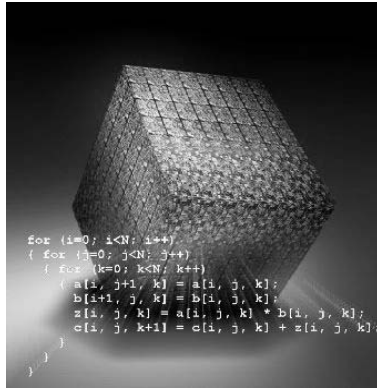


# Reconfigurability Issues of Future Massively Parallel SoCs

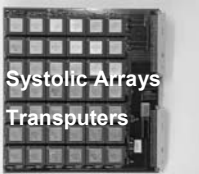

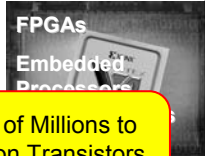



Prof. Dr.-Ing. Jürgen Teich  
Hardware/Software Co-Design  
University of Erlangen-  
Nuremberg  
Am Weichselgarten 3  
91056 Erlangen  
teich@cs.fau.de

## Structure of Presentation

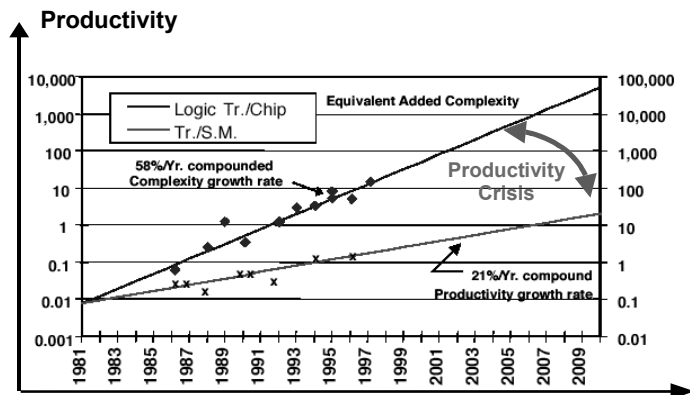
- Motivation
  - MPSoC Trends and Challenges in the nano area
  - Needs for and Potentials of Reconfigurability
  - New: Invasive Algorithms and Architectures
- Invasion: Hardware Challenges
  - Hardware/Software Reconfigurability
  - Decentralized Control of Resources
- Invasion: Algorithmic and Programming Challenges
  - Notation of commands supporting Invasion
  - Languages
  - Compilers
- Core research fields of the future

# Motivation

1980s	1990s	2000s
Class of Algo. RIA	Class of Algo. PLA	HL languages for HW & SW desired
 <p>Systolic Arrays Transputers</p>		 <p>FPGAs Embedded Processors</p>
<ul style="list-style-type: none"> <li>Low performance</li> <li>Too specific</li> <li>Too expensive</li> </ul>	<ul style="list-style-type: none"> <li>Several cells in one chip</li> <li>Still too specific</li> <li>Still too expensive</li> </ul>	<ul style="list-style-type: none"> <li>High performance</li> <li>Programmability of cells</li> <li>Moderate price</li> </ul>
<p>Jürgen Teich University of Erlangen-Nuremberg</p>		<p>MPSoC'08, June 27, 2008</p>

100s of Millions to 1 Billion Transistors integrated in one Chip

# Motivation: SIA Roadmap



How to organize, manage and program 1000s of processors in 2020?

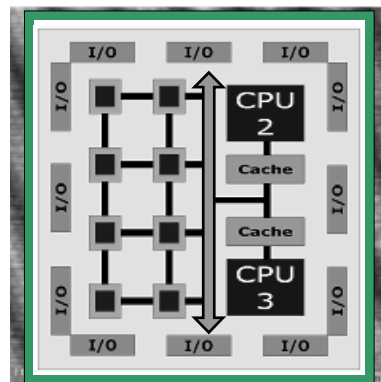
# Inevitable Challenges of the Future

- Complexity
  - How to map algorithms to 1000 processors or more in space and time to benefit from the massive parallelism available including memory, communication, and processing?
- Adaptivity
  - How and to what degree should MPSoCs be equipped with support for adaptivity, resp. reconfigurability and to what degree (HW/SW, bit/word/loop/thread/process-level)?
- Scalability
  - How to specify algorithms and generate executable programs that are able to run efficiently without change on 1,2,4, or N processors?
- Physical constraints
  - Low power, performance exploitation, management overhead

# Reconfigurable MPSoCs

Possible future:

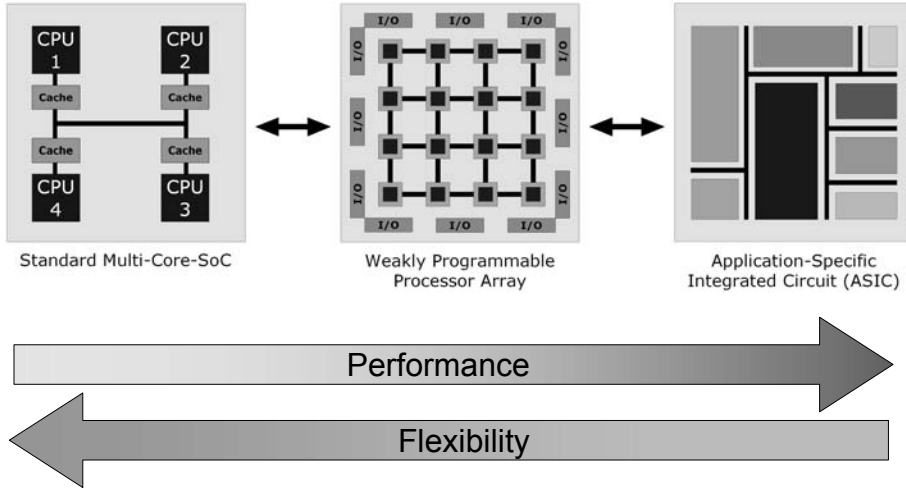
- Different levels of reconfigurability:
  - Nano
  - Fine-grained
  - Coarse-grained
  - Software programmable cores



Research directions:

- Scalability of algorithms
- Algorithms and applications for self-management of resources
- Yield and reliability management

# Parallel Architectures' Trade-offs



# Overview of MPSoC Research

- MorphoSys (University of California, Irvine)
- DinoBench (Carnegie Mellon University)

**Required are  
new programming paradigms for  
decentralized and dynamic resource  
management using reconfiguration  
for adaptivity**

- FastMCHT
- IPFlex
- Stretch
- EU projects: Shapes, Aether,...

# Invasion

## ➤ Definition

[J. Teich. Invasive Algorithms and Architectures. Journal of Information Technology, 2008. To appear.]

*Invasive programming denotes the capability of a program running on an MPSoC to request and temporarily claim processing, communication and memory resources in the neighborhood of its actual computing environment, to then execute in parallel using these claimed resources, and to be capable to subsequently release these resources again.*

# Structure of Presentation

## ➤ Motivation

- MPSoC Trends and Challenges in the nano area
- Needs for and Potentials of Reconfigurability
- New: Invasive Algorithms and Architectures

## ➤ Invasion: Hardware Challenges

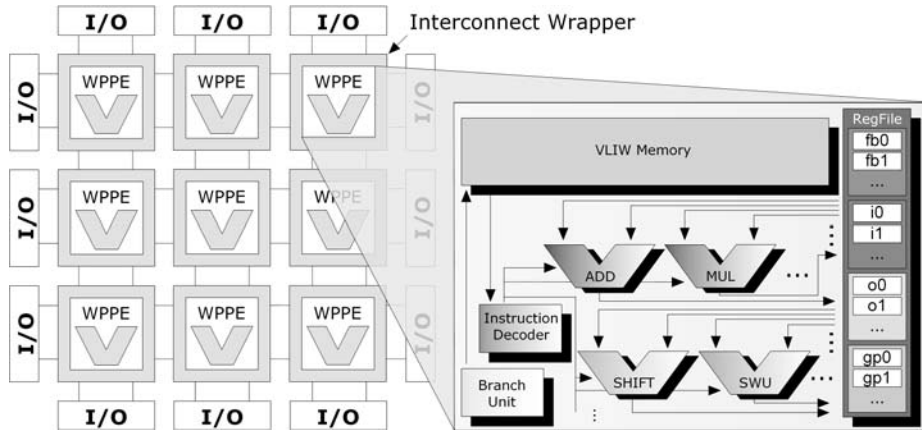
- Hardware/Software Reconfigurability
- Decentralized Control of Resources

## ➤ Invasion: Algorithmic and Programming Challenges

- Notation of commands supporting Invasion
- Languages
- Compilers

## ➤ Core research fields of the future

# Invasive MPSOCs - InvasICs



## Example of an InvasIC

- WPPA: Coarse-grained, highly parameterizable, massively parallel architecture templates with (re)programming capabilities
- Tightly coupled processing with multiple levels of parallelism:
  - Array (loop)- level parallelism
  - Instruction level parallelism
  - Sub-word parallelism
- Basic building blocks: weakly programmable processing elements (WPPE)
  - VLIW architecture with small instruction memory
  - Instruction set small, optimized for digital signal processing
  - Instruction set parameterizable at synthesis time

# Example FIR-Filter

- Sequential C Code:

```
for (i=0; i<T; i++)
  for (j=0; j<N; j++)
    y[i] += a[j] * u[i-j];
```

- Single Assignment Specification

$$\langle i, j : 0 \leq i < T \wedge 0 \leq j < N ::$$

$$a[i, j] = a[i - 1, j] \quad \text{If } i > 0$$

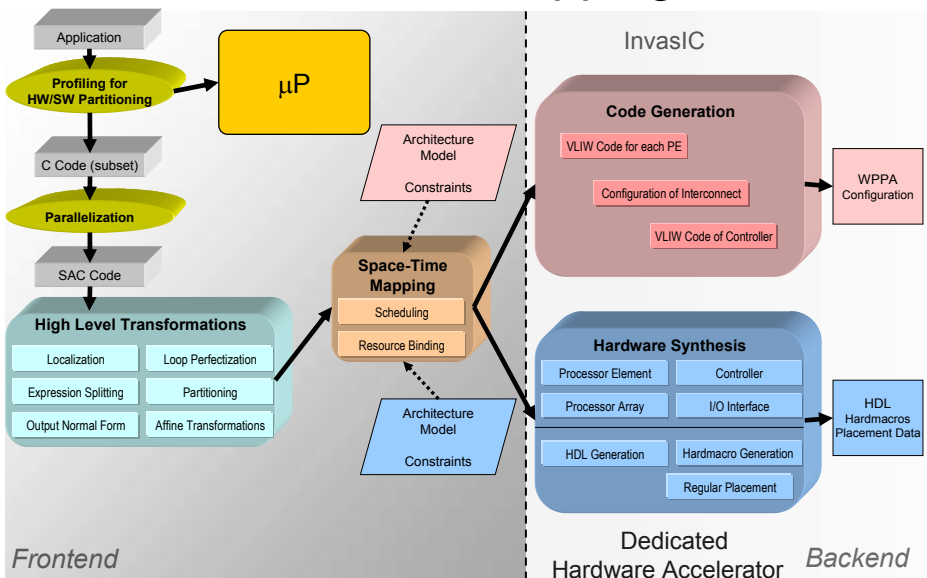
$$= A_j \quad \text{If } i = 0$$

$$u[i, j] = u[i - 1, j - 1] \quad \text{If } i > 0 \wedge j > 0$$

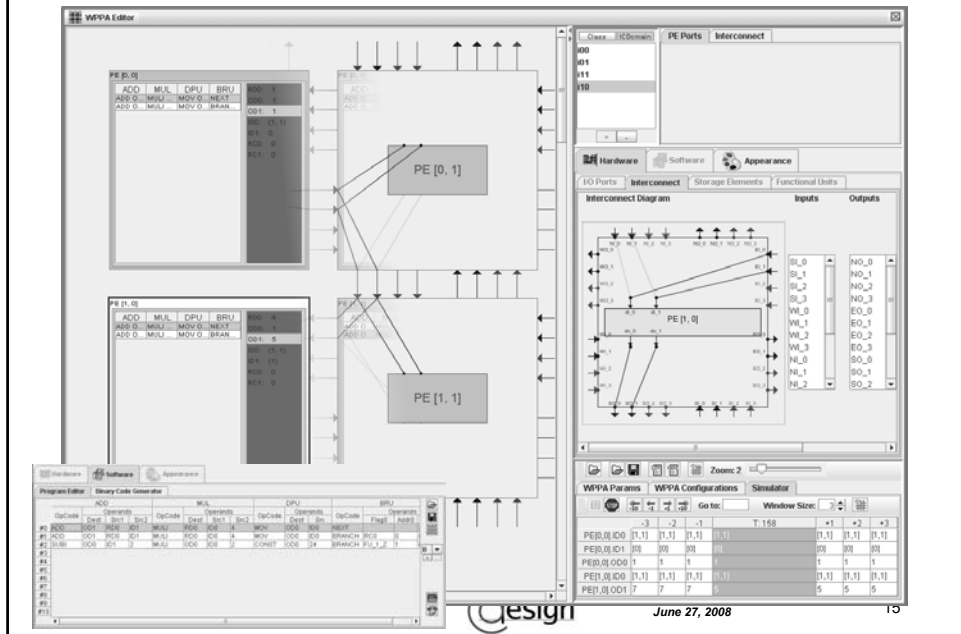
$$= U_{i-j} \quad \text{If } i = 0 \vee j = 0$$

$$y[i, j] = y[i, j - 1] + a[i, j] \cdot u[i, j] \rangle$$

# Parallelization and Mapping



# Tools: Editor & simulator



## Invasive FIR-Filter

$P = \text{INVADE}(\text{my\_id}, \text{WEST}, N);$

$\langle \text{par} : 0 \leq p < P ::$

$\langle \text{seq} : t = N/P \cdot \eta_1 + (N/P + 1) \cdot \eta_2 + 2 \cdot N/P \cdot \eta_3 : 0 \leq \eta_1 < 2 \wedge 0 \leq \eta_2 < N/P \wedge 0 \leq \eta_3 < \lceil T/2 \rceil ::$

$$a[p, t] = \begin{cases} a[p, t - N/P] & \text{if } \eta_1 > 0 \\ a[p - 1, t - 1] & \text{if } p > 0 \wedge \eta_1 = 0 \\ a[p + 1, t - 1] & \text{if } \eta_3 > 0 \wedge p = 0 \wedge \eta_1 = 0 \\ A_{\eta_2} & \text{if } \eta_3 = 0 \wedge p = 0 \wedge \eta_1 = 0 \\ u[p, t - N/P - 1] & \text{if } \eta_1 > 0 \wedge \eta_2 > 0 \\ u[p - 1, t - 2] & \text{if } p > 0 \wedge \eta_1 = 0 \wedge \eta_2 > 0 \\ u[p + 1, t - 2] & \text{if } \eta_3 > 0 \wedge p = 0 \wedge \eta_1 = 0 \\ U_{\eta_1 + 2 \cdot p + 4 \cdot \eta_3 - \eta_2} & \text{if } \eta_3 = 0 \wedge p = 0 \wedge \eta_1 = 0 \\ & \wedge \eta_2 > 0 \end{cases}$$

$$z[p, t] = a[p, t] \cdot b[p, t]$$

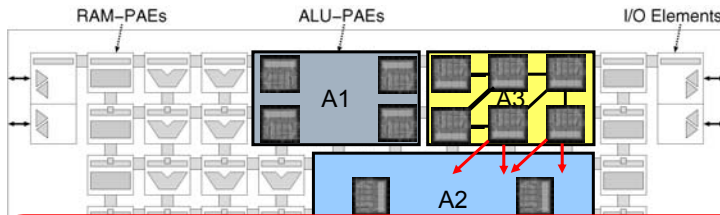
$$y[p, t] = \begin{cases} y[p, t - 1] + z[p, t] & \text{if } \eta_2 > 0 \\ z[p, t] & \text{if } \eta_2 = 0 \end{cases}$$

$\rangle\rangle$

$\text{RETREAT}();$



# Invasion - Example



**New programming and architecture paradigm:  
Invasion**

**Algorithms that are able to adapt their  
implementation at run-time autonomously on the SoC**

# Invasive FIR-Filter - Performance

# of PEs $P$	Throughput (output samples/clock cycle)
64	1.00
32	0.50
16	0.25
8	0.125
4	0.062
2	0.031

# Structure of Presentation

- Motivation
  - MPSoC Trends and Challenges in the nano area
  - Needs for and Potentials of Reconfigurability
  - New: Invasive Algorithms and Architectures
  
- Invasion: Hardware Challenges
  - Hardware/Software Reconfigurability
  - Decentralized Control of Resources
  
- Invasion: Algorithmic and Programming Challenges
  - Notation of commands supporting Invasion
  - Languages
  - Compilers
  
- Core research fields of the future

# Reconfigurability Challenges (1)

```

P = INVADE(my_id, WEST, N);
(par : 0 ≤ p < P ::
(seq : t = N/P · η1 + (N/P + 1) · η2 + 2 · N/
...
a[p, t]
u[p, t] = { u[p + 1, t - 2]
            Uη1+2·p+4·η2-m
z[p, t] = a[p, t] · b[p, t]
y[p, t] = { y[p, t - 1] + z[p, t] if η2 > 0
           z[p, t]           if η2 = 0
})
RETREAT();
                
```

**Notation and commands (instructions) supporting INVASION?**

# Reconfigurability Challenges (2)

$P = \text{INVADE}(\text{my\_id}, \text{WEST}, N);$

$\langle \text{par} : 0 \leq p < P ::$

$\langle \text{seq} : t = N/P \cdot \eta_1 + (N/P + 1) \cdot \eta_2 + 2 \cdot N$

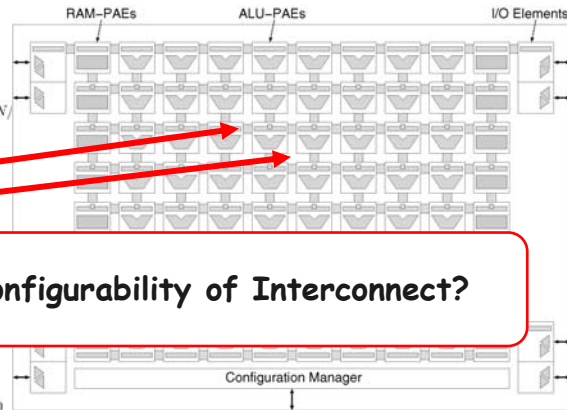
$$a[p, t] = \begin{cases} a[p, t - N/P] \\ a[p - 1, t - 1] \\ a[p + 1, t - 1] \end{cases}$$

$$u[p, t] = \begin{cases} A_{\eta_2} \\ u[p, t - N/P] \\ u[p - 1, t - 1] \\ u[p + 1, t - 1] \\ U_{\eta_1 + 2 \cdot p + 4 \cdot \eta_2} \end{cases}$$

$$z[p, t] = a[p, t] \cdot b[p, t]$$

$$y[p, t] = \begin{cases} y[p, t - 1] + z[p, t] & \text{if } \eta_2 > 0 \\ z[p, t] & \text{if } \eta_2 = 0 \end{cases}$$

$\rangle \rangle$   
 $\text{RETREAT}();$



**Reconfigurability of Interconnect?**

# Reconfigurability Challenges (3)

$P = \text{INVADE}(\text{my\_id}, \text{WEST}, N);$

$\langle \text{par} : 0 \leq p < P ::$

$\langle \text{seq} : t = N/P \cdot \eta_1 + (N/P + 1) \cdot \eta_2 + 2 \cdot N$

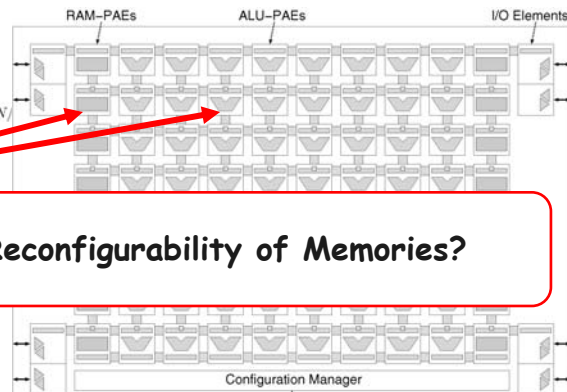
$$a[p, t] = \begin{cases} a[p, t - N/P] \\ a[p - 1, t - 1] \\ a[p + 1, t - 1] \end{cases}$$

$$u[p, t] = \begin{cases} A_{\eta_2} \\ u[p, t - N/P] \\ u[p - 1, t - 2] \\ u[p + 1, t - 2] \\ U_{\eta_1 + 2 \cdot p + 4 \cdot \eta_2 - \eta_2} \end{cases}$$

$$z[p, t] = a[p, t] \cdot b[p, t]$$

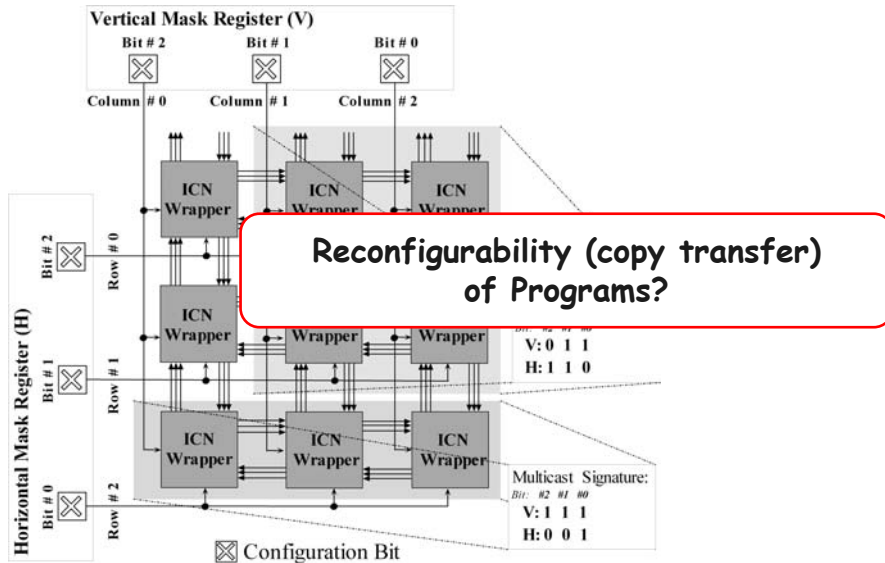
$$y[p, t] = \begin{cases} y[p, t - 1] + z[p, t] & \text{if } \eta_2 > 0 \\ z[p, t] & \text{if } \eta_2 = 0 \end{cases}$$

$\rangle \rangle$   
 $\text{RETREAT}();$



**Reconfigurability of Memories?**

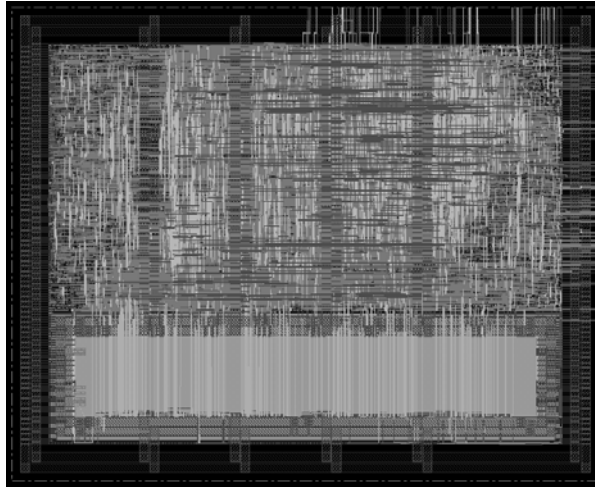
# Reconfigurability Challenges (4)



# Hardware design flow

- Cell Library:
  - Faraday Technology UMC 90nm Logic SP (LowK) process standard core cell library + memory compiler
- Synthesis:
  - Design Compiler
- Simulation:
  - ModelSim
- Place&Route, Power Estimation:
  - SoC Encounter
- Case study WPPE parametrization:
  - 16 Bit data path
  - 2 Adder modules
  - 1 Multiplier module
  - 16x128 Bit VLIW instruction memory macro cell

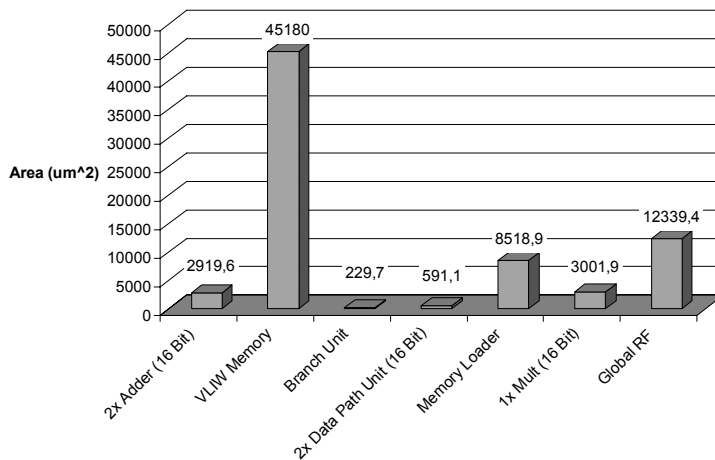
# Processing Element Design



- P&R results for one processing element  
(31 kGates, 74003.8  $\mu\text{m}^2$ )

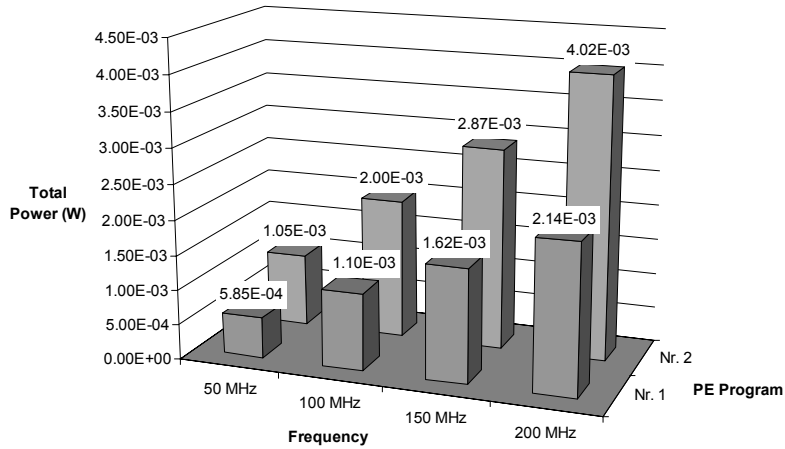
# Design Space Exploration

- Chip area breakdown for one processing (PE)

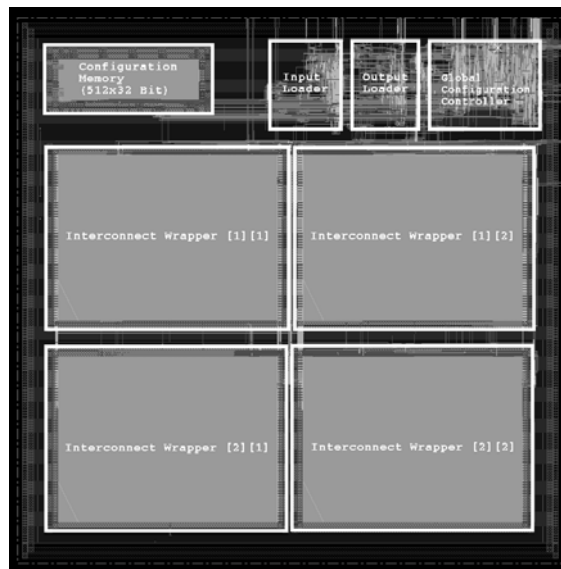


# Design Space Exploration

- Total power consumption for one processing element and different frequency/SW settings



# Example 2x2 array of processors



# Case Study Results

- Two algorithms implemented on both architectures:
  - Edge Detection (ED) and
  - FIR Filter
- 16-bit fixed point arithmetic
- Implementation on a Xilinx Virtex II Pro xc2vp30 FPGA
- Synthesis results:

Array type	# LUTs	# FFs	# BRAMs	# MULTs	Freq. (MHz)
Dedicated ED	1525	1449	5	4	250
Dedicated FIR Filter	312	470	-	4	277
InvasIC for both algorithms	4563	1493	8	4	150

# Case Study Results

- Size of reconfiguration data:

Array type	Reconfiguration data size (bytes)
Dedicated ED	72437
Dedicated FIR Filter	14489
WPPA ED	144
WPPA FIR Filter	40

- InvasIC Reconfiguration speed:  
(32 Bit reconfiguration bus @133 MHz)
  - Program & Interconnect FIR: 0.3  $\mu$ s
  - Program & Interconnect ED: 1  $\mu$ s

# Structure of Presentation

- Motivation
  - MPSoC Trends and Challenges in the nano area
  - Needs for and Potentials of Reconfigurability
  - New: Invasive Algorithms and Architectures
  
- Invasion: Hardware Challenges
  - Hardware/Software Reconfigurability
  - Decentralized Control of Resources
  
- Invasion: Algorithmic and Programming Challenges
  - Notation of commands supporting Invasion
  - Languages
  - Compilers
  
- Core research fields of the future

# Languages

- **Industry:**
  - C, C++, SystemC, (MATLAB) based design flow
  
- **Current state:**
  - 👍 Accepted standard for software development, specification and modeling
  - 👍 Software developers are already trained for these languages
  - 👍 Seamless design flow from specification down to implementation ⇒ time-to-market, flexibility
  - 👎 Only subsets of above languages are supported
  - 👎 Extensions: Pragmas, special macros and functions, or library based design approach
  - 👎 Difficult to parallelize
  - 👎 No notion for symbolic mapping



# Languages

## ➤ Academia:

Well defined models, e.g.,

- Mathematical models,
- Models of computation such as process networks,
- Functional programming languages

which allow for

- 👍 Verification/analysis of different properties
- 👍 Better parallelization techniques ⇒ higher performance
- 👎 Restricted to dedicated algorithm classes and domains
- 👎 Exotic, not widespread, no standard
- 👎 Project managers have to be convinced, engineers have to be trained
- 👎 Also: No notion of symbolic mappings allowing for efficient reconfiguration

# Programmability challenges

- Close (or reduction of) the gap between C-based and HDL-based designs
- Parallelization and mapping techniques for reconfigurable systems
- Need for Domain-specific approaches, i.e., applications such as
  - Streaming (audio, video, etc.)
  - High-performance computing (dynamic fluid simulation, protein folding, financial analytics, etc.)
  - Linear Algebra: (LU, QR, SVD, Least-square methods, Bilinear, Trilinear Interpolation, ...)
  - Video, audio, and other digital processing: (FIR, AR, Kalman Filter, ...)
  - Image Processing: (Median Filter, Hough Transformation, 2D convolution)
  - Reconstruction: (ART (Algebraic Reconstruction Technique), Kaczmarz, FBP (Filtered Back-Projection))
  - Edge Detection, Feature Extraction

# Phases and commands for invasion

- Invade  
P = invade(sender\_id, direction, constraints)
- Infect  
- copies local program to all invaded cells
- Execute
- Retreat

# Structure of Presentation

- Motivation
  - MPSoC Trends and Challenges in the nano area
  - Needs for and Potentials of Reconfigurability
  - New: Invasive Algorithms and Architectures
- Invasion: Hardware Challenges
  - Hardware/Software Reconfigurability
  - Decentralized Control of Resources
- Invasion: Algorithmic and Programming Challenges
  - Notation of commands supporting Invasion
  - Languages
  - Compilers
- Core research fields of the future

# Research Fields (1)

- Algorithm research:
  - Algorithm composition
  - Self-restricted invasion
  - Applications and complexity of invasion
  - Dynamic computation graphs
  - CombatWhat happens if two or more algorithms invade simultaneously processors?

# Research Fields (2)

- Architectural research - InvasICs:
  - Microarchitectures for invasive programming
  - Instruction set definition and implementation for invasive commands
  - Communication network design
  - Control organization
    - Who decides where to place seeds?
    - Who decides and how to restrict invasion
    - Centralized vs. Decentralized (hierarchical) control
  - Quantitative cost analysis

## Research Fields (3)

- Compiler and tool support:
  - Simulation
  - Compiler
    - Loop Invader
    - Machine Markup Languages
  
- Operating System Concepts
  - Resource negotiation protocols
  - Concepts for immunity (against) invaders
  - Control issues
  - Implementation

## Research Fields (4)

- Applications:
  - Mobile Robots
  - Video Image Processing
  - Bioinformatics
  - ...

# Conclusions

- Introduction of a new paradigm for parallel programming: Invasion and corresponding MPSoC architectures: InvasICs
- Cost/flexibility trade-off:
  - **Dedicated Processor Arrays**
    - ☞ Small area, high-speed
    - ☞ Low power
    - ☞ Too specific
  - **Standard processor multi-core architectures**
    - ☞ lack of reconfigurability of interconnect, memory banks, and instructions to support invasion
    - ☞ suffer from global and centralized control
    - ☞ suffer from high power consumption
  - **InvasICs**
    - ☞ Highly parameterizable
    - ☞ Extremely fast reconfiguration in real time
    - ☞ Support for scalability and dynamic use of resources
    - ☞ Ultra-low power

# Questions?

*Invasion -  
A New Paradigm for  
Parallel Algorithms and Architectures*

