SIEMENS

Corporate Technology

Kernfragen:
Multicore-Prozessoren in der Industrie

InvasIC-Kolloquium, FAU Erlangen
10. Februar 2012

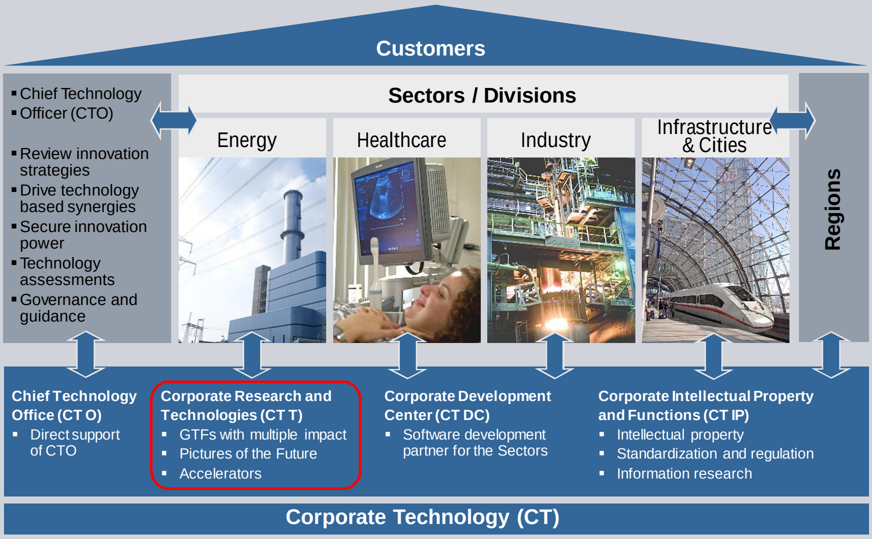
Urs Gleim
Siemens AG – Corporate Technology
System Architecture & Platforms
Parallel Processing Systems Program
urs.gleim@siemens.com

System
Architecture
& Platforms

Introduction

Corporate Technology

SIEMENS



Customers

- Chief Technology Officer (CTO)
- Review innovation strategies
- Drive technology based synergies
- Secure innovation power
- Technology assessments
- Governance and guidance

Sectors / Divisions

Energy | Healthcare | Industry | Infrastructure & Cities

Regions

Chief Technology Office (CT O)
- Direct support of CTO

Corporate Research and Technologies (CT T)
- GTFs with multiple impact
- Pictures of the Future
- Accelerators

Corporate Development Center (CT DC)
- Software development partner for the Sectors

Corporate Intellectual Property and Functions (CT IP)
- Intellectual property
- Standardization and regulation
- Information research

Corporate Technology (CT)

Technical Challenges

Vision

Strong Focus on Embedded Systems

Industrial control systems

- e.g. rolling mill

Transportation systems

- e.g. railway, car

Medical equipment

- e.g. magnetic resonance imaging

Communication systems

- e.g. network switch

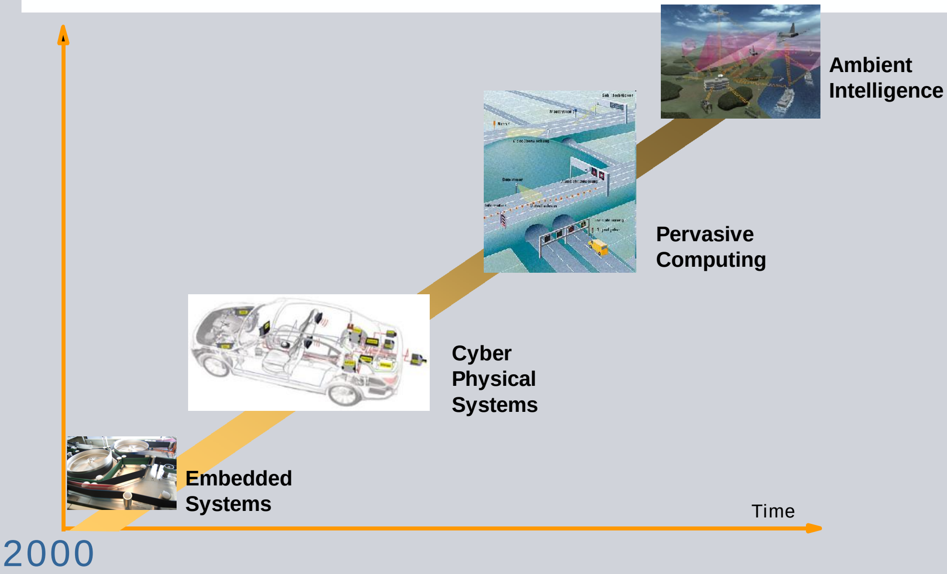
Energy management

- e.g. smart meter

Vision

Future Evolution in General

SIEMENS

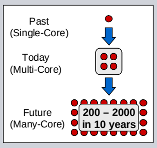


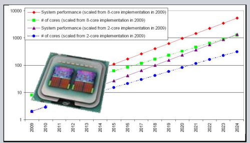
Vision

Enabling Technologies

SIEMENS

Multicore processors will provide still increasing computing power, even in embedded environments with electric power limitations.

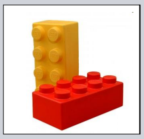




source: International Roadmap for Semiconductors 2009 (http://www.itrs.net/)

Increasing **semiconductor integration** levels provide powerful on-chip networking facilities.

Progress in **wireless communication** allows for flexible interconnect topologies.



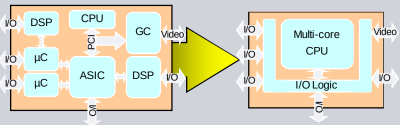
Hardware and software **commoditization** and Open Source provide powerful platforms.

Challenges
1. Consolidation

SIEMENS

Hardware Consolidation

- reduced number of controllers and DSPs
- increased flexibility due to software solutions

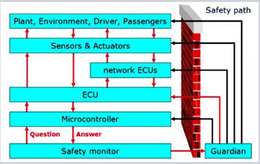


Real-time system architectures

- ensure real-time behavior with partitioning and virtualization technologies

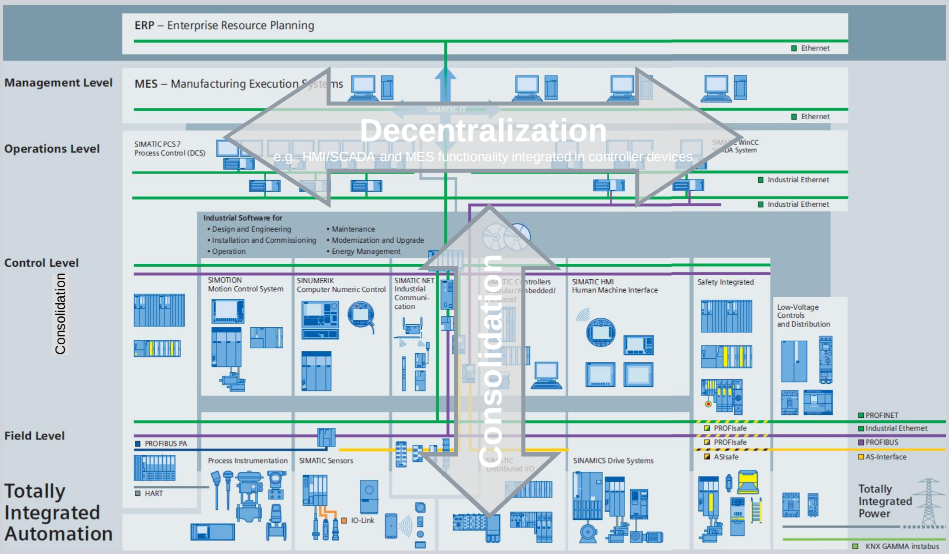
Architectures for mixed critical systems

- separation of safety-relevant subsystems on the same processor
- Efficient development and evolvability of safety-critical systems (e.g., independent certification of safety-critical (software) components)
- Dependability in open systems



Challenges
2. Decentralization

SIEMENS



Challenges

3. Heterogeneity

Heterogeneous multi-/many-core architectures

- Utilization of special purpose cores of multicore processors (portable programming models, load balancing)

Hardware accelerators

- Optional acceleration units (e.g., GPUs)

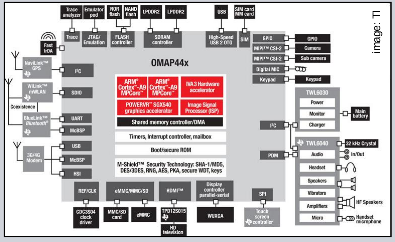


image: TI

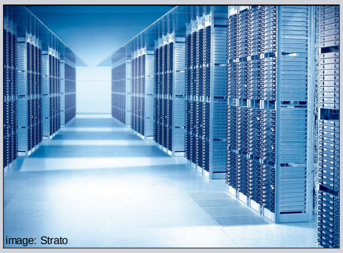


image: Strato

Cloud computing

- Flexible deployment
- Scalability of resources
- Communication bottleneck
- Security and data privacy

Challenges

4. Security

„Der digitale Erstschlag ist erfolgt.“
Frankfurter Allgemeine Zeitung

The more complex and interconnected a system is, the bigger the number of security vulnerabilities: We have to defend against cyber-attacks.

Challenges
5. Energy Management

Power Efficiency

- Mobility: Energy consumption of on-board electronics must be minimal. (public transportation, eCar)
- Mobile devices driven by battery or energy harvesting (e.g., healthcare in rural areas with unreliable energy supply)
- limited installation space in industrial devices or energy management (waste heat problem)

➔ Universal energy management architecture needed!

Challenges
6. Programming Models

For parallel hardware architectures today's Programming Models are

- too complex
- error prone
- not scaling with number of processing units
- non-deterministic

We need programming models that are

- suitable for the masses "taking parallelism mainstream" (Microsoft)
- development efficiency comparable to sequential software development
- abstraction from hardware architectures to the greatest possible extent
- compatibility to common programming languages (huge code base available)
- flat learning curve for developers

Challenges
7. Migration Strategies

SIEMENS

Parallel processing units enable

- doing more → data volume increases constantly
- doing faster → interactive work with IT systems in healthcare and industry (simulation)

huge code base of sequential code to be parallelized

- where to start?
- how to parallelize?
- how to ensure correctness?

Summary
The 7 Challenges of Embedded Software Development

SIEMENS

1. Consolidation

- shift from HW to SW
- utilization of multi-/many-core systems
- taking into account safety and real-time requirements

2. Decentralization

- flexible deployment of functionality in distributed systems

3. Heterogeneity

- heterogeneous multi-/many-core architectures
- hardware accelerators
- cloud computing

4. Security

- data privacy
- protection against manipulation

5. Energy management

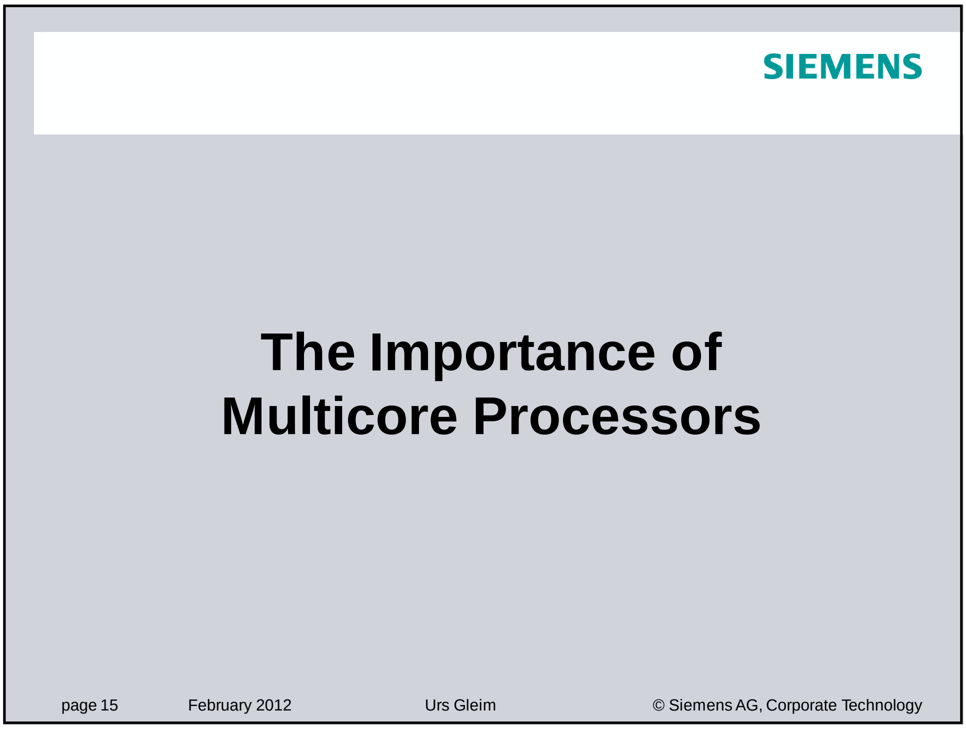
- power-efficient hard- and software

6. Programming models

- Development efficiency and future-proofness
- Portability, HW-independence
- Scalability with processing power (more cores)

7. Migration strategies

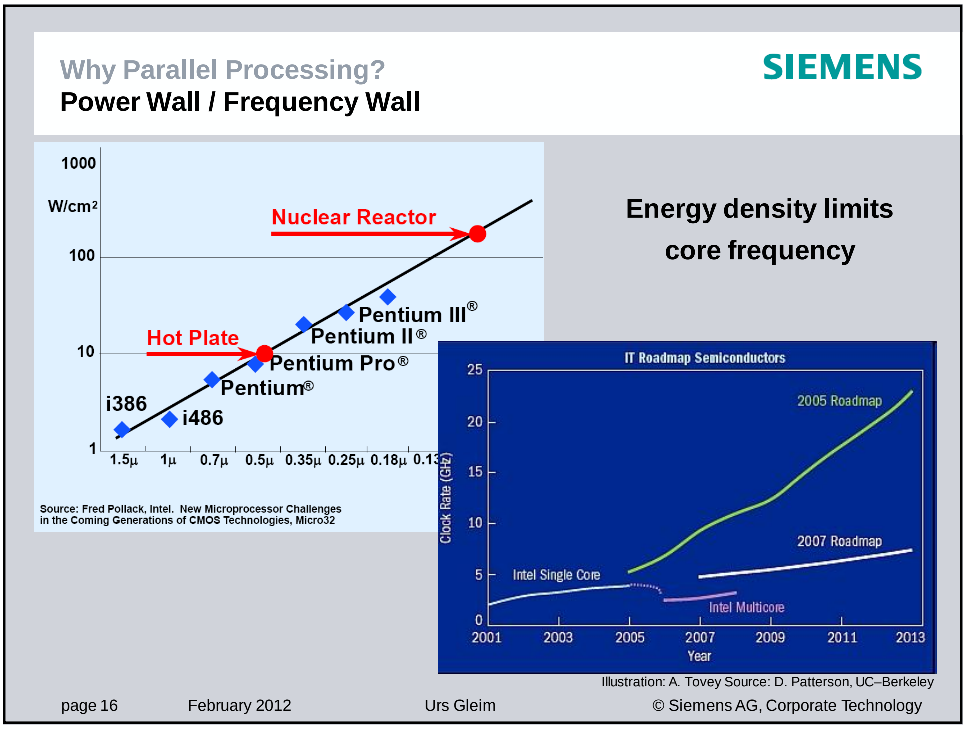
- utilize parallel hardware preserving existing code bases



SIEMENS

The Importance of Multicore Processors

page 15 February 2012 Urs Gleim © Siemens AG, Corporate Technology



SIEMENS

Why Parallel Processing?

Power Wall / Frequency Wall

Source: Fred Pollack, Intel. New Microprocessor Challenges in the Coming Generations of CMOS Technologies, Micro32

Energy density limits core frequency

Illustration: A. Tovey Source: D. Patterson, UC–Berkeley

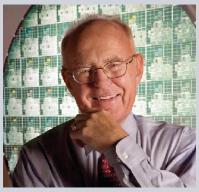
page 16 February 2012 Urs Gleim © Siemens AG, Corporate Technology

Why Parallel Processing?
Moore's Law

SIEMENS

- Moore's Law

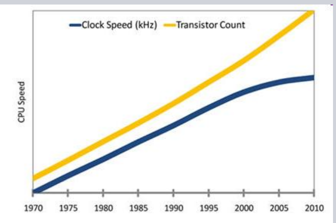
The number of transistors on an integrated circuit increases exponentially, doubling approximately every two years.



- Intel predicts this trend through 2029
(IDF 2008)

- What does this mean for multi-core CPUs?

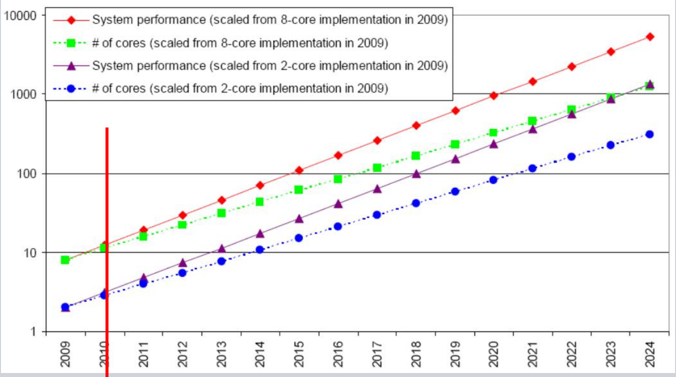
200-2000 cores/CPU in mass market in 10 years!



Why Parallel Processing?
Rapidly Growing Number of Cores

SIEMENS

Expected evolution of networking SoCs:
- Number of cores increases by 1.4 × / year
- Core frequency increases by 1.05 × / year



Reality Check
(summer 2010)

- Standard x86 based 48 core Server:
6-12 cores/CPU,
up to 4 CPUS

- Embedded CPUs:
2-4 cores

source: International Roadmap for Semiconductors 2009 (http://www.itrs.net/)

Why Parallel Processing?
How does Hardware Evolve?

SIEMENS

Example Calxeda's EnergyCard

- 4 quad-core ARM Cortex-A9 cores

image: http://www.windowsfordevices.com/c/a/News/Calxeda-ECX1000-and-HP-Redstone/

➔ 16 cores

Why Parallel Processing?
How does Hardware Evolve?

SIEMENS

HP's Redstone servers

- 18 of Calxeda's Energy Cards

➔ 72 cores

image: HP, http://www.windowsfordevices.com/c/a/News/Calxeda-ECX1000-and-HP-Redstone/

Why Parallel Processing?
How does Hardware Evolve?

SIEMENS

Rack HP's Redstone servers

- 4 x 18 cards

➔ 288 cores

- up to 3000 chips in one rack

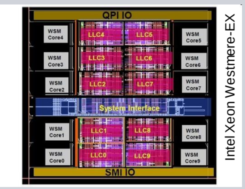
➔ 12.000 cores

image: HP, http://www.windowsfordevices.com/c/a/News/Calxeda-ECX1000-and-HP-Redstone/

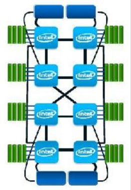
Why Parallel Processing?
Trends

SIEMENS

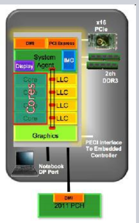
more CPU cores (SMP)



more processors (SMP)



integrated graphics and chipset features



SIMD instructions (ILP)

heterogeneous cores (AMP)
HW accelerators, DSPs

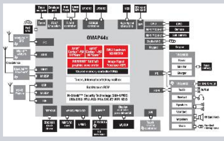
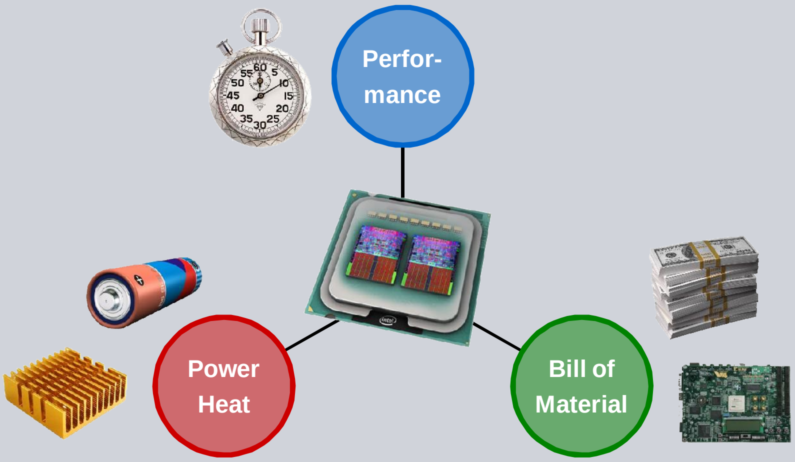


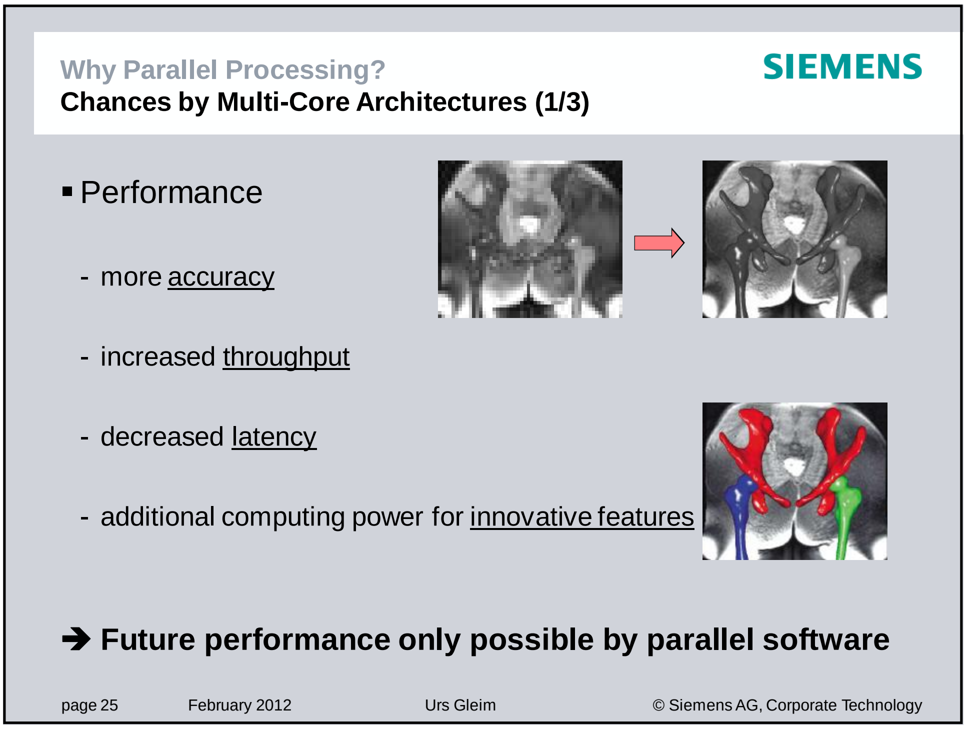
image: TI

Bilder: Intel, wenn nicht anders angegeben

What can we do with Multicore-Processors?

Benefits of Parallel Architectures





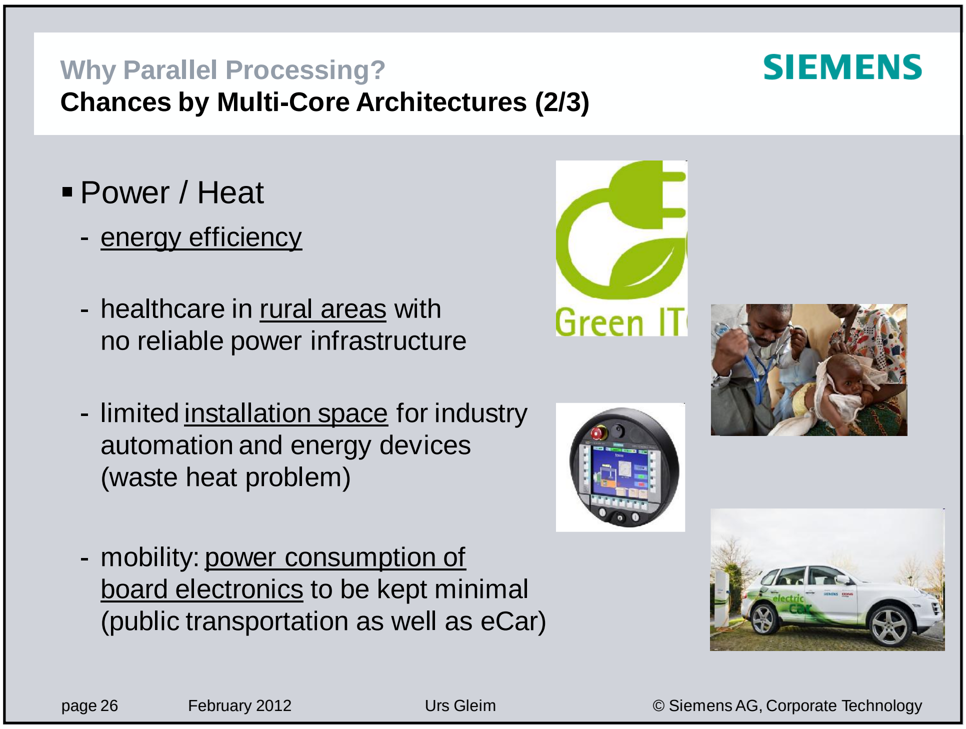
Why Parallel Processing?

Chances by Multi-Core Architectures (1/3)

SIEMENS

- Performance
  - more accuracy
  - increased throughput
  - decreased latency
  - additional computing power for innovative features

➔ **Future performance only possible by parallel software**



Why Parallel Processing?

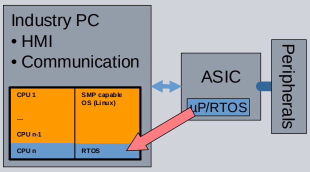
Chances by Multi-Core Architectures (2/3)

SIEMENS

- Power / Heat
  - energy efficiency
  - healthcare in rural areas with no reliable power infrastructure
  - limited installation space for industry automation and energy devices (waste heat problem)
  - mobility: power consumption of board electronics to be kept minimal (public transportation as well as eCar)

Why Parallel Processing?
Chances by Multi-Core Architectures (3/3)

- BoM
  - lower production costs by replacing specialized processing HW by general purpose processors
  - use cheaper sensor technology and actuating elements and compensate quality loss by intelligent software
  - high performance low cost products

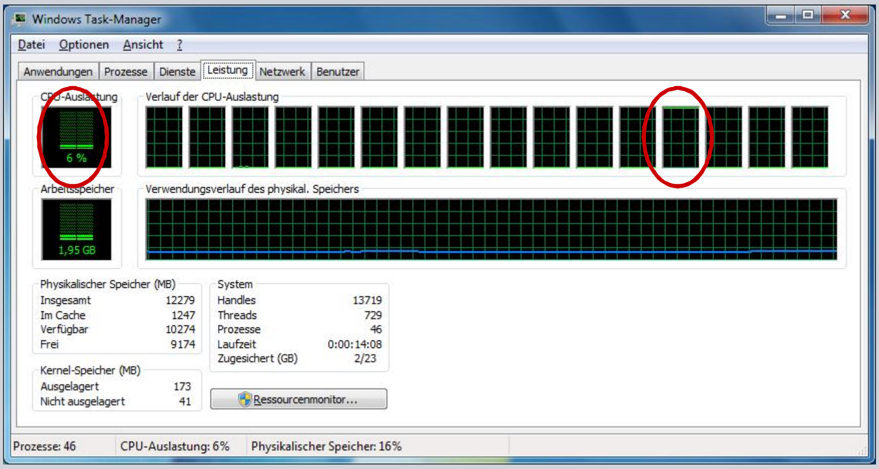


Our Work

Consequences for Software Development

The Free Lunch is Over*

SIEMENS



Windows Task-Manager
Datei Optionen Ansicht ?
Anwendungen Prozesse Dienste Leistung Netzwerk Benutzer
CPU-Auslastung
6 %
Verlauf der CPU-Auslastung
Arbeitsspeicher
1,95 GB
Verwendungsverlauf des physikal. Speichers
Ressourcenmonitor...

Physikalischer Speicher (MB)	
Insgesamt	12279
Im Cache	1247
Verfügbar	10274
Frei	9174

System	
Handles	13719
Threads	729
Prozesse	46
Laufzeit	0:00:14:08
Zugesichert (GB)	2/23

Kernel-Speicher (MB)	
Ausgelagert	173
Nicht ausgelagert	41

Prozesse: 46 CPU-Auslastung: 6% Physikalischer Speicher: 16%

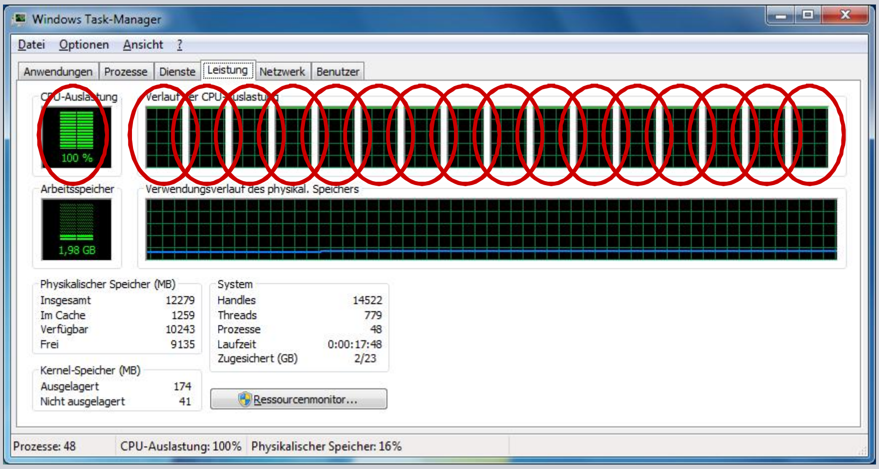
Today's software!

* Herb Sutter; The free lunch is over: A fundamental turn toward concurrency in software"; Dr. Dobb's Journal, 30(3), 2005

Consequences for Software Development

Goal

SIEMENS



Windows Task-Manager
Datei Optionen Ansicht ?
Anwendungen Prozesse Dienste Leistung Netzwerk Benutzer
CPU-Auslastung
100 %
Verlauf der CPU-Auslastung
Arbeitsspeicher
1,98 GB
Verwendungsverlauf des physikal. Speichers
Ressourcenmonitor...

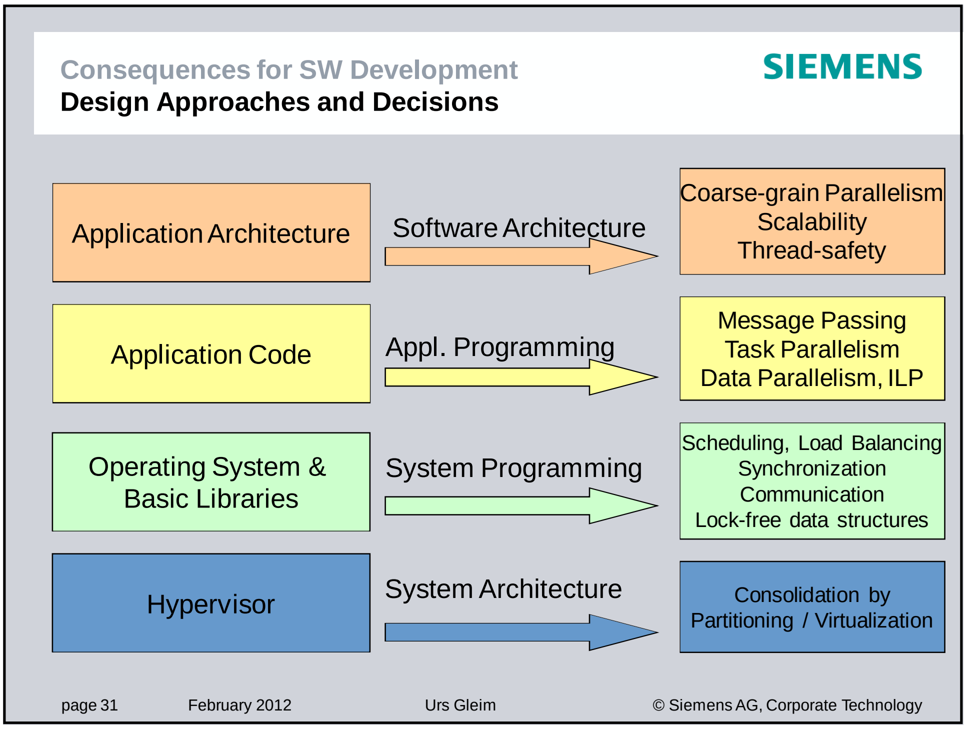
Physikalischer Speicher (MB)	
Insgesamt	12279
Im Cache	1259
Verfügbar	10243
Frei	9135

System	
Handles	14522
Threads	779
Prozesse	48
Laufzeit	0:00:17:48
Zugesichert (GB)	2/23

Kernel-Speicher (MB)	
Ausgelagert	174
Nicht ausgelagert	41

Prozesse: 48 CPU-Auslastung: 100% Physikalischer Speicher: 16%

Utilization of all core ➔ only possible with parallel sofware!

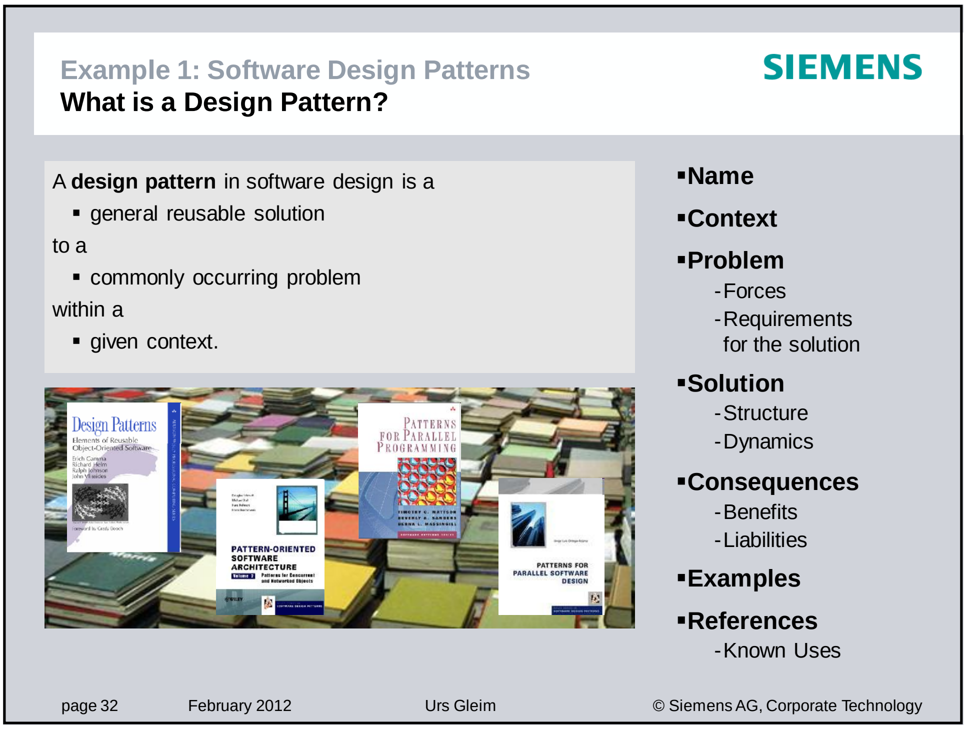


Consequences for SW Development

Design Approaches and Decisions

SIEMENS

Application Architecture	Software Architecture →	Coarse-grain Parallelism Scalability Thread-safety
Application Code	Appl. Programming →	Message Passing Task Parallelism Data Parallelism, ILP
Operating System & Basic Libraries	System Programming →	Scheduling, Load Balancing Synchronization Communication Lock-free data structures
Hypervisor	System Architecture →	Consolidation by Partitioning / Virtualization



Example 1: Software Design Patterns

What is a Design Pattern?

SIEMENS

A **design pattern** in software design is a

- general reusable solution

to a

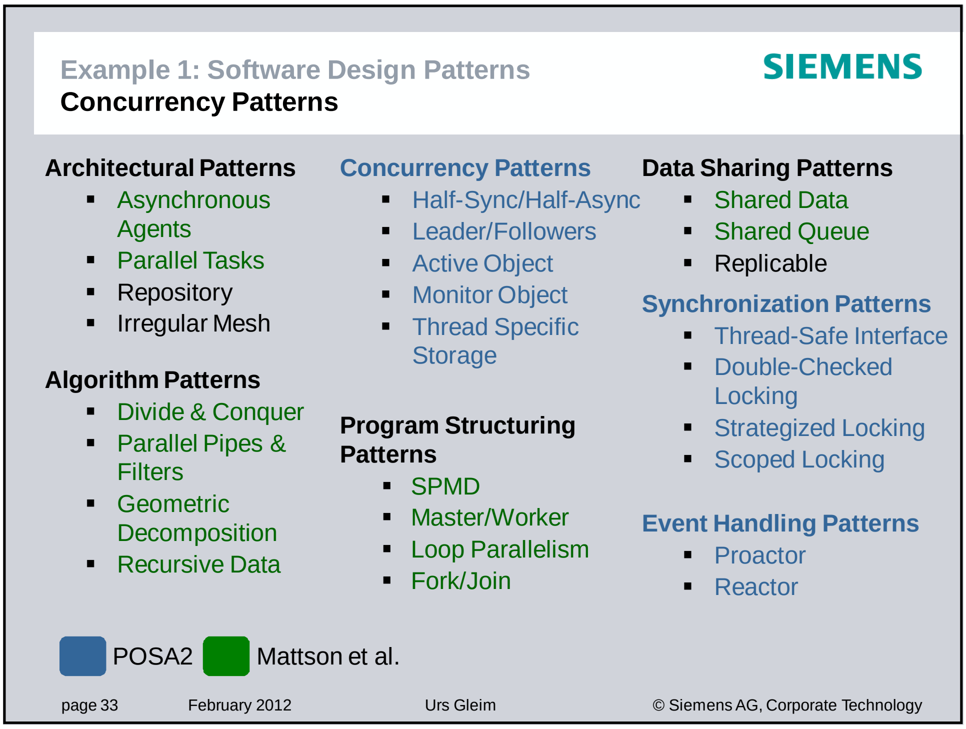
- commonly occurring problem

within a

- given context.

Design Patterns
Elements of Reusable Object-Oriented Software
Pattern-Oriented Software Architecture
Patterns for Parallel Programming
Patterns for Parallel Software Design

- **Name**
- **Context**
- **Problem**
  - Forces
  - Requirements for the solution
- **Solution**
  - Structure
  - Dynamics
- **Consequences**
  - Benefits
  - Liabilities
- **Examples**
- **References**
  - Known Uses



Example 1: Software Design Patterns

Concurrency Patterns

SIEMENS

Architectural Patterns

- Asynchronous Agents
- Parallel Tasks
- Repository
- Irregular Mesh

Algorithm Patterns

- Divide & Conquer
- Parallel Pipes & Filters
- Geometric Decomposition
- Recursive Data

Concurrency Patterns

- Half-Sync/Half-Async
- Leader/Followers
- Active Object
- Monitor Object
- Thread Specific Storage

Program Structuring Patterns

- SPMD
- Master/Worker
- Loop Parallelism
- Fork/Join

Data Sharing Patterns

- Shared Data
- Shared Queue
- Replicable

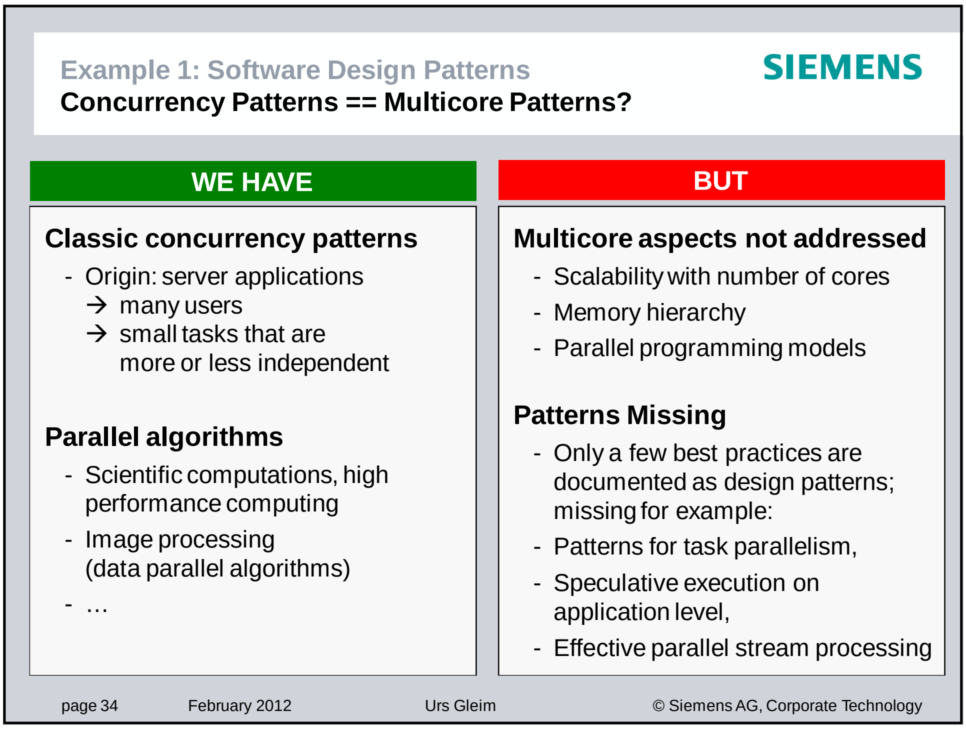
Synchronization Patterns

- Thread-Safe Interface
- Double-Checked Locking
- Strategized Locking
- Scoped Locking

Event Handling Patterns

- Proactor
- Reactor

🟦 POSA2 🟩 Mattson et al.



Example 1: Software Design Patterns

Concurrency Patterns == Multicore Patterns?

SIEMENS

WE HAVE

Classic concurrency patterns

- Origin: server applications
  → many users
  → small tasks that are more or less independent

Parallel algorithms

- Scientific computations, high performance computing
- Image processing (data parallel algorithms)
- …

BUT

Multicore aspects not addressed

- Scalability with number of cores
- Memory hierarchy
- Parallel programming models

Patterns Missing

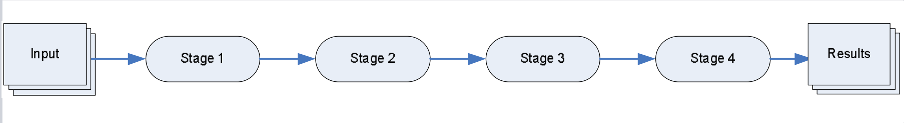
- Only a few best practices are documented as design patterns; missing for example:
- Patterns for task parallelism,
- Speculative execution on application level,
- Effective parallel stream processing

Example 1: Software Design Patterns

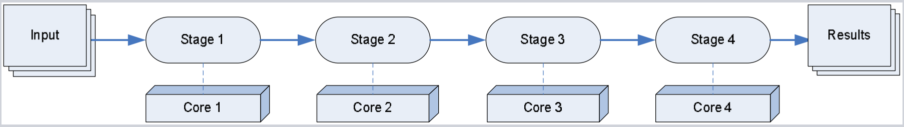
Example: Stream Processing Pipleline

SIEMENS

Pipeline:



Parallel pipeline boosts throughput (*Parallel Pipes-and-Filters* pattern):



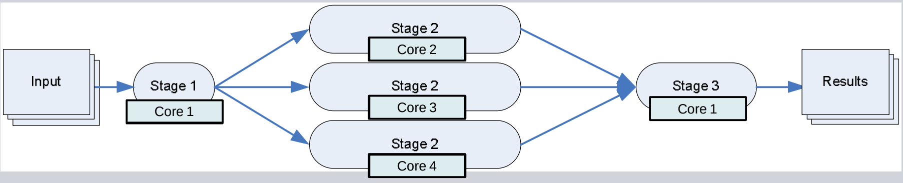
➔ no load balancing
➔ does not scale
➔ synchronization overhead for data transfer
➔ bad locality (memory hierarchy, cache effects)

Example 1: Software Design Patterns

Example: Parallel Pipeline Stages

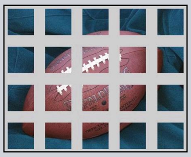
SIEMENS

Improvement of load balancing and throughput:



Multiple instances of long-running stages
➔ scalability still limited
➔ order gets lost, reordering might be necessary

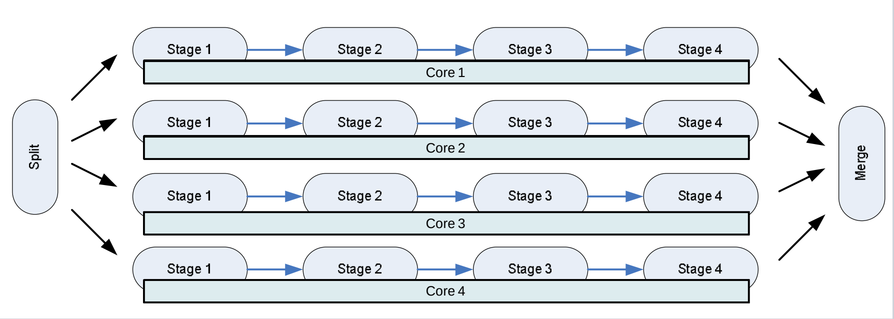
Inside stages, data parallelism may be used
➔ improved latency



Example 1: Software Design Patterns

Example: Mutliple Sequential Pipelines

Multiple sequential pipelines in parallel:



➔ good load balancing
➔ good cache locality
➔ but, order gets lost, reordering necessary

Example 1: Software Design Patterns

Conclusion

➔ Find a scalable partitioning of the problem

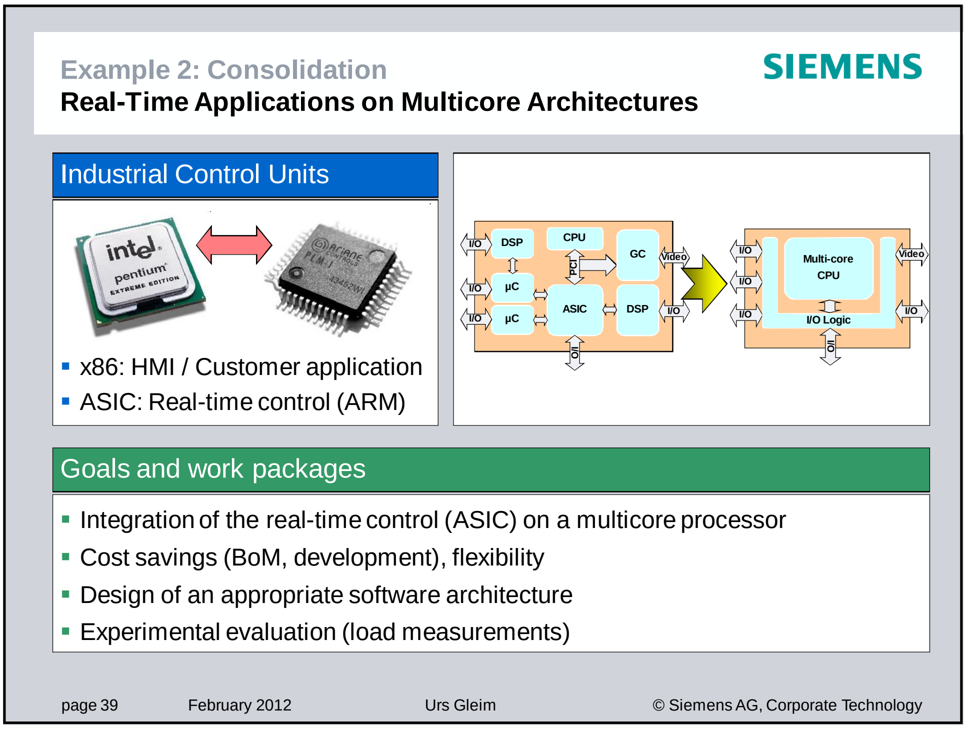
- Architecture should support load balancing
- Parallelism should be scalable with number of cores
- Avoid waiting times

➔ Keep data local

- bad locality can slow down an application massively (costs for data transfer, false sharing)
- No complicated architecture needed

➔ Parallel execution can change the processing order

- Only possible if no dependencies between data elements
- Additional effort for restoring order


DSP
CPU
GC
µC
PCI
ASIC
DSP
µC
I/O
I/O
I/O
Video
I/O
I/O
I/O
I/O
I/O
Multi-core
CPU
I/O Logic
Video
I/O
I/O
intel
pentium
EXTREME EDITION
ARIANE
PLM-1

SIEMENS

Example 2: Consolidation
Real-Time Applications on Multicore Architectures

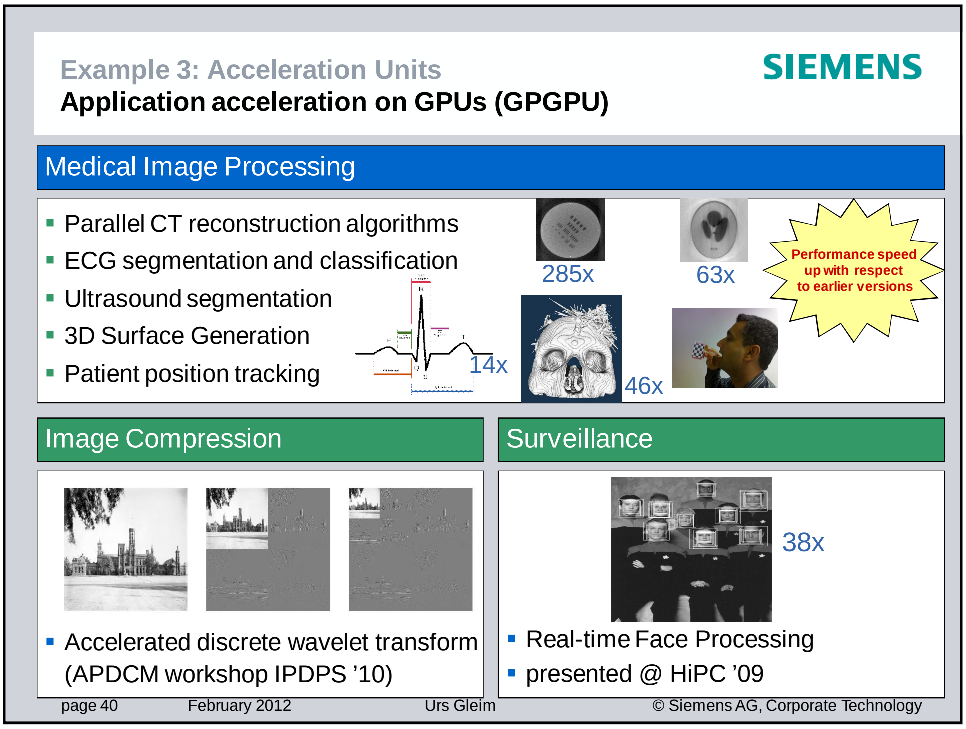
Industrial Control Units

- x86: HMI / Customer application
- ASIC: Real-time control (ARM)

Goals and work packages

- Integration of the real-time control (ASIC) on a multicore processor
- Cost savings (BoM, development), flexibility
- Design of an appropriate software architecture
- Experimental evaluation (load measurements)

page 39 February 2012 Urs Gleim © Siemens AG, Corporate Technology


285x
63x
14x
46x
Performance speed
up with respect
to earlier versions
38x

SIEMENS

Example 3: Acceleration Units
Application acceleration on GPUs (GPGPU)

Medical Image Processing

- Parallel CT reconstruction algorithms
- ECG segmentation and classification
- Ultrasound segmentation
- 3D Surface Generation
- Patient position tracking

Image Compression

- Accelerated discrete wavelet transform (APDCM workshop IPDPS '10)

Surveillance

- Real-time Face Processing
- presented @ HiPC '09

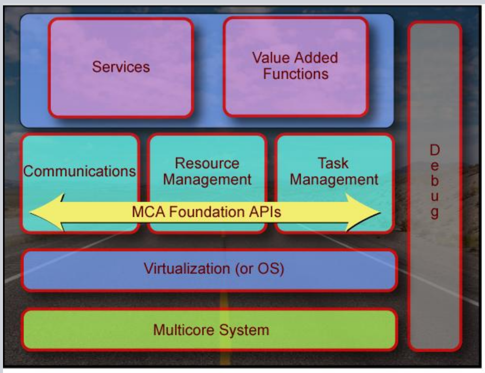
page 40 February 2012 Urs Gleim © Siemens AG, Corporate Technology

Example 4: MTAPI
The Multicore Association (www.multicore-association.org)

SIEMENS

Support for Complete System Design

- Improve time to market for applications through the use of standards
- MCA foundation APIs provide infrastructure to support other multicore services and value-added functions
- Communications and Resource Management are now in place; MTAPI will complete the foundation APIs to enable end-to-end multicore system development



MCA Foundation APIs

Communications (MCAPI)
- Lightweight messaging

Resource Management (MRAPI)
- Basic synchronization
- Shared/Distributed Memory
- System Metadata

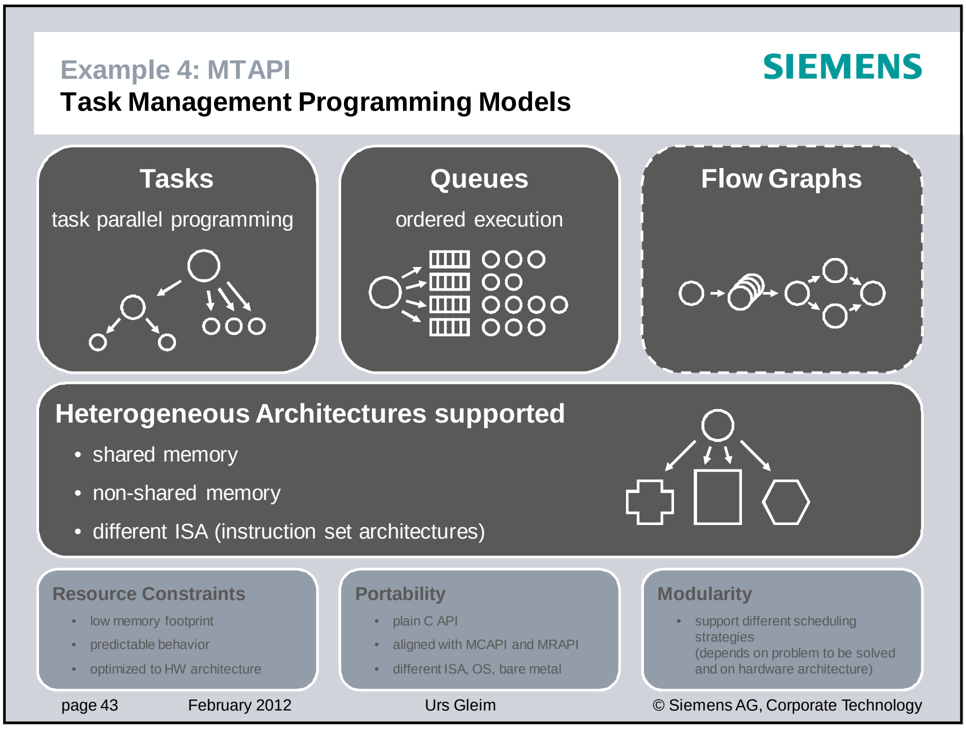
Task Management (MTAPI)
- Task lifecycle
- Task placement
- Task synchronization

Example 4: MTAPI
MTAPI Working Group – Multicore Association

SIEMENS

Multicore Association Board Members
AMD
EADS NORTH AMERICA
freescale
intel
Mentor Graphics
Nokia Siemens Networks
PolyCore Software
PLURALITY
SAMSUNG
Texas Instruments
WIND RIVER
Multicore Association Working Group Members
ARGON DESIGN
CAPS
CAVIUM NETWORKS
codeplay
criticalblue
EfficiOS
ENEA
eSOL Embedded Excellence
imec
LG
LSI
MIPS TECHNOLOGIES
NATIONAL INSTRUMENTS
nCore
NETLOGIC
NETRONOME
ONEACCESS
QUALCOMM
RadiSys
SAGE
SIEMENS
TILERA
UltraSoC Technologies
Multicore Association University Members
Åbo Akademi University
Carnegie Mellon
TU Delft
Ecole Polytechnique Montréal
Technikum Wien
UH
University of Tsukuba
THE Multicore ASSOCIATION

Green highlighted companies are active participants in MTAPI WG



Example 4: MTAPI

Task Management Programming Models

SIEMENS

Tasks

task parallel programming

Queues

ordered execution

Flow Graphs

Heterogeneous Architectures supported

- shared memory
- non-shared memory
- different ISA (instruction set architectures)

Resource Constraints

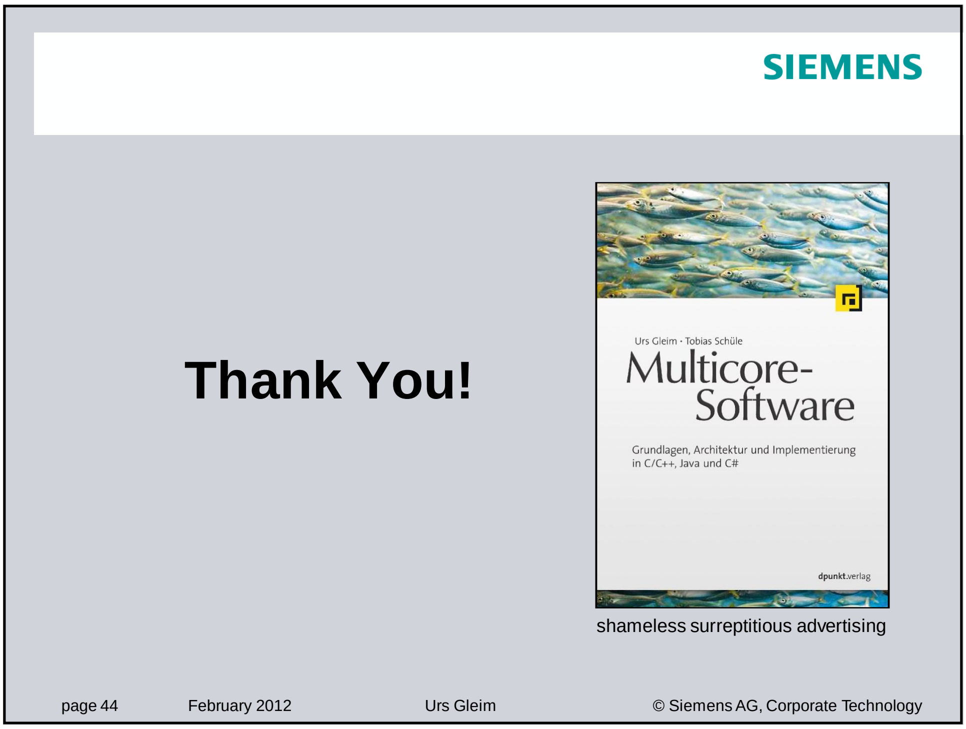
- low memory footprint
- predictable behavior
- optimized to HW architecture

Portability

- plain C API
- aligned with MCAPI and MRAPI
- different ISA, OS, bare metal

Modularity

- support different scheduling strategies (depends on problem to be solved and on hardware architecture)



SIEMENS

Thank You!

Urs Gleim · Tobias Schüle
Multicore-Software
Grundlagen, Architektur und Implementierung in C/C++, Java und C#
dpunkt.verlag

shameless surreptitious advertising