Hardware/Software Co-Design of Signal Processing Systems

R. Ernst

Technical University of Braunschweig

Germany

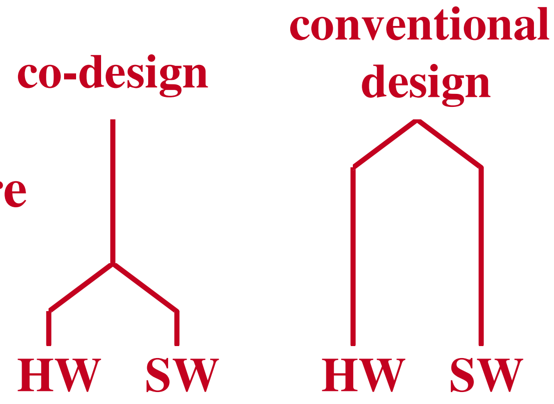
0 Overview

- introduction
- hardware/software co-design
- hardware/software co-synthesis
  - principles and applications
  - problems
- flexibility in the design process - a fundamental problem
- conclusions

1 Introduction

- challenges in the design of signal processing systems
  - increasing application complexity
    - combination of different signal processing functions
    - combination data flow and reactive tasks
    - flexibility requirements
  - increasing target system complexity
    - mixture of different technologies, processor types, and design styles
    - large systems-on-a-chip combining components from different sources (IP market, reuse)
  - numerous constraints and design objectives

2 Hardware/software co-design



- hardware/software co-design:
  Combined design of hardware and software

- goals
  - design process optimization
    $\Rightarrow$ Increased design productivity
  - design optimization
    $\Rightarrow$ Improved product quality

- tasks
  - co-specification and co-modeling
  - co-verification
  - co-design process integration and optimization
  - design optimization and co-synthesis

Hardware/software co-design cont'd

- **all co-design problems require computer support**

⇒ **computer aided hardware/software co-design**
is the new challenge

Hardware/software co-design: Industrial practice

- co-simulation and executable co-specification are the most advanced topics in computer aided hardware/software co-design

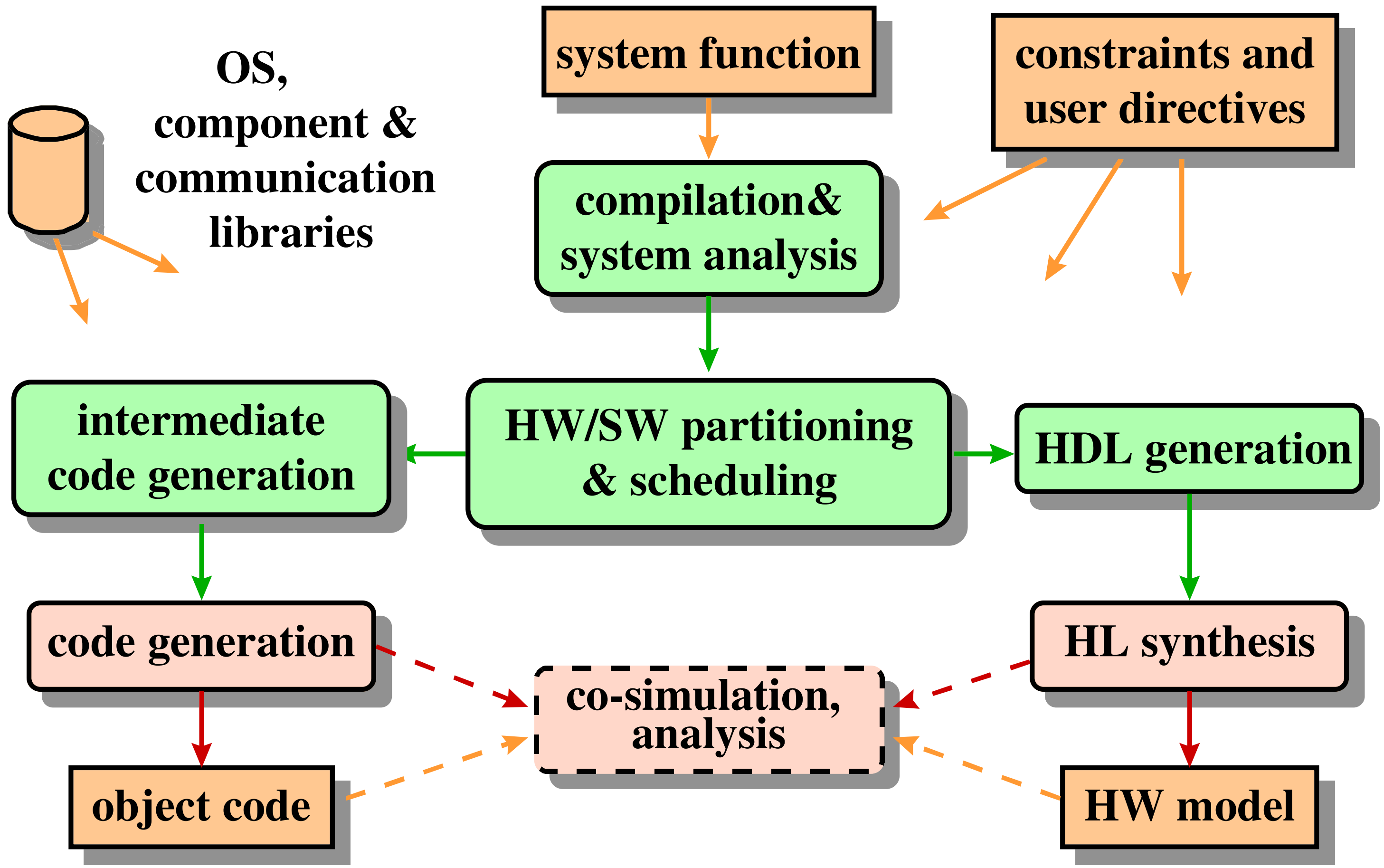
  → corresponds to immediate industrial needs of a concurrent engineering environment

Hardware/software co-design: Industrial practice cont'd

- **co-synthesis, optimization and design space exploration are far less developed**
  - **co-synthesis is a complex task**
    - **software synthesis and code generation - lack of efficiency**
    - **hardware synthesis - limited architecture**
    - **interface and communication synthesis - protocol limitations**
    - **hardware/software partitioning and component selection - nil**
    - **hardware/software scheduling - restricted mechanisms**

    ⇒ **back-end support still immature**

3 Hardware/software co-synthesis

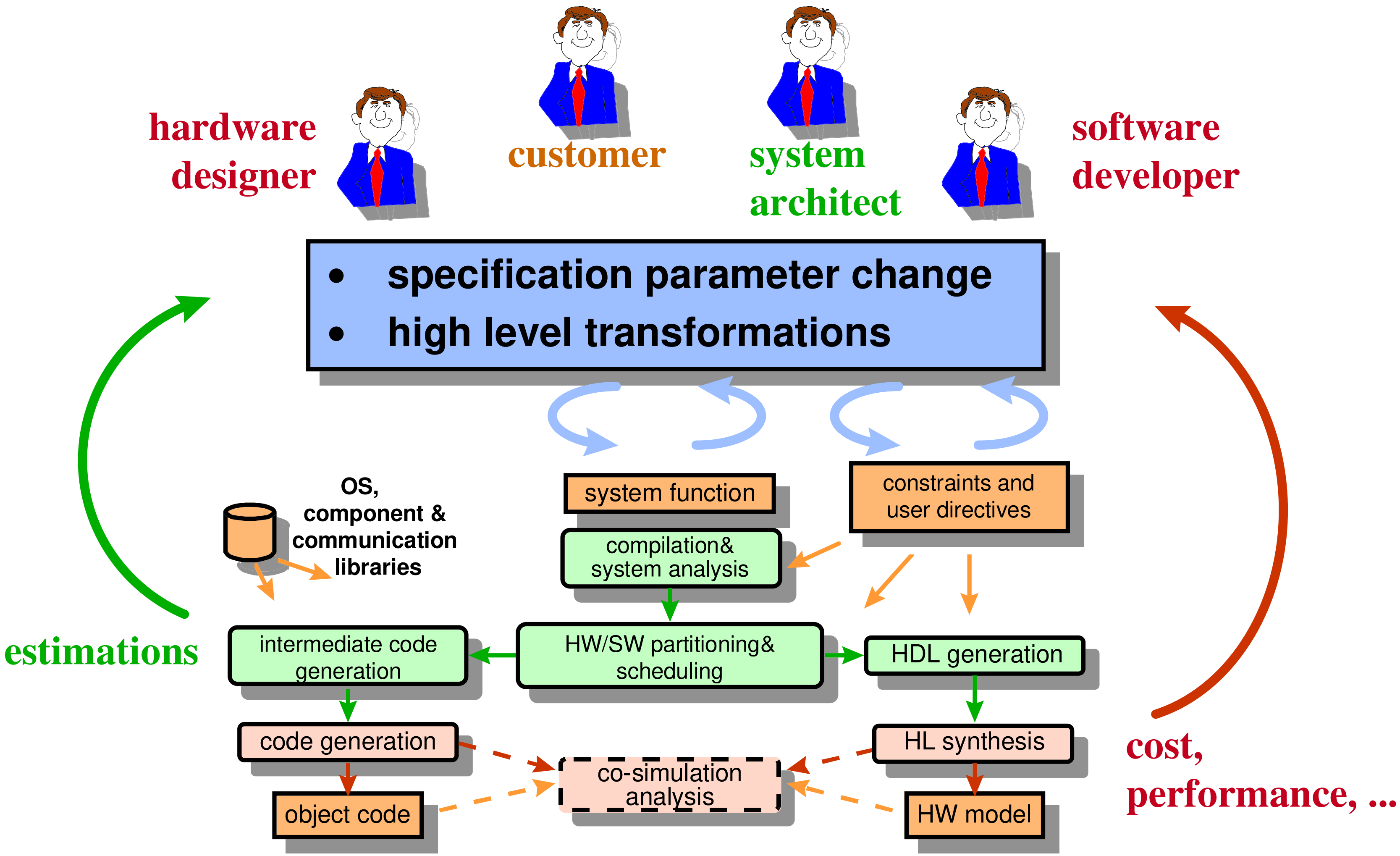


Co-synthesis design flow - Principle

Synthesis and user directives - Overview

- **hardware synthesis**
  - HL synthesis of new components
  - HW interface and communication synthesis synthesis for new and predefined complex components (processors, reuse)
  - HW scheduling and (global) control synthesis
- **software synthesis**
  - code generation
  - SW interface synthesis
  - SW scheduling (operating system)
- **user directives**
  - preselection of HW and SW components and communication channels
  - preselected mapping of input processes to components and communication to channels
  - constraint specification (see e.g. [EB94])

Co-design with design space exploration using co-synthesis



28.10.97

Specification parameter changes and high-level transformations - Overview

- **specification parameter changes**

  examples

  - modify performance requirements (throughput/response)
  - add/remove “nice-to-have” features
  - modify product scope (i.e. adapt market scope)
  - reduce/extend precision (word length)

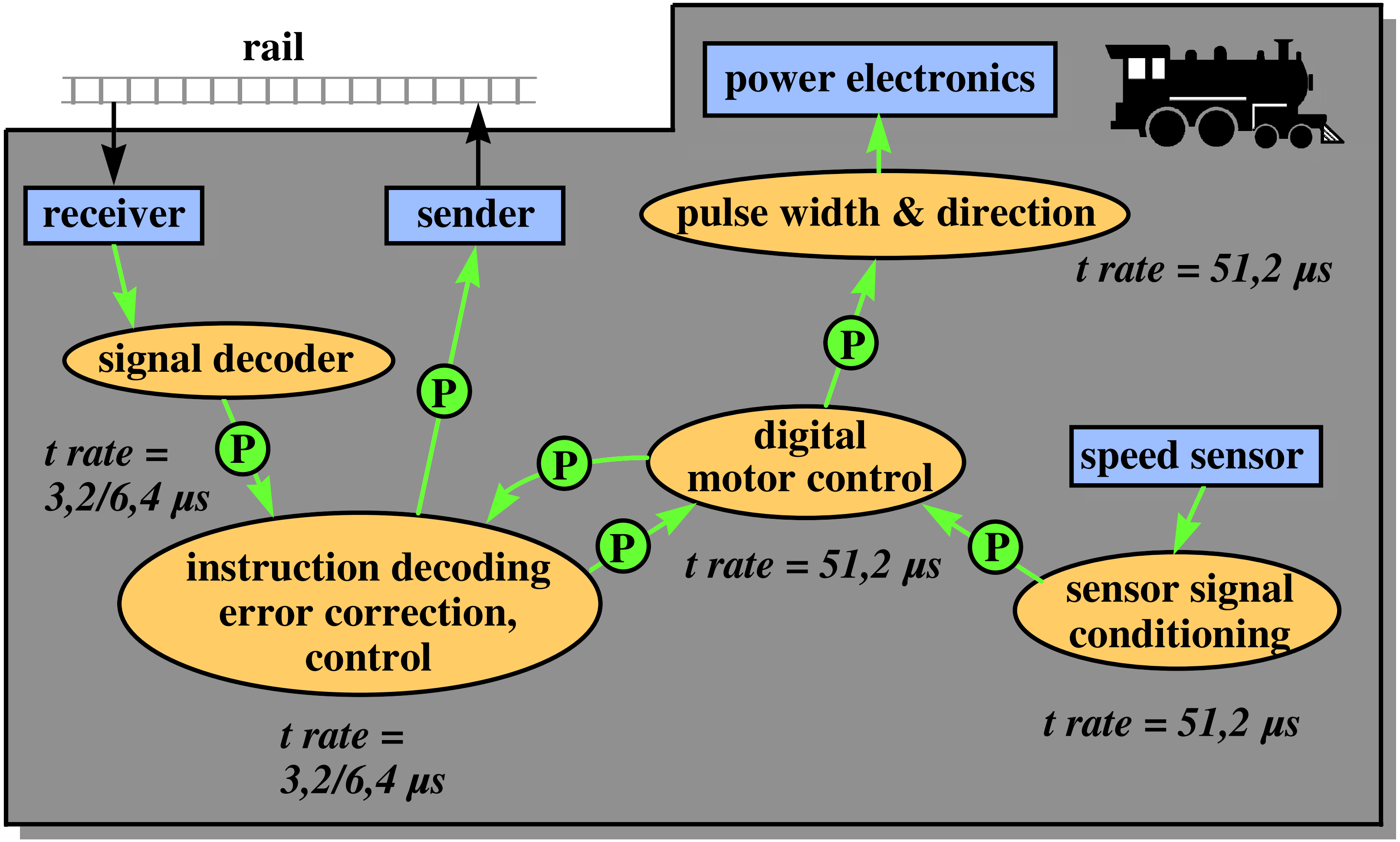
- **high-level transformations**

  examples

  - change algorithm implementation (e.g. change FFT implementation)
  - memory access optimization
  - loop structuring (see experiences in parallel processor design)

$\Rightarrow$ **very high optimization potential**

Co-design using co-synthesis and design space exploration - Example: Model train control [BE 97]



Co-design using co-synthesis and design space exploration - Example: Model train control cont'd

user directives

- **G granularity (in execution time)**
- **P process pipelining**

specification parameter change (product performance)

- **train A: rail token rate 3,2us (max. 256 trains on track)**
- **train B: rail token rate 6,4us (max. 128 trains on track)**

	P	G	8051	Sparc	Gates	t(s)
Train A	n	max		1	26200	28
Train A	y	max		1	27212	36
Train A	y	9us		1	22279	392
Train A	n	max	1		43629	30
Train A	y	max	1		43629	34
Train A	y	9us	1		38677	664
Train B	y	max		1	4623	40
Train B	n	9us		1	7128	88
Train B	y	9us	1		40876	11
Train B	n	9us	1		40101	519

Results (system COSYMA)

Hardware/software co-synthesis and high-level synthesis - where are the differences?

- co-synthesis design space
- application system properties
- flexibility requirements

Co-synthesis design space

- predefined specialized structures for SW execution
  - processors, communication, memory
  - constrained control and combinations of operations (ISA)
  - constrained memory: register sets (spilling), DSP special registers
  - constrained interconnect

  ⇒ SW code generation of signal processing architectures is highly constrained synthesis problem
    - conventional compiler approach:
      dynamic programming on tree graphs (s. Aho, Ullman)
    - DSP:
      deep local search (Goossens), CLP (Marwedel), rule based with optimization across compiler phases (Paulin, Liem)

    ⇒ code optimization for specialized architectures is still open issue

Co-synthesis design space cont'd

- **dynamic effects**
  - **caches**
  - **out-of-order execution/completion**
  - **speculation**
  - **scheduling with preemption (interrupt)**

Application system properties

- combination of different subtasks on one target system

  ⇒ software is used to manage application system complexity

- data dependent control flow and communication subtask execution times
  supported by cache, speculation, o.o.o.e

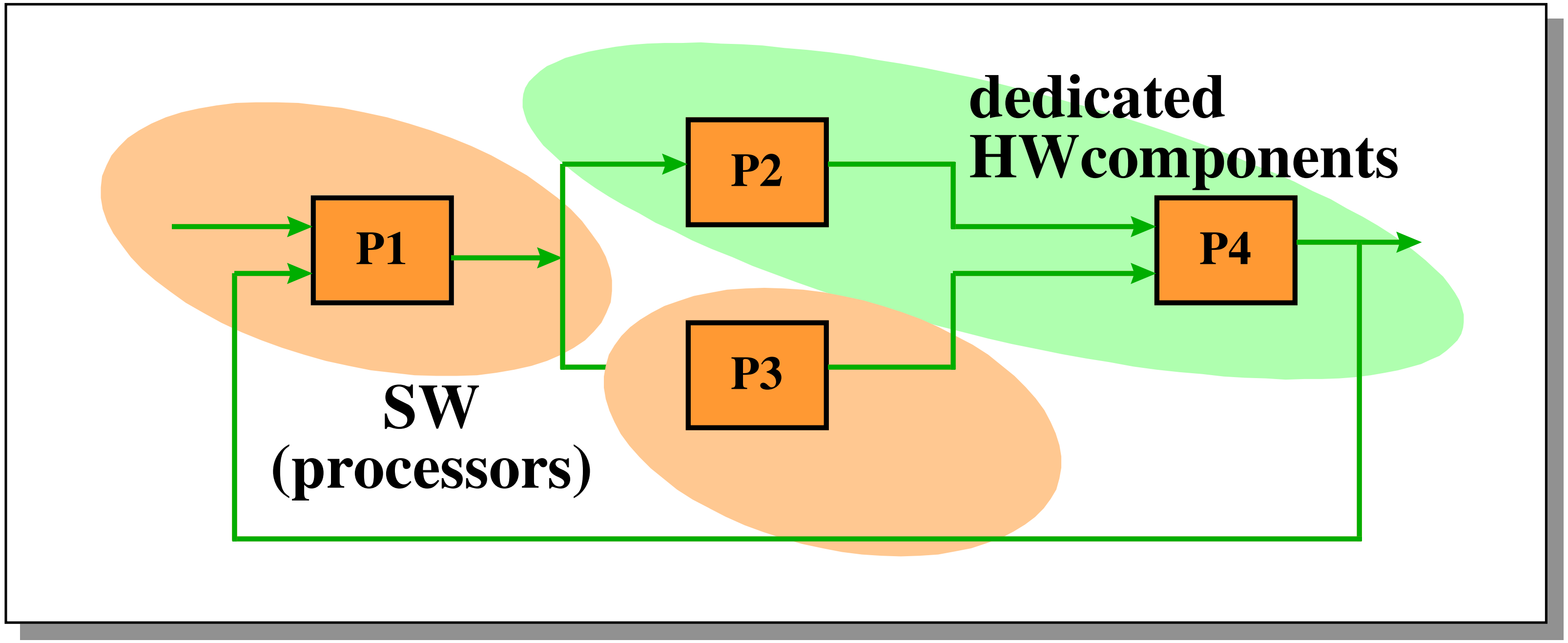
- data dependent execution times and event driven system functions (mixed systems, e.g. mobile communication):
  supported by preemption (possibly OS)

  ⇒ resource sharing problem with larger design space than high-level synthesis

HW/SW partitioning: Combining the uneven couple

- hardware/software partitioning:
  Partitioning of system function to programmable components (software) and hard-wired or parametrized components (hardware)

- similarity to allocation (or load distribution) problem in high-level synthesis (or real-time operating systems)



HW/SW partitioning: Combining the uneven couple cont'd

hardware	⇔	software
few constraints in synthesis	⇔	highly constrained synthesis
user controlled dynamic effects	⇔	many and transparent dynamic effects
high-level or RT synthesis tools	⇔	compilers

⇒ **communication problem (global control and data flow optimization)**

⇒ **tool coupling problem**

HW/SW partitioning cont'd

- **intermediate step: VLIW-ASIPs with specialized function units**
  - **predefined template with minimum constraints on communication and instruction set**

  ⇒ allows use of
    - **Hennessy/Patterson type quantitative architecture optimization**
    - **enhanced standard compiler techniques**

  examples:
  CASTLE co-synthesis system,
  TriMedia processor

HW/SW partitioning problems: A closer look

- partitioning algorithms
- available design space - optimization
- available design space - target architecture

Partitioning algorithms

- represents most of the research work in partitioning
- numerous approaches, much similarity with high-level synthesis
  - first approaches: stochastic optimization (SA), local search
  - now: list scheduling, ILP, stochastic with pareto optimization (GA), Min Cut, rule based, fuzzy logic, tabu search, ...
- approaches differ in level of partitioning granularity and target architecture
- partitioning often combined with scheduling

HLS	RTOS	co-synthesis
allocation +	load distribution	partitioning +
scheduling	+ scheduling	scheduling

⇒ phase coupling problem as in high-level synthesis or compilation

Available design space - Optimization

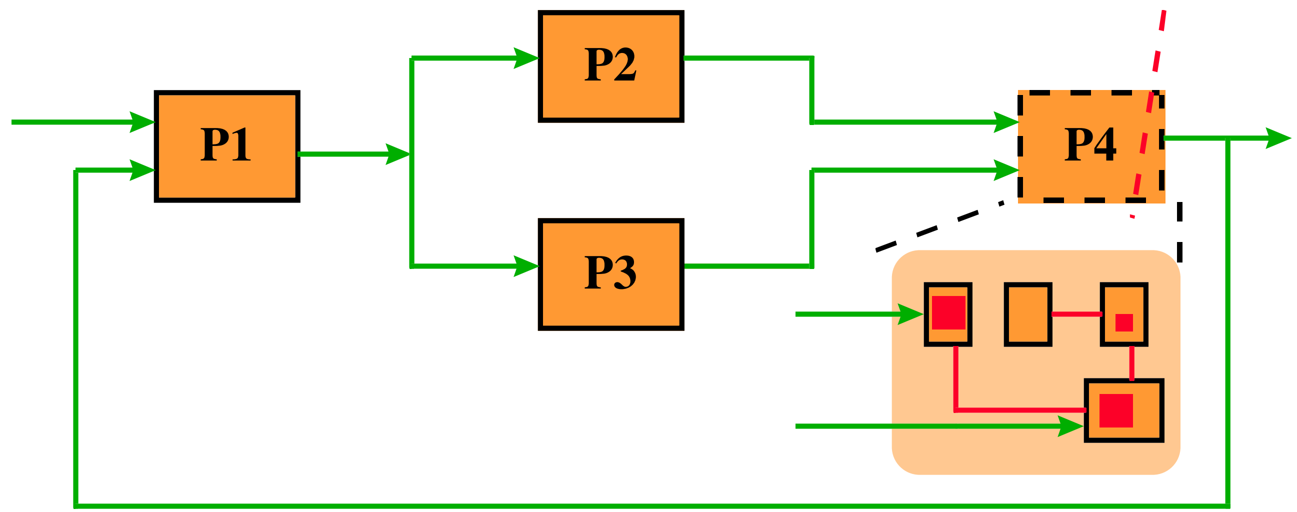
HW/SW system design mostly starts with coarse grain description to manage input system complexity (C, C++, processes)

granularity	comm. analysis	optimization potential	design effort
input process	no (explicit comm.)	↓	↓
function	global data flow analysis		
basic block	global and local DFA		
statement (ASIP)	global und local DFA		

↓ comm. cost

⇒ **granularity adaptation useful** (Henkel/Ernst 97)

Available design space - Optimization cont'd



Granularity and data flow

- process communication is typically explicit
  - ⇒ no data flow analysis for process communication synthesis
- access to local data is implicit
  - ⇒ data flow analysis and data partitioning required for smaller granularity

Available design space - target architecture

- current approaches cover basic communication mechanisms
  more complex mechanisms required: split transaction, burst, ...

- communication minimization
  e.g. using shared data object allocation and access for minimized communication
  ⇔ hardly exploited in co-synthesis

- memory architecture and mapping
  - control dynamic effects
    - improve cache predictability (see RTOS)
    - use of scratch pad RAM with static mapping (Nicolau/Dutt)
  - data memory optimization
    - most work targets (static) data flow
    - little work on mixed systems (e.g. Matisse)

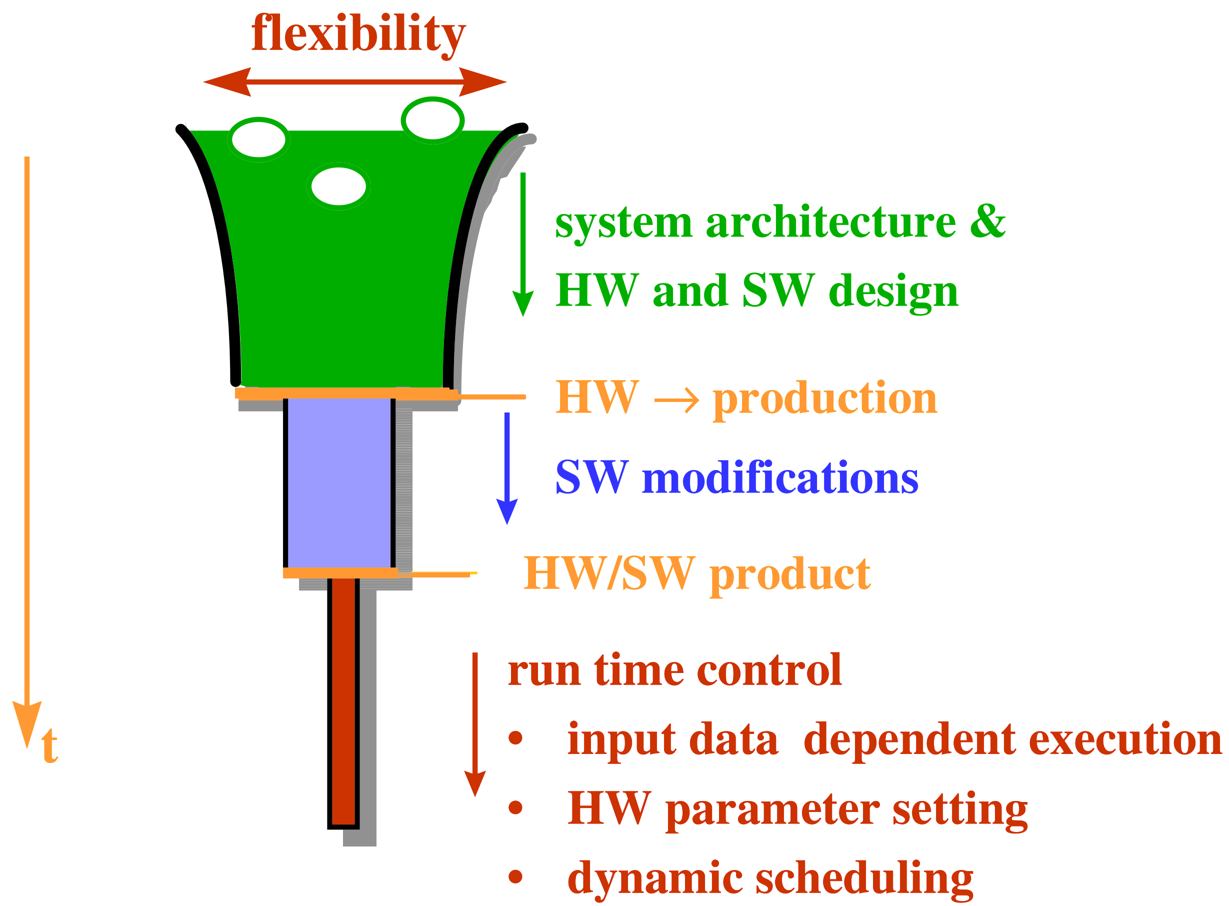
Hardware/software scheduling

- **non preemptive scheduling**
  - **preferred for small granularity and data dominated applications**
    → low context switch overhead
  - **typically includes communication scheduling**
  - **examples: *VULCAN, COSYMA, PTOLEMY, SpecSyn, LYCOS, CHINOOK, ...***
- **preemptive scheduling and dynamic scheduling**
  - **preemptive: Using interrupts for context switch**
    dynamic scheduling: Changing process priorities at run time
  - **preferred in systems with reactive components**
  - **examples: *Wolf et al., TOSCA, ...***
- **mixed approaches**
  - **clustering of processes to static sequences of tasks**
  - **example: *Goossens et al.***

Interface and communication synthesis

- supported mechanisms
  - RPC (COSMOS, Matisse)
  - message exchange (COSYMA, TOSCA, ...)
  - client-server (CoWare)
- point-to-point communication and busses
- preemptive and non preemptive communication scheduling
- limited optimization for block transfer and pipelining

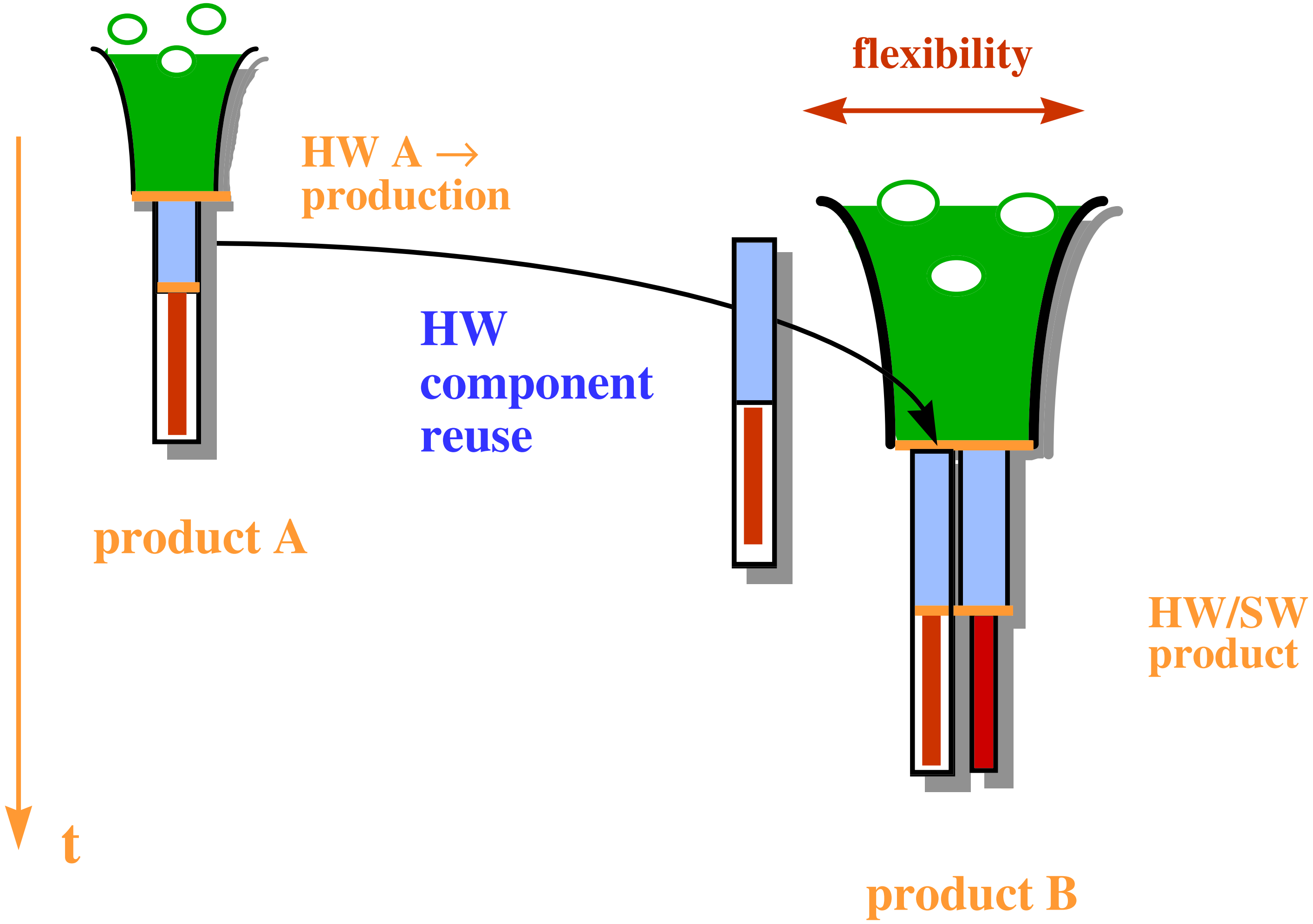
4 Flexibility in the design process - a fundamental problem



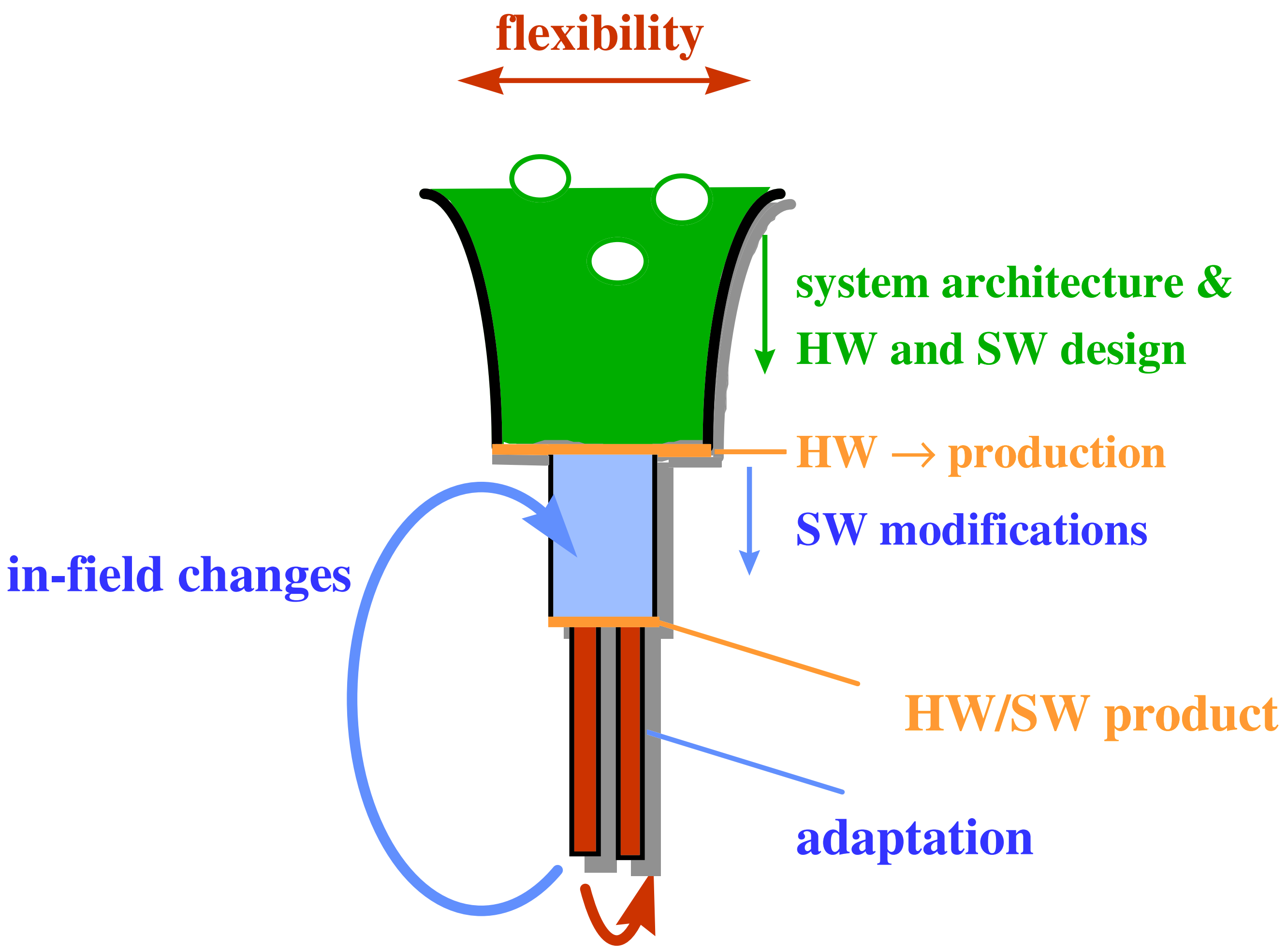
Benefits of flexibility

- **in the design process**
  - to reduce design risk
- **after hardware production**
  - to adapt system to different applications
  - to improve reusability
- **after product delivery**
  - to adapt system to different applications $\Rightarrow$ field programmability
  - to allow function updates in the field
- **at run time**
  - to adapt to different run time situations

Flexibility and reuse

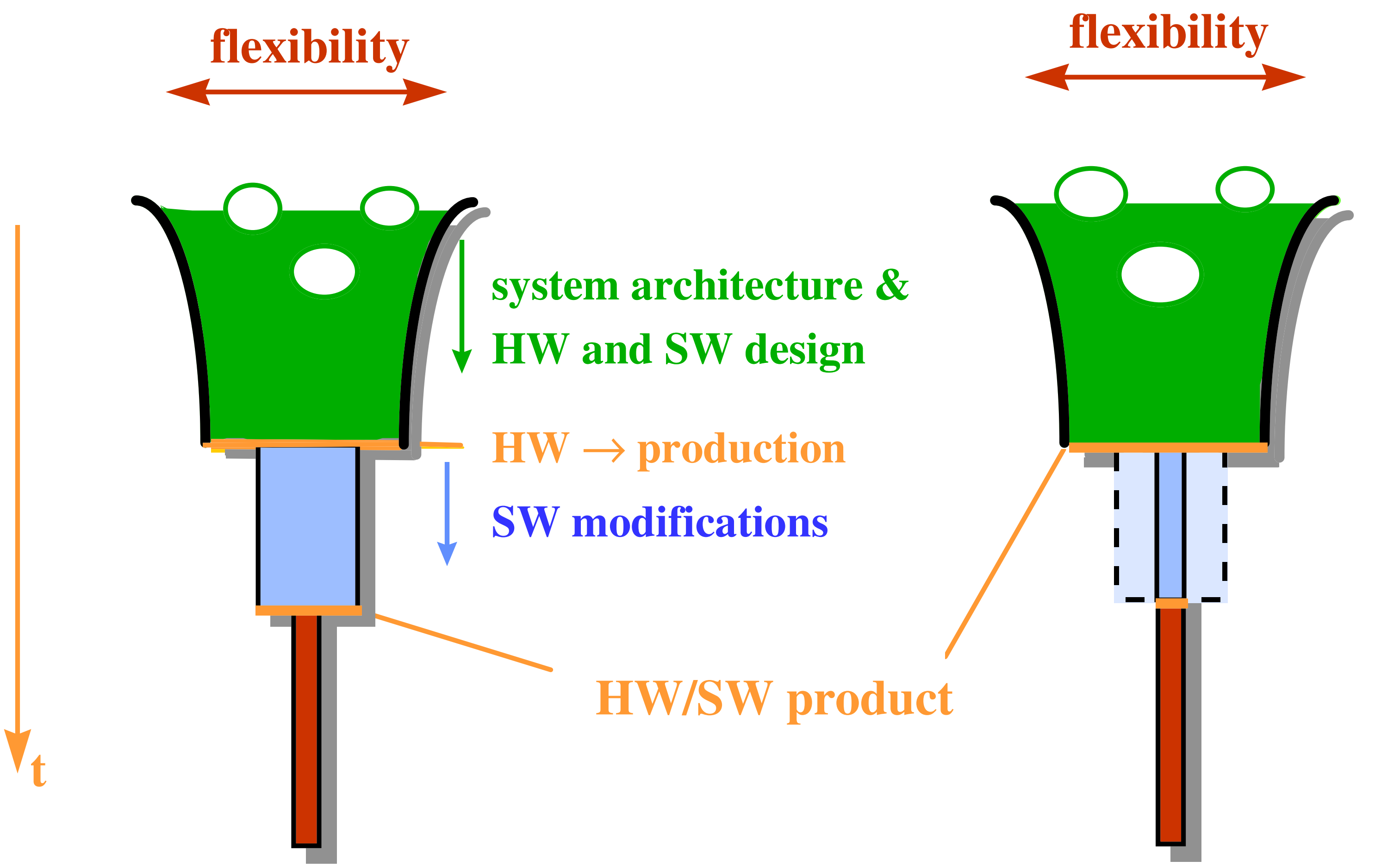


Adaptation and in-field changes



28.10.97

Co-design - high-level product specification

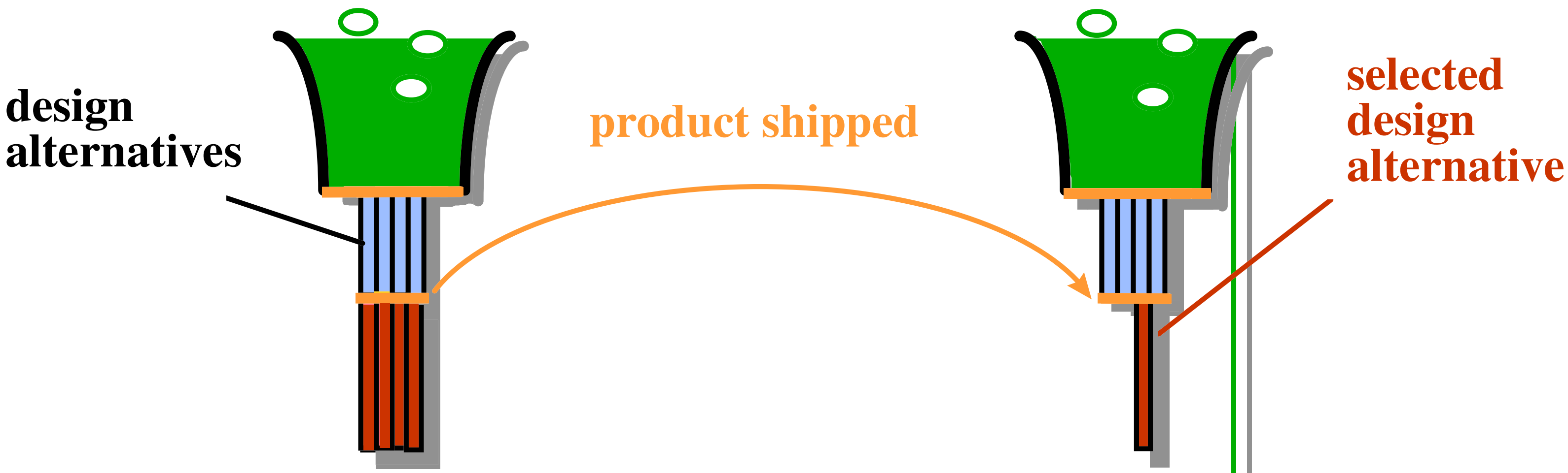


conventional design process — current co-design process

Improving flexibility in co-design

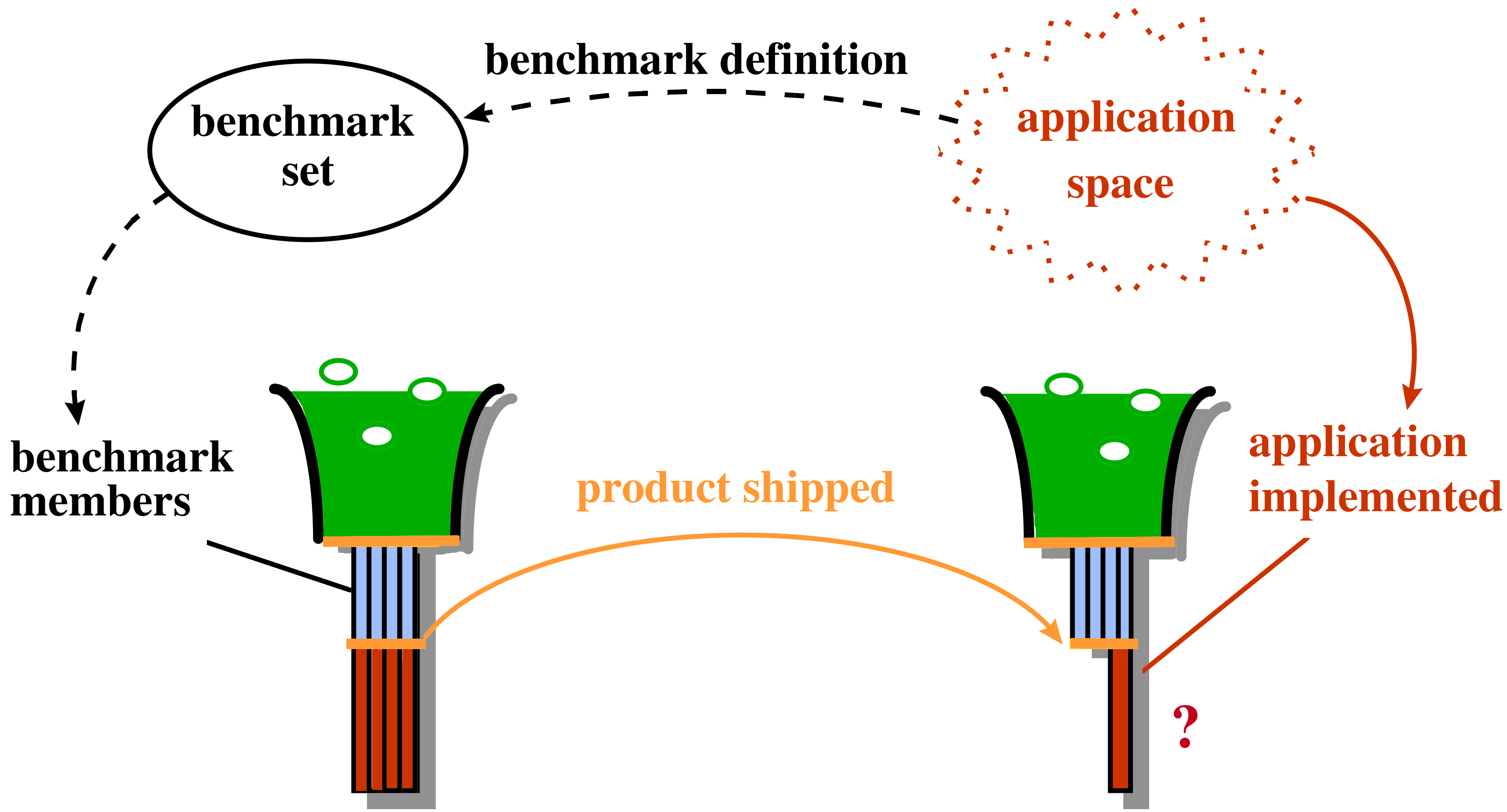
- flexibility in co-design is reduced to the specified system independent of manual or automated design approach
- possible solutions
  - Independent component “design for reuse” (library approach)
  - Design for all required applications

    ⇒ extension of high-level synthesis approach to run time adaptability (Wolf, Potkonjak), but with SW adaptation (Herrmann et al. ICCD 96)



Improving flexibility in co-design cont'd

- **possible solutions cont'd**
  - **Optimization for benchmarks: see Hennessy/Patterson**



Improving flexibility in co-design cont'd

- **inherent specification problems**
  - **Independent component "design for reuse" (library approach)**

    ⇔ **design process integration, flexible component system specification and optimization are unsolved problems**
  - **Design for all required applications**

    ⇔ **only suitable for subset of flexibility requirements**

    ⇔ **subsets of alternatives create exponential design space**
  - **Optimization for benchmarks: see Hennessy/Patterson**

    ⇔ **sensitive to benchmark quality**

    ⇔ **design risk increases with degree of specialization**

    ⇒ **suitable for certain architecture templates (e.g. VLIW)**

⇒ **specification with flexibility definition required**

5 Conclusions

- computer aided hardware/software co-design is a new and rapidly growing area
- the current focus in industry is on co-simulation and co-specification to increase productivity in a mostly manual highly concurrent design process
- hardware/software co-synthesis will support a highly automated abstract co-design process with design space exploration and function reuse across architectures
- co-synthesis requires progress in many areas
  - CAD and algorithms
    - ⇒ better hardware and software synthesis tools
  - design space exploration and function migration support
    - ⇒ high level transformations as user support
  - systematic approach to flexibility
    - ⇒ flexibility specification and better understanding of user design decisions

Literature and related events (selected)

This presentation

- Th. Benner, R. Ernst. *An Approach to Mixed System Co-Synthesis.* Proc. IEEE/ACM/IFIP Workshop on Hardware/Software Co-Design, Braunschweig, 97.
- R. Ernst, Th. Benner, *Communication, Constraints and User Directives in COSYMA*, Technical Report CY-94-2, Institut für DV-Anlagen, Technische Universität Braunschweig, June 1994.

Overview articles and books:

- W. Wolf, J. Staunstrup (eds.). Hardware/Software Co-Design - Principles and Practice. Kluwer Academics, 97. (extended course material)
- G. DeMicheli, R. Gupta, Hardware/Software Co-Design, IEEE Proceedings, Mar. 97, pp. 349 - 365.
- W. Wolf, Hardware/software co-design of embedded systems. IEEE Proceedings, July 94, pp. 965 - 989.
- R. Ernst. Hardware/software co-design of embedded systems. APCHDL 97.
- D. Gajski et al.. *Specification and Design of Embedded Systems*. Kluwer, 95.

Studies:

OMIMO/MCC Study on HW/SW Co-Design. For availability contact J.M. Laporte at OMIMO, JMLaporte@omimo.be

Special issues:

IEEE Micro, Aug 94, IEEE Design and Test, Sep. and Dec. 93, IEEE Proceedings, Mar. 97, Mini Micro Systems vol. 20 (96), Journal of Computer and Software Engineering vol. 3 (94).

Journal with codesign focus:

Design Automation for Embedded Systems (Kluwer)

Round table discussion: IEEE Design and Test, Spring 94

Literature and related events cont'd

Events:

- Intl. Workshop on HW/SW Co-design (Codes/CASHE), Intl. Symposisum on System Synthesis (ISSS), Design Automation Conference (DAC), Design and Test in Europe (DATE), Intl. Conf. on Computer Aided Design of VLSI Circuits (ICCAD), Intl. Conf. on Computer Design (ICCD)
- Courses: NATO ASI on Hardware/Software Co-Design 95 (contributions published by Kluwer), Nato ASI on System Level Synthesis, 98 (www.ecsi.org)

Email-lists:

- codesign@ifi.unizh.ch (to subscribe codesign-request@ifi.unizh.ch)
- isss@ics.uci.edu (to subscribe isss-request@ics.uci.edu)

Web-Page: www.ifi.unizh.ch/~pilz/codesign.html