**The Language Paradigm Shift**

- hard to understand from Frog's Perspective
  - known HW/SW communication barriers
- Orientation only from a Bird's Perspective:
  - Understanding the Paradigm Shift
  - Smelling the Application Trends
  - Predicting the Significance to EDA in general
Host / Accelerator Symbiosis - the new dominant machine model

Reconfigurable Architecture (RA)

Partitioning compiler

Structural programming

Systolic arrays etc.

Migration by retiming and other transformations

Co-Compilation

Co-Compilation

Co-Compilation

The end of Hypergrowth?
EDA Industry (R)evolution
The basic EDA paradigms change every 7 years
- 1978: Transistor entry: Applicon, Calma, CV ...
- 1985: Schematics entry: Daisy, Mentor, Valid ...
- 1992: Synthesis: Cadence, Synopsys ...
- 1999: Component-based design, ASPP

Development of Hypergrowth Markets
Mainstream
Paradigm Shift
we are here

The last EDA Industry Revolution?
SoC: partitioning compilers ...
- 1978: Transistor entry: Applicon, Calma, CV ...
- 1985: Schematics entry: Daisy, Mentor, Valid ...
- 1992: Synthesis: Cadence, Synopsys ...

Longevity by Configware Download
Compresses needed space - 3D hardware
- Reconfigured at customer's site
- Design from vendor's site

R. Hartenstein et al. (invited embedded tutorial): Coarse Grain Reconfigurable Architectures; Asian and South Pacific Design Automation Conf. 2001 (ASP-DAC'01), Yokohama, Japan, Jan 30 - Feb. 2, 2001
High density catalogue FPGA – System on Programmable Chip

- Configurable System on Chip (CSoC)
- Specialised SOC chip with FPL as IP block

new cellular standard: up to 2 Mbit/sec: new CDMA standard:
> 500 MIPS needed just for RF receiver part

- wide variety of end-user’s devices: smart handles, palm pilots, laptops, games, camcorder-likes, ..the internet car, many new types of devices to come ...
- increasing wide variety of services available from network provider: download just what a particular customer is subscribed to

Classes of FPL SOC Chip

Network Processors: > 20 Players
Cisco: Xilinx’s largest Customer

Company
Adaptive Silicon
Chameleon Systems
Silicon Spice
Systelik

table

Architecture
Net disclosed
32-bit datapath array
Net disclosed
Net disclosed
Bit Serial Systolic Array
System on Chip

Business Model
Sell
Sell Chips
Sell Chips
Sell Chips
Sell Chips

Markets
Embedded DSP
Networking
Voice over IP
Wireless Communication
Signal Conditioning
Embedded Systems

R. Hartenstein et al. (invited embedded tutorial): Coarse Grain Reconfigurable Architectures; Asian and South Pacific Design Automation Conf. 2001 (ASP-DAC’01), Yokohama, Japan, Jan 30 - Feb. 2, 2001
Informatik (CS&E Dept.)

DSEs

Design Space Exploration Systems

<table>
<thead>
<tr>
<th>Explorer System</th>
<th>year</th>
<th>source</th>
<th>interactive</th>
<th>status evaluation</th>
<th>status generation</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIP</td>
<td>1997</td>
<td>[66]</td>
<td>no</td>
<td>abstract models</td>
<td>rule-based</td>
</tr>
<tr>
<td>Dda</td>
<td>1997</td>
<td>[67]</td>
<td>yes</td>
<td>prediction models</td>
<td>device generator</td>
</tr>
<tr>
<td>DAI</td>
<td>1998</td>
<td>[68]</td>
<td>yes</td>
<td>prediction from library</td>
<td>rule-based</td>
</tr>
<tr>
<td>DSE for RAW</td>
<td>1999</td>
<td>[69]</td>
<td>no</td>
<td>analytical models</td>
<td>analytical</td>
</tr>
<tr>
<td>ICOS</td>
<td>1999</td>
<td>[70]</td>
<td>no</td>
<td>fuzzy logic</td>
<td>greedy search</td>
</tr>
<tr>
<td>DSE for Multimedia</td>
<td>1999</td>
<td>[71]</td>
<td>no</td>
<td>simulation</td>
<td>synthesis and binding</td>
</tr>
<tr>
<td>Xplorer</td>
<td>1999</td>
<td>[72]</td>
<td>yes</td>
<td>fuzzy rule-based</td>
<td>simulated annealing</td>
</tr>
</tbody>
</table>

Design Flow of Domain-Specific Architecture Optimization

Nagelnder's KressArray Design Space

Xplorer: accessible by internet:
http://kressarray.de

runs best with Netscape 4.6.1

Design Space Exploration

• Introduction
• Coarse Grain Architectures
• Programming Coarse Grain RAs
• Compilation Techniques
  • Design Space Exploration
  • Design Space Explorer (DSEs)
  • Platform Space Explorers (PSEs)
  • Compiler / PSE symbiosis
  • Parallel computing vs. reconfigurable
• Conclusions

Primarily Mesh-based ....

DSEs: an overview

• For VLSI design in general
• for parallel Computer Systems
• Xplorer the only one for reconfigurable platforms (auch MATRIX?)

Design Space Exploration Systems

<table>
<thead>
<tr>
<th>Explorer System</th>
<th>year</th>
<th>source</th>
<th>interactive</th>
<th>status evaluation</th>
<th>status generation</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSE for RAW</td>
<td>1998</td>
<td>[49]</td>
<td>no</td>
<td>analytical models</td>
<td>analytical</td>
</tr>
<tr>
<td>Xplorer</td>
<td>1999</td>
<td>[50]</td>
<td>yes</td>
<td>fuzzy rule-based</td>
<td>simulated annealing</td>
</tr>
</tbody>
</table>

Multigranular: Per Fullnote!

DSEs

http://www.fpl.uni-kl.de

Informatik (CS&E Dept.)

Reiner Hartenstein, Informatik Department,
TU Kaiserslautern
http://hartenstein.de
### Coarse Grain

<table>
<thead>
<tr>
<th>Source</th>
<th>Project</th>
<th>Bits Granularity</th>
<th>Fabric</th>
<th>Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>UC Berkeley</td>
<td>PADDI</td>
<td>16</td>
<td>2-D mesh</td>
<td>Multimedia</td>
</tr>
</tbody>
</table>

### Coarse Grain Architectures

<table>
<thead>
<tr>
<th>Source</th>
<th>Project</th>
<th>Bits Granularity</th>
<th>Fabric</th>
<th>Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>UC Berkeley</td>
<td>PADDI-2</td>
<td>16</td>
<td>2-D array</td>
<td>Multimedia</td>
</tr>
</tbody>
</table>

### Primarily Mesh-based ...

<table>
<thead>
<tr>
<th>Source</th>
<th>Project</th>
<th>Bits Granularity</th>
<th>Fabric</th>
<th>Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chameleon Systems</td>
<td>FIPSOC</td>
<td>16 &amp; analog</td>
<td>SIDSA</td>
<td>Commercial</td>
</tr>
</tbody>
</table>

### PADDI-II Architecture

<table>
<thead>
<tr>
<th>Source</th>
<th>Project</th>
<th>Bits Granularity</th>
<th>Fabric</th>
<th>Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>UC Berkeley</td>
<td>PADDI-2</td>
<td>16</td>
<td>2-D array</td>
<td>Multimedia</td>
</tr>
</tbody>
</table>

### RaPID Architecture

<table>
<thead>
<tr>
<th>Source</th>
<th>Project</th>
<th>Bits Granularity</th>
<th>Fabric</th>
<th>Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>UC Irvine</td>
<td>Chameleon Systems</td>
<td>16 &amp; analog</td>
<td>SIDSA</td>
<td>Commercial</td>
</tr>
</tbody>
</table>

R. Hartenstein et al. (invited embedded tutorial): Coarse Grain Reconfigurable Architectures; Asian and South Pacific Design Automation Conf. 2001 (ASP-DAC'01), Yokohama, Japan, Jan 30 - Feb. 2, 2001
Coarse Grain Architectures

- Brief Overview on Architectures
- Dynamic Reconfiguration
- Memory bandwidth problems
- Architecture Generators

Conclusions

Routing Congestion [DeHon]:
- often 50% or less of CLBs used

> 1000 transistors

> 15 transistors

at each switching point

at each crossbar

Routing Overhead in FPGAs

>40 transistors

>1000 transistors

>15 transistors

part of the hidden RAM

most FPGA vendors gate count:
1 flipflop of configuration RAM = 4 gates

Reconfigurable Computing
is going Mainstream

FPL (early bird before FPGA and FCCM)
- having been a little workshop,
founded 1991 at Oxford, UK

FPL-2000 at Villach: submissions & attendance doubled within a year
http://www.fpl.uni-kl.de/FPL/

... now major conferences follow the bandwagon — e.g. ASP-DAC and
http://www.date-conference.com

predicted by Dr. Makimoto