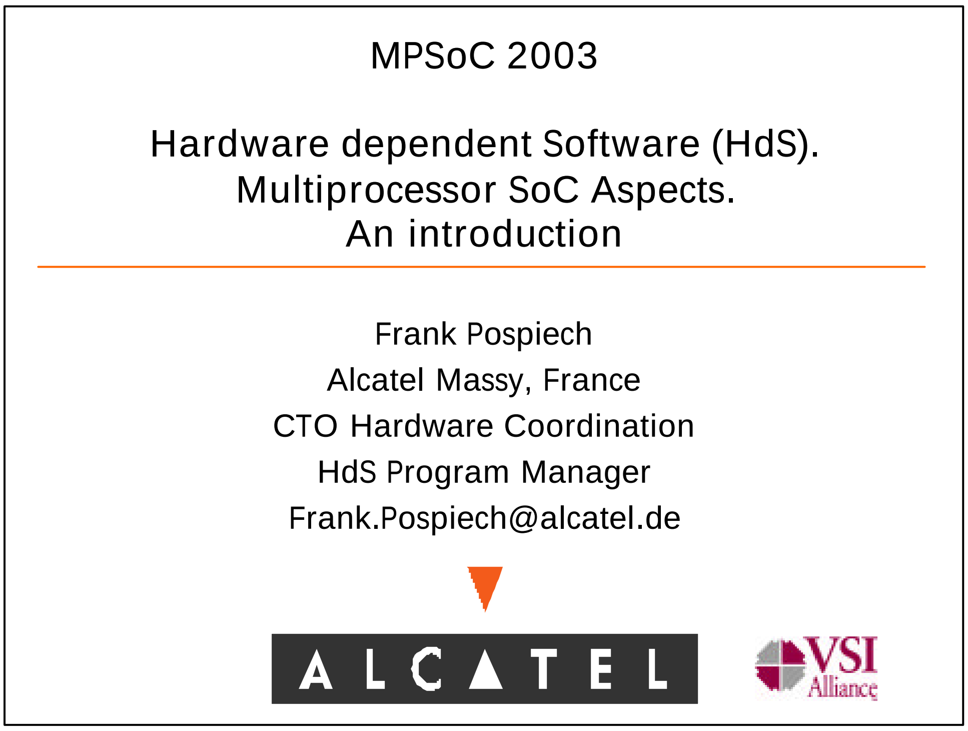


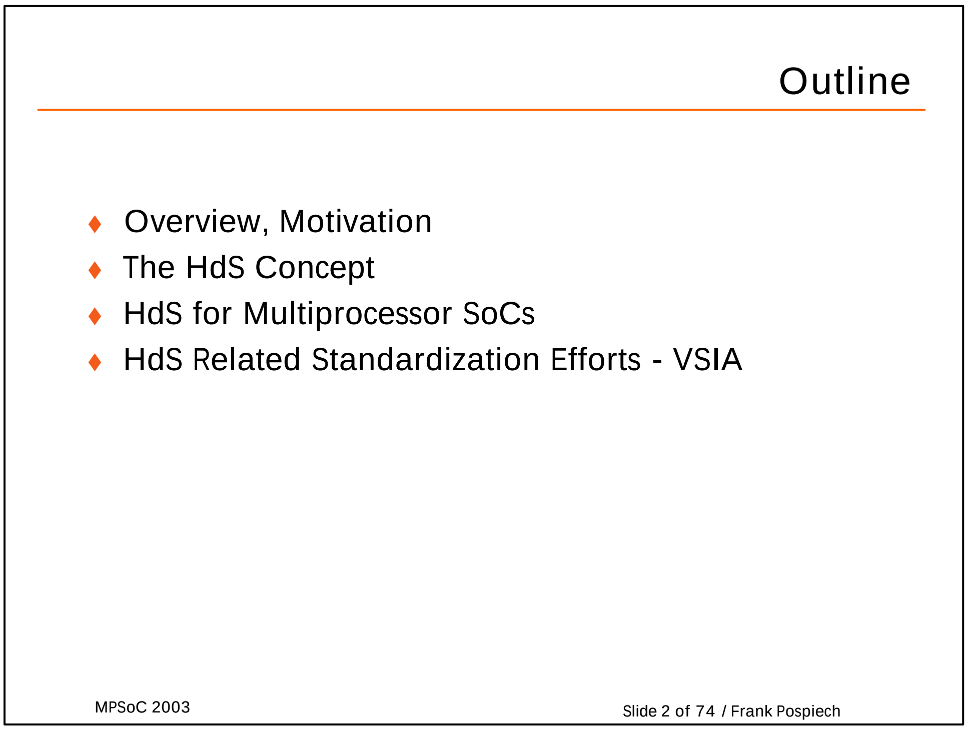
MPSoC 2003

Hardware dependent Software (HdS). Multiprocessor SoC Aspects. An introduction

Frank Pospiech
Alcatel Massy, France
CTO Hardware Coordination
HdS Program Manager
Frank.Pospiech@alcatel.de

ALCATEL

VSI Alliance



Outline

- Overview, Motivation
- The HdS Concept
- HdS for Multiprocessor SoCs
- HdS Related Standardization Efforts - VSIA

Acknowledgements

- Parts of this presentation are adapted from material provided by
  - Ian Philips, ARM
  - Roel Marichal, Alcatel
  - Sebastien Bocq, Alcatel
  - VSIA
- I'd like to thank all of them for their very valuable contributions to this presentation
- Thanks especially to all of my colleagues, who provided very useful comments when reviewing this presentation.

Outline

Overview
SoC Challenge

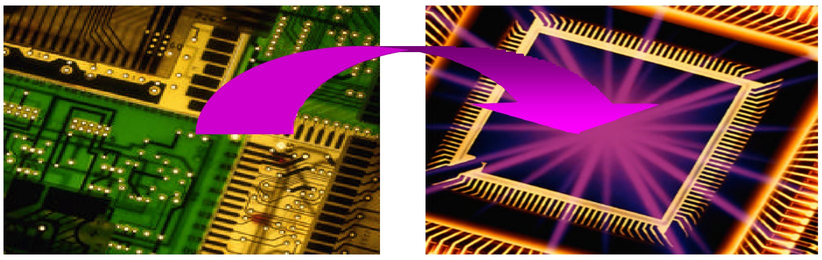
Yesterday’s system is today’s SoC!

Today’s system is tomorrow’s component!

Source:
Bob Altizer, BASYS
VSIA 2002

Overview
Software Challenges for SoC Design

Traditional Design **System-on-Chip Design**

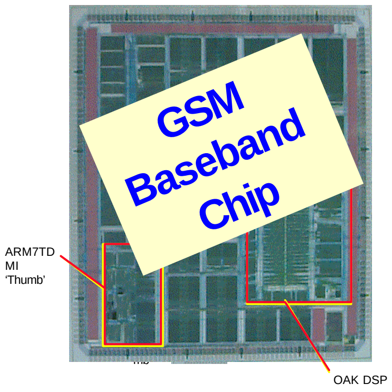


- Increased Design Complexity – Higher Integration
- Ratio of Software to Hardware Engineers Increasing (2-5x)
- Requirement to Port Software Across Various Processors & DSPs
- HW Design is Becoming More SW Focused
- Verification of SoC “Internals” Difficult
- Application Software Needs to be Ported with Each New Generation

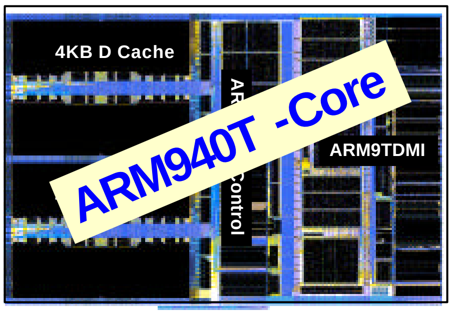
Source:
Mike Kaskowitz, Mentor Graphics
VSIA 2002

Overview. What's the Problem?
SoC complexity. ARM Evidence

1996: ~ 80 mm^2 on 0.5µ

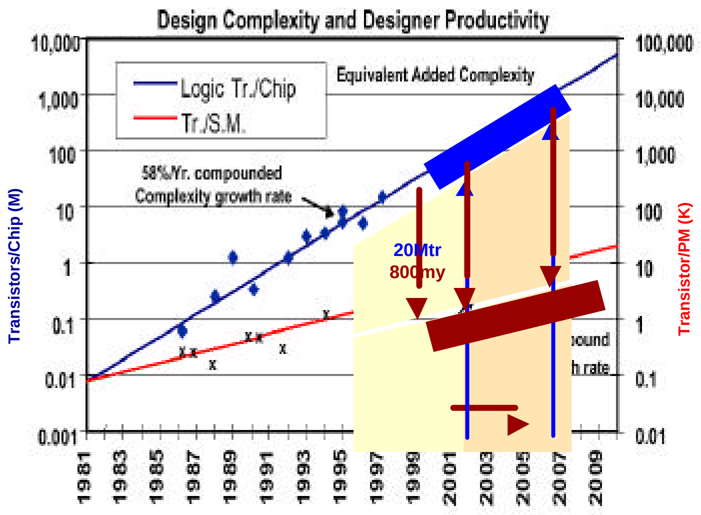


1999: ~ 8 mm^2 on 0.25µ



Source:
Ian Phillips, ARM
VSIA 2001

Overview. What's the Problem?
The Productivity Gap



ITRS'99

Overview. What’s the Problem?
Hardware is not the whole System !!!

- The Productivity-Gap is a real issue …
  - But Hardware alone does not make a System!
    … Hardware alone is no more than a Component.
- A Micro-Electronic System is the result of a projection of …
  - Architecture
  - Hardware
  - Software

  … Distinguished by its gross Functional Behaviour !
- Software is an important part of the Product and must be part of the Design Process

  … or we are only designing a Component of the system.

Overview What’s the Problem?
Embedded System Challenges for HW Folks

- PARADIGM CHANGE!
  - Designers main tasks convert from **processor integration** to **performance analysis**. Concentration on functional requirements instead of integration work
  - Concentration on architectural exploration (including performance analysis

  → **Re-use** and **Platform-based design** become key!

→ Early validation of system/solution correctness
→ Parallel hardware and software development
→ More effective use of previous work
→ Faster ways to build new elements of a solution
→ Ways to test more effectively, efficiently, and quickly

Overview. What's the Problem?
Hardware *is* Software

An example Sub-System …

♦ Reference BlueTooth Implementation :-

• ARM core	…	40K lines of RTL
• 80K gates	…	30K lines of RTL
• 110KB binary	… _____	70K lines of C
	total 140K source lines	

… Including Memory <10% of Core Area at 0.1µ !

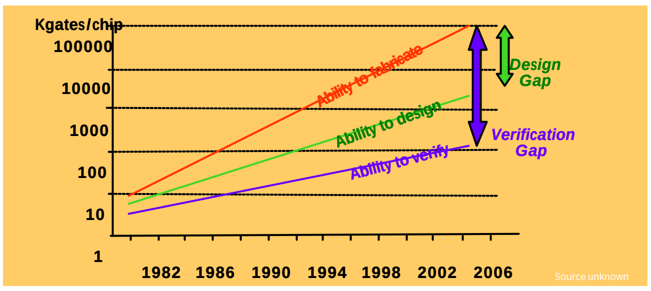
Source:
Ian Phillips, ARM
VSIA 2001

MPSoC 2003

Slide 11 of 74 / Frank Pospiech

Overview. What's the Problem?
Verification & Software Gap

♦ **The Verification Gap**

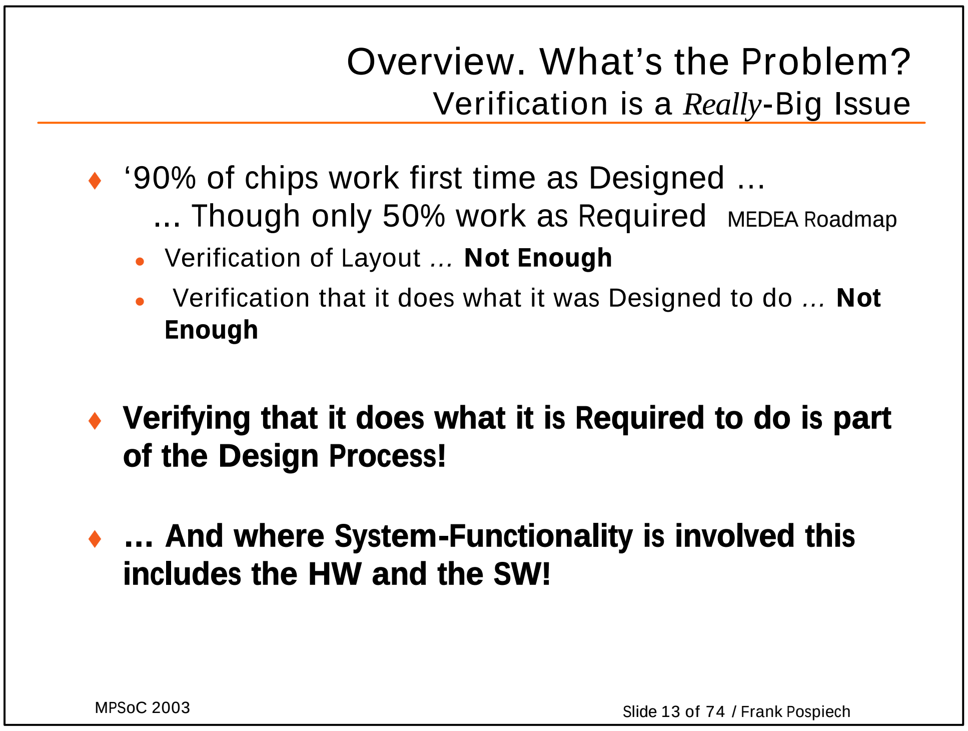


♦ **The Software Gap :**
"The complexity of a System-on-Chip design is not only in the million transistors packed in a square millimetre. The major challenge for technical success of a SoC is to make sure that millions lines of software fit in with millions gates" Source : DAC2002 *

* **Going Mobile: The Next Horizon for Multi-million Gate Designs in the Semi-Conductor Industry,** Christian BERTHET (STMicroelectronics), *DAC2002*

MPSoC 2003

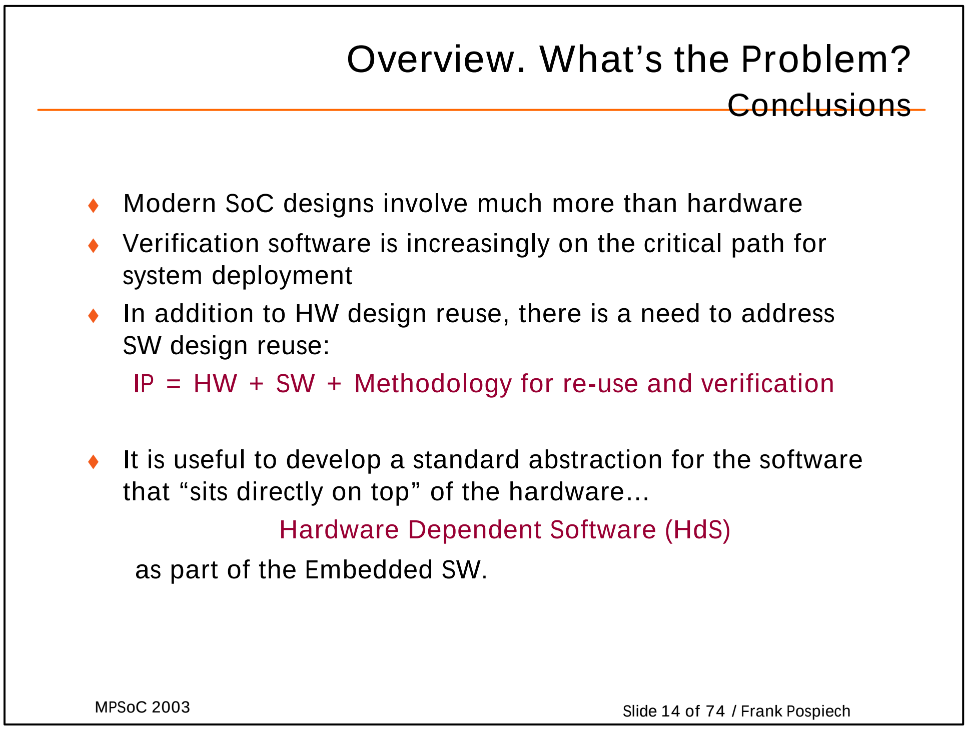
Slide 12 of 74 / Frank Pospiech



Overview. What's the Problem?
Verification is a *Really*-Big Issue

- '90% of chips work first time as Designed …
  … Though only 50% work as Required MEDEA Roadmap
  - Verification of Layout … **Not Enough**
  - Verification that it does what it was Designed to do … **Not Enough**
- **Verifying that it does what it is Required to do is part of the Design Process!**
- **… And where System-Functionality is involved this includes the HW and the SW!**

MPSoC 2003 Slide 13 of 74 / Frank Pospiech



Overview. What's the Problem?
Conclusions

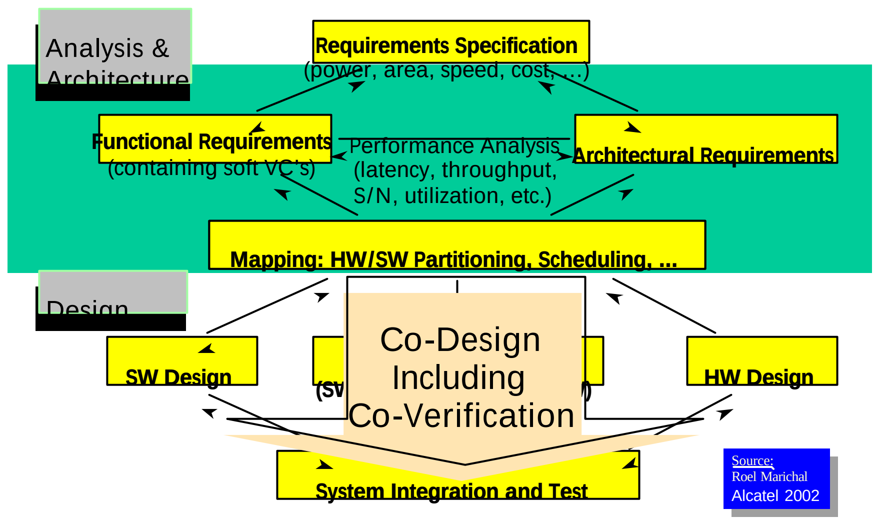
- Modern SoC designs involve much more than hardware
- Verification software is increasingly on the critical path for system deployment
- In addition to HW design reuse, there is a need to address SW design reuse:
  IP = HW + SW + Methodology for re-use and verification
- It is useful to develop a standard abstraction for the software that "sits directly on top" of the hardware…
  Hardware Dependent Software (HdS)
  as part of the Embedded SW.

MPSoC 2003 Slide 14 of 74 / Frank Pospiech

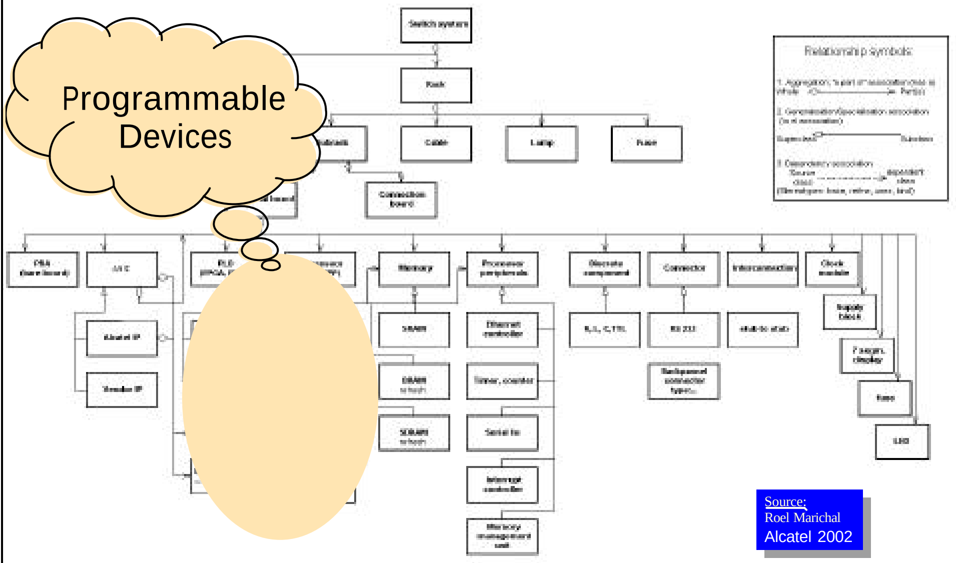
Outline

- Overview, Motivation
- **The HdS Concept**
  - **A HW-SW Co-Development Process**
  - HdS
  - Isn’t HdS Just Software?
  - HdS-API
- HdS for Multiprocessor SoCs
- HdS Related Standardization Efforts - VSIA

Development process investigations: Co-design of HW and SW

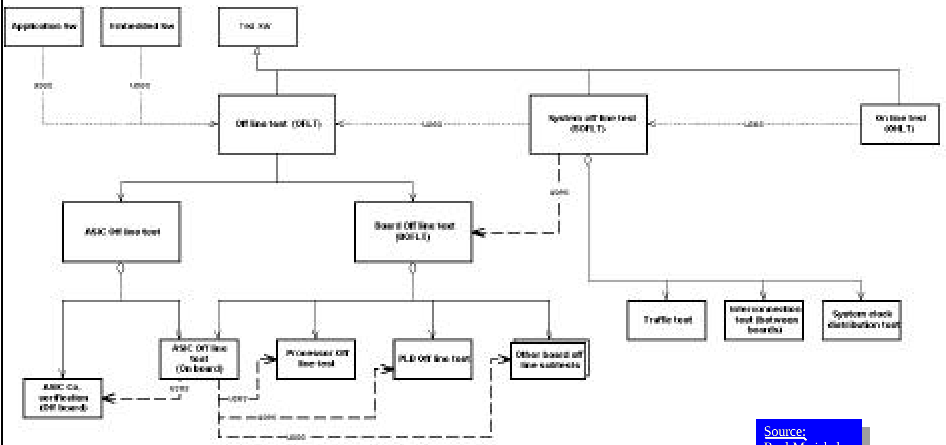

Analysis & Architecture
Requirements Specification
(power, area, speed, cost, ...)
Functional Requirements
(containing soft VC's)
Performance Analysis
(latency, throughput, S/N, utilization, etc.)
Architectural Requirements
Mapping: HW/SW Partitioning, Scheduling, ...
Design
SW Design
(SV
)
HW Design
Co-Design Including Co-Verification
System Integration and Test
Source:
Roel Marichal
Alcatel 2002

“Hardware view” of Switch Fabric



Source:
Roel Marichal
Alcatel 2002

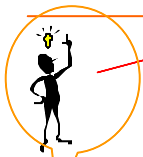
A HW-SW Co-Development Process
SW organisation w.r.t Verification and Test



Application sw
Embedded sw
Test SW
uses
uses
Off line test (OFLT)
uses
System off line test (SOFLT)
uses
On line test (ONLT)
uses
ASIC Off line test
Board Off line test (BOFLT)
ASIC Off line test (On board)
Processor Off line test
PLD Off line test
Other board off line subtests
ASIC Co-verification (Off board)
uses
uses
uses
uses
Traffic test
Interconnection test (between boards)
System clock distribution test

Source:
Roel Marichal
Alcatel 2002

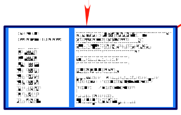
A HW-SW Co-Development Process
A Generic Life-Cycle



System Conception

Technical Requirements

Top Level Design
Detailed Design



- Coding
- Circuit Design
- ASIC design
- Embedded SW design
- Physical design

Design

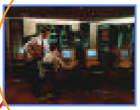
Verification/Validation

Lab Test

- Module Test
- Functional Group Test

Integration Acceptance

Integration Platform



- SW Integration with HW
- System Integration Test

Deployment Field Support

Alcatel product with HW + HdS + SW (e.g. ADSL)

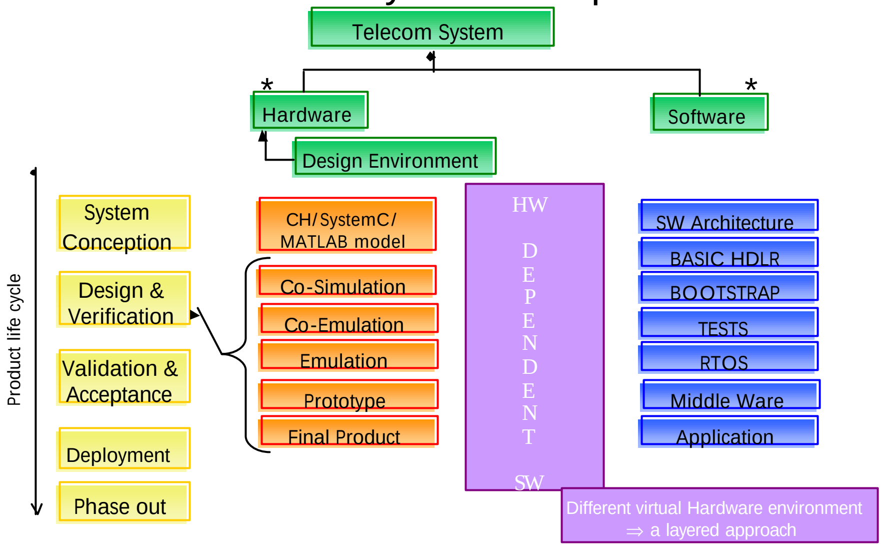
Source:
Roel Marichal
Alcatel 2002

A HW-SW Co-Development Process
Life Cycle and Impact on Test SW

System Conception & Feasibility	Design & Verification	Validation & Acceptance	Deployment & Field Support	Phase Out
SUD = Executable models balancing: •functional req. •Architectural aspects •Non-functional req. **System Tests** checking the models	SUD =Refinement of Models in HW/SW **Module and functional group tests** •ASIC co-Verification •Board Verification •System tests •Off-line & On-line tests •**System Tests** are **REUSED** •Test Supporting framework is **REUSED:** *Bootstrap & Basic Handler*, a general framework that p[illegible]vides a platform for Tests, Basic actions,etc.	Validate & Integration of the SUD **Validation & Acceptance Tests** Subsets of **tests of previous** phases are **REUSED**	**New Tests addressing aspects not thought off** All **tests of previous phases** are **REUSED** to “tro[illegible]hoot”	

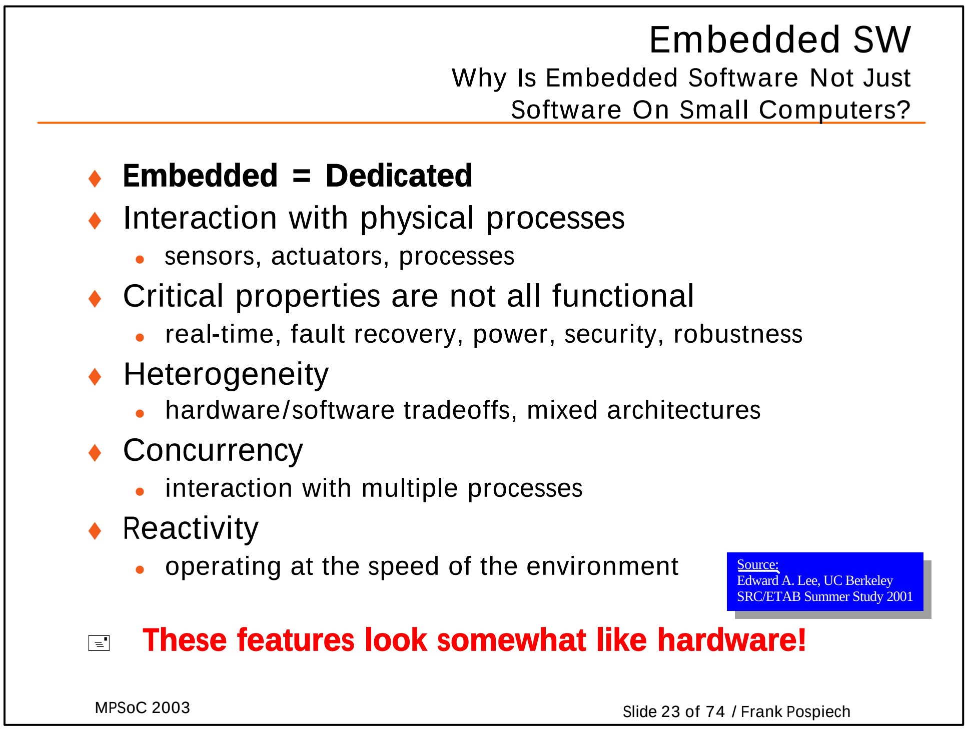
Source:
Roel Marichal
Alcatel 2002

A HW-SW Co-Development Process
Life Cycle and Impact on HW & SW



Outline

- Overview, Motivation
- **The HdS Concept**
  - A HW-SW Co-Development Process
  - **HdS**
  - Isn’t HdS Just Software?
  - HdS-API
- HdS for Multiprocessor SoCs
- HdS Related Standardization Efforts - VSIA



Embedded SW

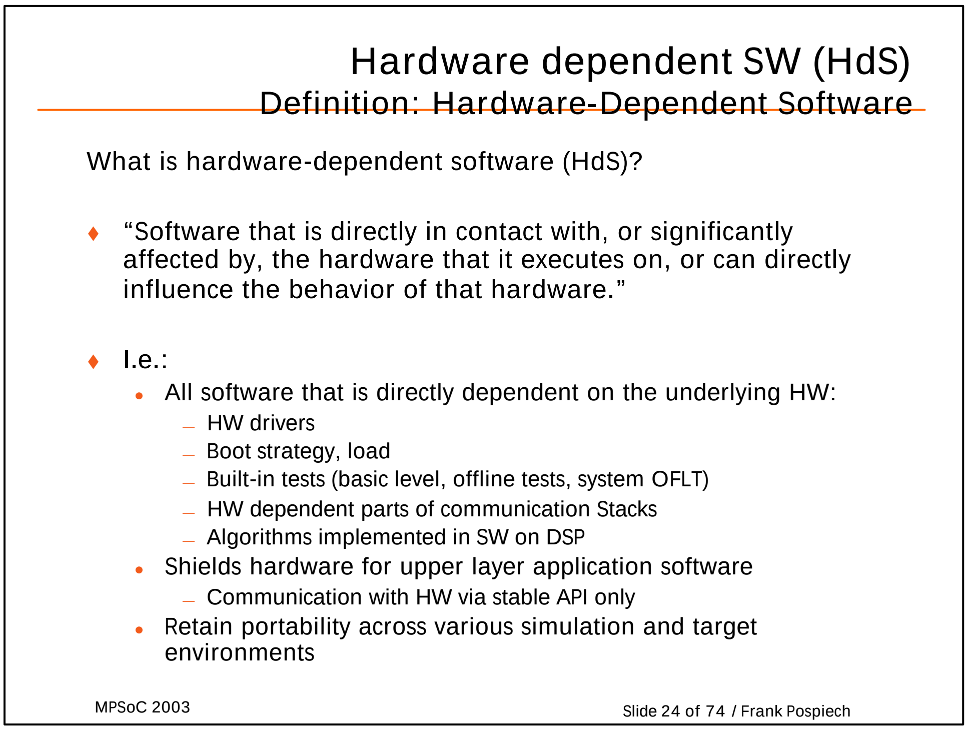
Why Is Embedded Software Not Just Software On Small Computers?

- **Embedded = Dedicated**
- Interaction with physical processes
  - sensors, actuators, processes
- Critical properties are not all functional
  - real-time, fault recovery, power, security, robustness
- Heterogeneity
  - hardware/software tradeoffs, mixed architectures
- Concurrency
  - interaction with multiple processes
- Reactivity
  - operating at the speed of the environment

Source:
Edward A. Lee, UC Berkeley
SRC/ETAB Summer Study 2001

These features look somewhat like hardware!

MPSoC 2003 Slide 23 of 74 / Frank Pospiech



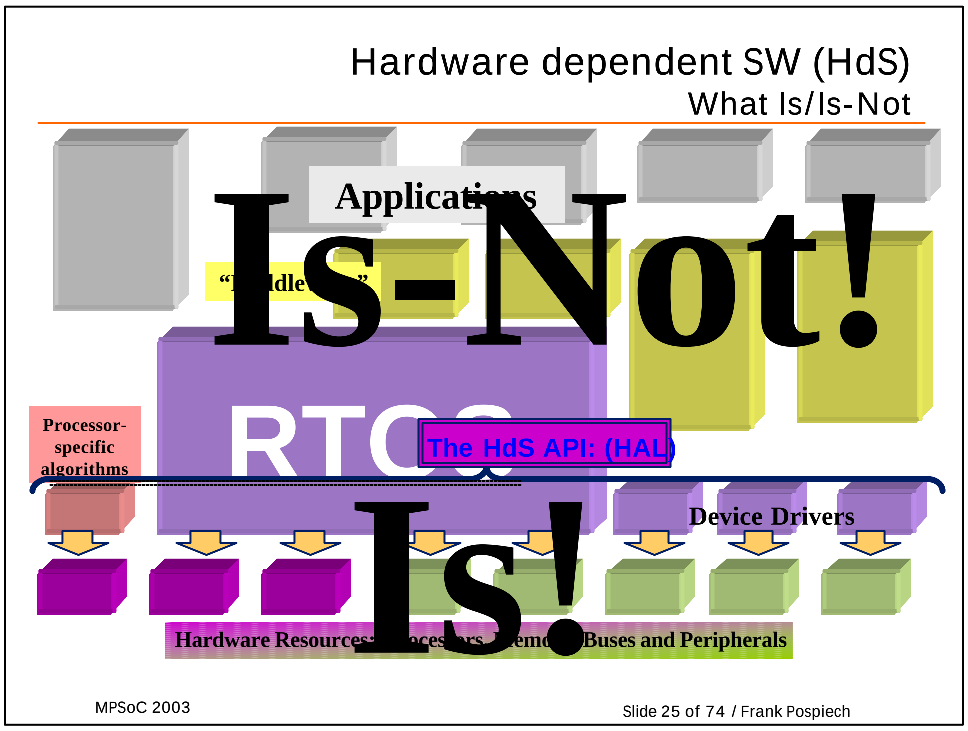
Hardware dependent SW (HdS)

Definition: Hardware-Dependent Software

What is hardware-dependent software (HdS)?

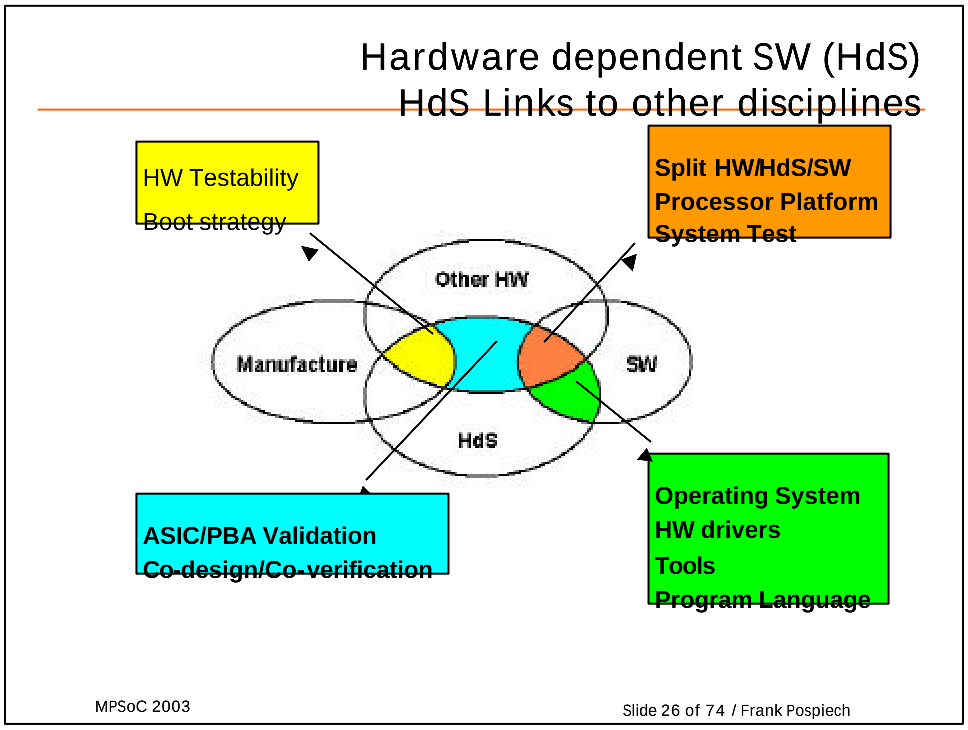
- “Software that is directly in contact with, or significantly affected by, the hardware that it executes on, or can directly influence the behavior of that hardware.”
- I.e.:
  - All software that is directly dependent on the underlying HW:
    - HW drivers
    - Boot strategy, load
    - Built-in tests (basic level, offline tests, system OFLT)
    - HW dependent parts of communication Stacks
    - Algorithms implemented in SW on DSP
  - Shields hardware for upper layer application software
    - Communication with HW via stable API only
  - Retain portability across various simulation and target environments

MPSoC 2003 Slide 24 of 74 / Frank Pospiech



Hardware dependent SW (HdS)
What Is/Is-Not

Applications
"Middleware"
Is-Not!
RTOS
Processor-specific algorithms
The HdS API: (HAL)
Device Drivers
Is!
Hardware Resources: Processors, Memory, Buses and Peripherals



Hardware dependent SW (HdS)
HdS Links to other disciplines

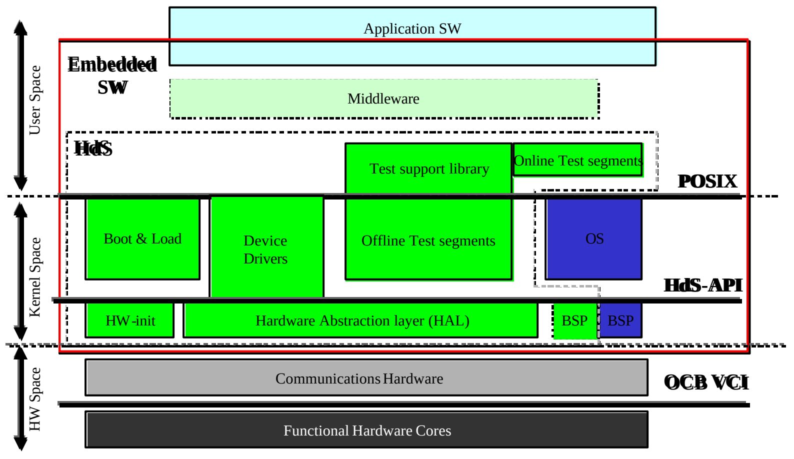
HW Testability
Boot strategy

Split HW/HdS/SW
Processor Platform
System Test

ASIC/PBA Validation
Co-design/Co-verification

Operating System
HW drivers
Tools
Program Language

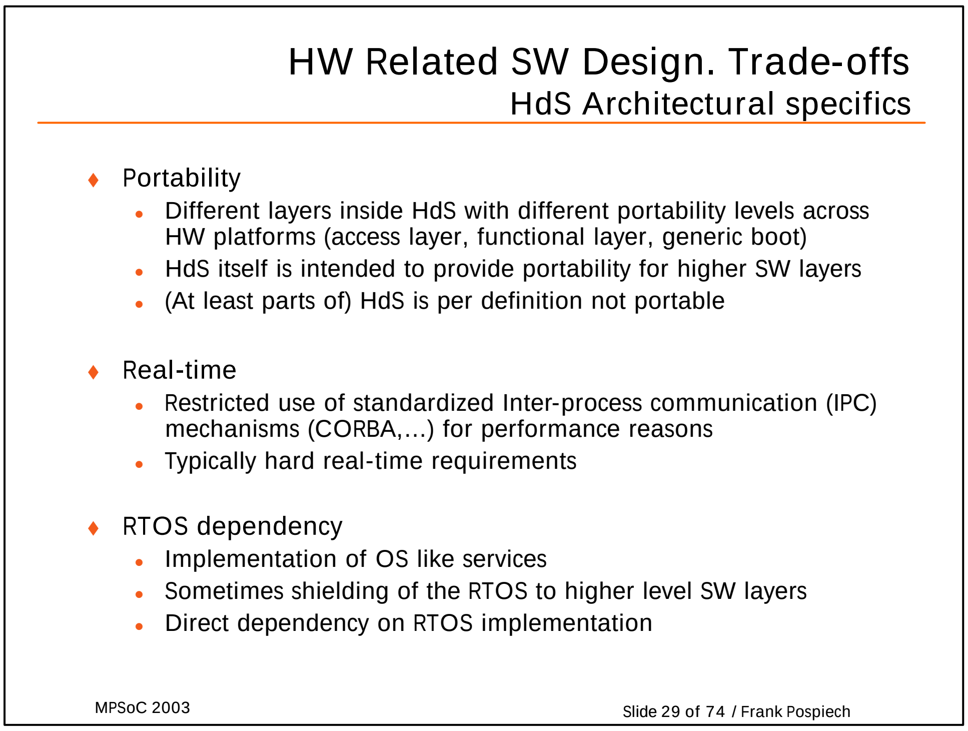
Hardware dependent SW (HdS)
HdS seen as a HW/SW Interface



SoC Design needs a New Standard

Outline

- Overview, Motivation
- **The HdS Concept**
  - A HW-SW Co-Development Process
  - HdS
  - **Isn’t HdS Just Software?**
  - HdS-API
- HdS for Multiprocessor SoCs
- HdS Related Standardization Efforts - VSIA

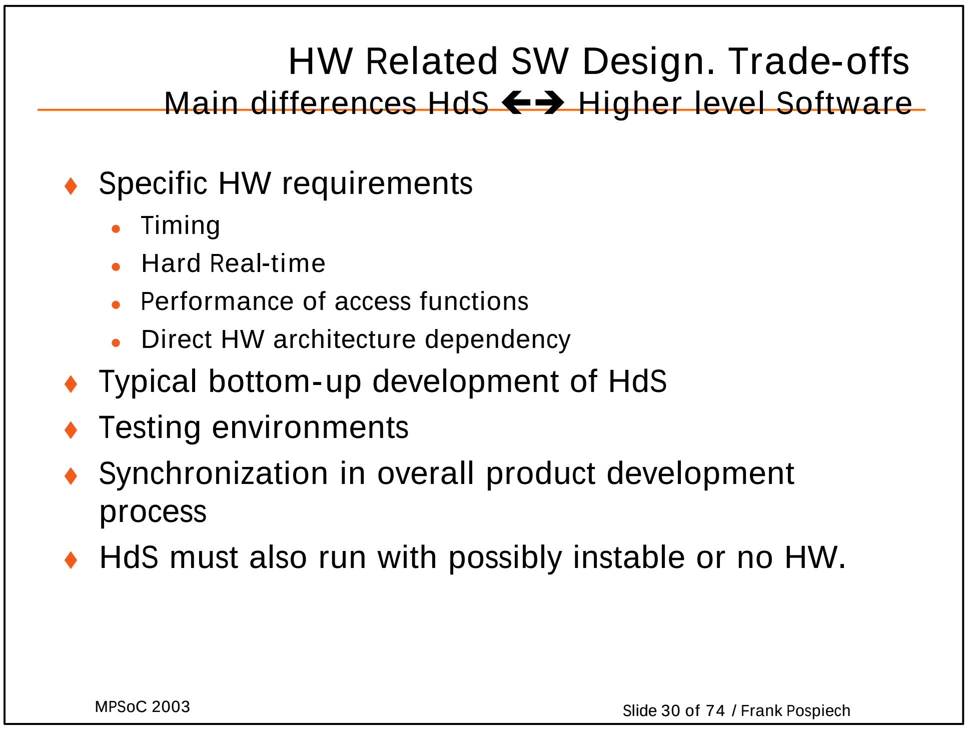


HW Related SW Design. Trade-offs
HdS Architectural specifics

- Portability
  - Different layers inside HdS with different portability levels across HW platforms (access layer, functional layer, generic boot)
  - HdS itself is intended to provide portability for higher SW layers
  - (At least parts of) HdS is per definition not portable
- Real-time
  - Restricted use of standardized Inter-process communication (IPC) mechanisms (CORBA,…) for performance reasons
  - Typically hard real-time requirements
- RTOS dependency
  - Implementation of OS like services
  - Sometimes shielding of the RTOS to higher level SW layers
  - Direct dependency on RTOS implementation

MPSoC 2003

Slide 29 of 74 / Frank Pospiech



HW Related SW Design. Trade-offs
Main differences HdS ←→ Higher level Software

- Specific HW requirements
  - Timing
  - Hard Real-time
  - Performance of access functions
  - Direct HW architecture dependency
- Typical bottom-up development of HdS
- Testing environments
- Synchronization in overall product development process
- HdS must also run with possibly instable or no HW.

MPSoC 2003

Slide 30 of 74 / Frank Pospiech

HW Related SW Design. Trade-offs
Standardization

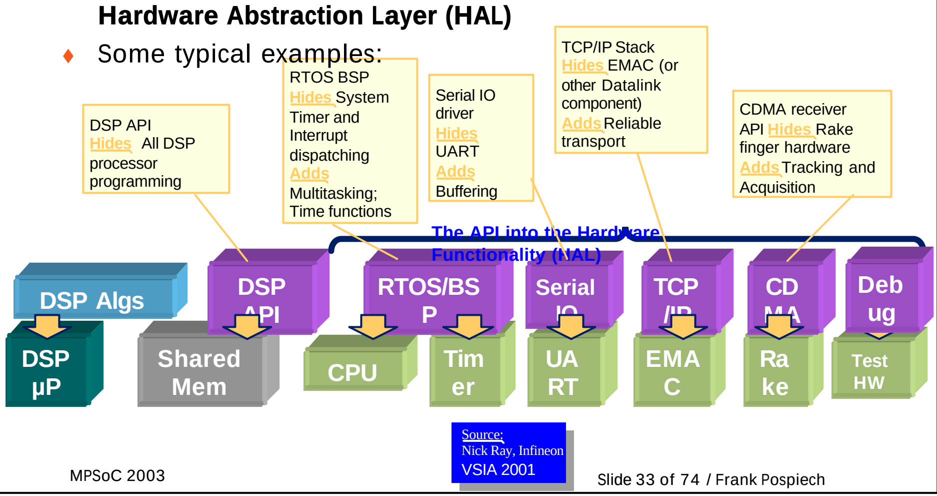
- Many standards are defined for higher level SW
  - Communication (CORBA,…)
  - High level languages, specification languages (ADA, Modula, UML, SDL,…)
  - Protocols (ITU,…)
  - OS Interface (POSIX)
- In the HdS domain, standardization is still quite poor
  - Activities starting (VSIA, see later in this presentation)
  - Project UDI

Outline

- Overview, Motivation
- **The HdS Concept**
  - A HW-SW Co-Development Process
  - HdS
  - Isn't HdS Just Software?
  - **HdS-API**
- HdS for Multiprocessor SoCs
- HdS Related Standardization Efforts - VSIA

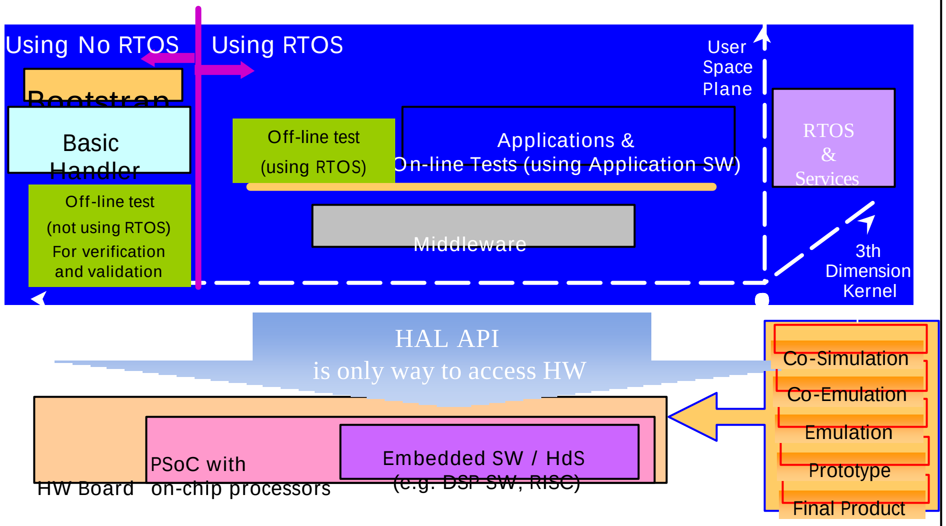
Hardware dependent SW (HdS)
The SoC as an API

- An SoC looks like an API to most embedded SW developers
- SoC functionality should have an abstraction layer, which may also extend the capabilities. This is often called a **Hardware Abstraction Layer (HAL)**
- Some typical examples:

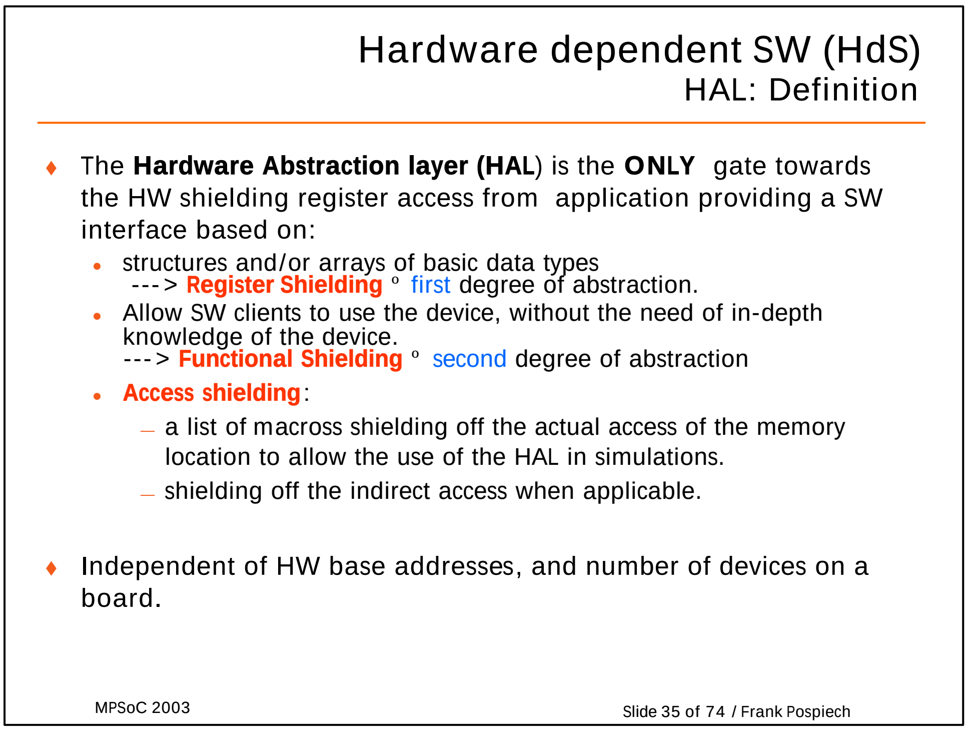

DSP API Hides All DSP processor programming
RTOS BSP Hides System Timer and Interrupt dispatching Adds Multitasking; Time functions
Serial IO driver Hides UART Adds Buffering
TCP/IP Stack Hides EMAC (or other Datalink component) Adds Reliable transport
CDMA receiver API Hides Rake finger hardware Adds Tracking and Acquisition
The API into the Hardware Functionality (HAL)
DSP Algs
DSP API
RTOS/BSP
Serial IO
TCP /IP
CD MA
Debug
DSP µP
Shared Mem
CPU
Timer
UART
EMAC
Rake
Test HW
Source: Nick Ray, Infineon VSIA 2001

MPSoC 2003 Slide 33 of 74 / Frank Pospiech

Why a Hardware Abstraction Layer?
Mapping Tests on Embedded software Framework


Using No RTOS
Using RTOS
Bootstrap
Basic Handler
Off-line test (not using RTOS) For verification and validation
Off-line test (using RTOS)
Applications & On-line Tests (using Application SW)
Middleware
User Space Plane
RTOS & Services
3th Dimension Kernel
HAL API is only way to access HW
HW Board
PSoC with on-chip processors
Embedded SW / HdS (e.g. DSP SW, RISC)
Co-Simulation
Co-Emulation
Emulation
Prototype
Final Product

MPSoC 2003 Slide 34 of 74 / Frank Pospiech

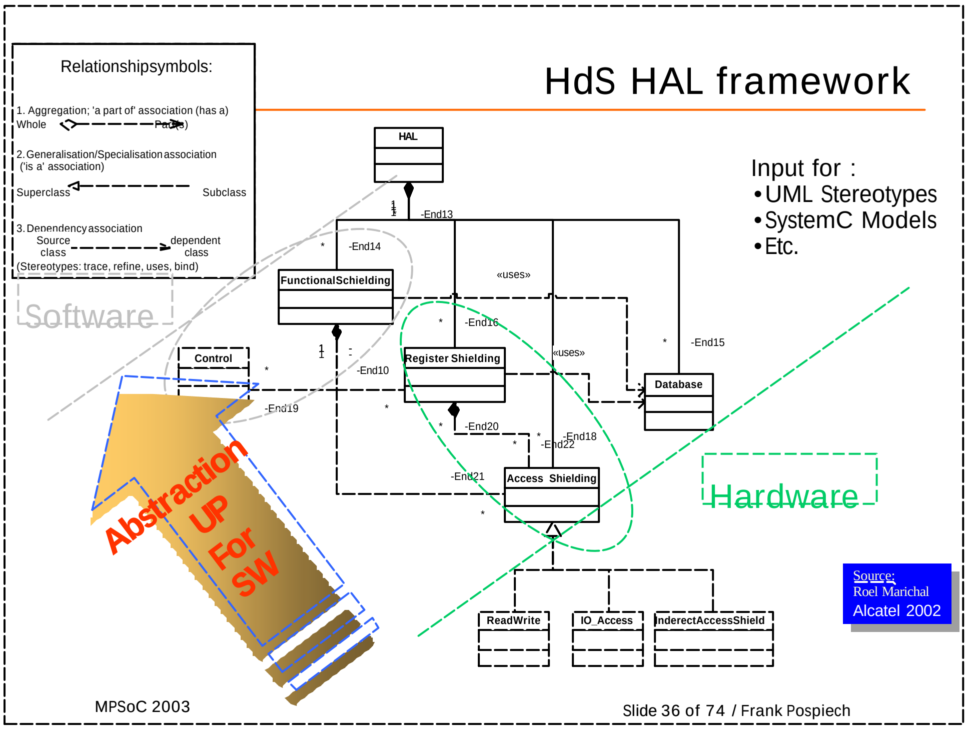


Hardware dependent SW (HdS)
HAL: Definition

- The **Hardware Abstraction layer (HAL)** is the **ONLY** gate towards the HW shielding register access from application providing a SW interface based on:
  - structures and/or arrays of basic data types
    ---> **Register Shielding** º first degree of abstraction.
  - Allow SW clients to use the device, without the need of in-depth knowledge of the device.
    ---> **Functional Shielding** º second degree of abstraction
  - **Access shielding**:
    - a list of macross shielding off the actual access of the memory location to allow the use of the HAL in simulations.
    - shielding off the indirect access when applicable.
- Independent of HW base addresses, and number of devices on a board.

MPSoC 2003

Slide 35 of 74 / Frank Pospiech



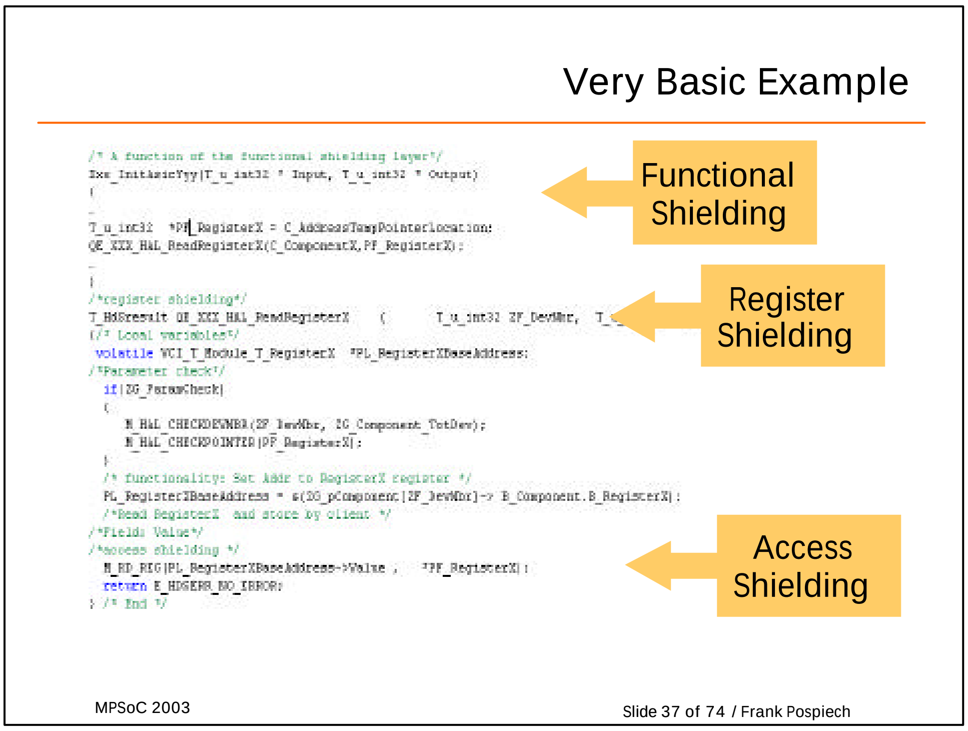
HdS HAL framework

Input for :
- UML Stereotypes
- SystemC Models
- Etc.

Source:
Roel Marichal
Alcatel 2002

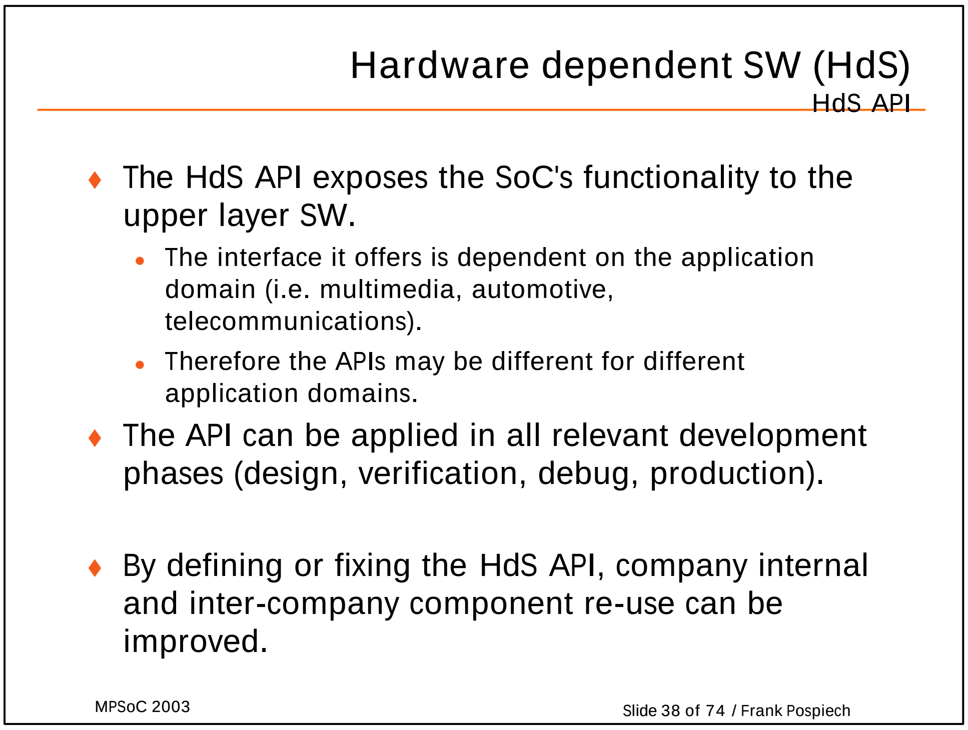
MPSoC 2003

Slide 36 of 74 / Frank Pospiech



Very Basic Example

```
/* A function of the functional shielding layer*/
Sxx_InitAsicYyy(T_u_int32 * Input, T_u_int32 * Output)
{
…
T_u_int32   *PF_RegisterX = C_AddressTempPointerLocation;
QE_XXX_HAL_ReadRegisterX(C_ComponentX,PF_RegisterX);
…
}
/*register shielding*/
T_HdSresult QE_XXX_HAL_ReadRegisterX    (       T_u_int32 ZF_DevNbr,  T_
{/* Local variables*/
 volatile VCI_T_Module_T_RegisterX  *PL_RegisterXBaseAddress;
/*Parameter check*/
  if(ZG_ParamCheck)
  {
     M_HAL_CHECKDEVNBR(ZF_DevNbr, ZG_Component_TotDev);
     M_HAL_CHECKPOINTER(PF_RegisterX);
  }
  /* functionality: Set Addr to RegisterX register */
  PL_RegisterXBaseAddress = &(ZG_pComponent[ZF_DevNbr]-> B_Component.B_RegisterX);
  /*Read RegisterX  and store by client */
/*Field: Value*/
/*access shielding */
  M_RD_REG(PL_RegisterXBaseAddress->Value ,   *PF_RegisterX);
  return E_HDSERR_NO_ERROR;
} /* End */
```



Hardware dependent SW (HdS)

HdS API

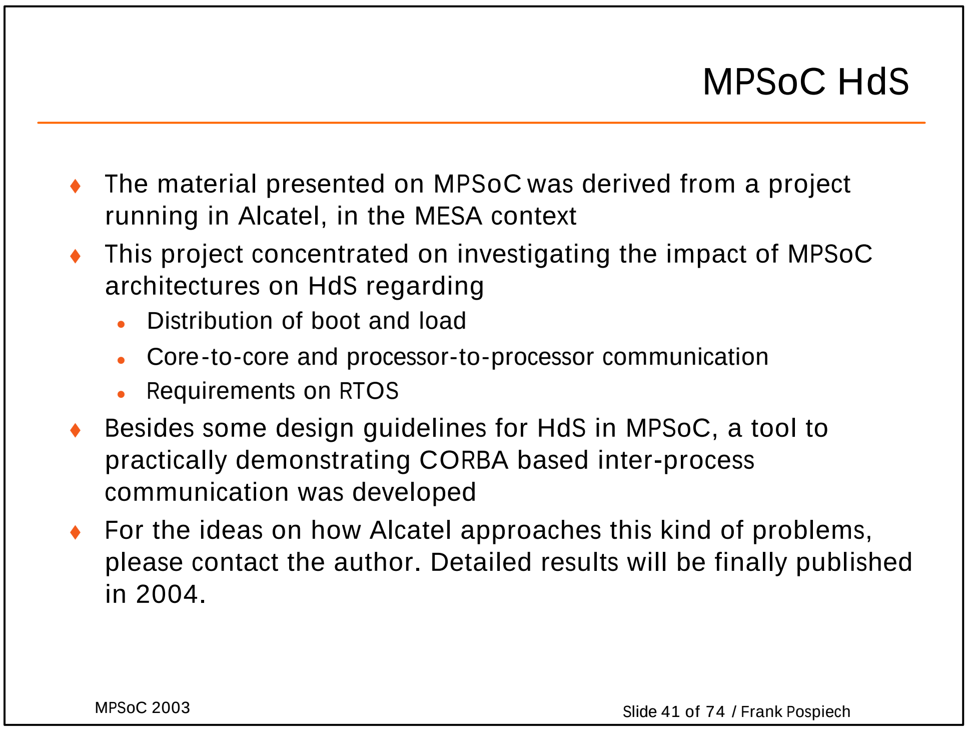
- The HdS API exposes the SoC's functionality to the upper layer SW.
  - The interface it offers is dependent on the application domain (i.e. multimedia, automotive, telecommunications).
  - Therefore the APIs may be different for different application domains.
- The API can be applied in all relevant development phases (design, verification, debug, production).
- By defining or fixing the HdS API, company internal and inter-company component re-use can be improved.

Conclusions

- Real Reuse of Test SW and effort reduction is possible throughout the product life cycle
- The enabler is the HAL architecture and an HdS supporting framework
  - It is itself reused throughout the complete product life cycle
  - It provides higher abstraction for application SW
  - It shields completely
  - It makes the SW portable through the access shielding towards whatever platform
  - It makes the SW scalable
  - Protection of SW as well as HW for misuse

Outline

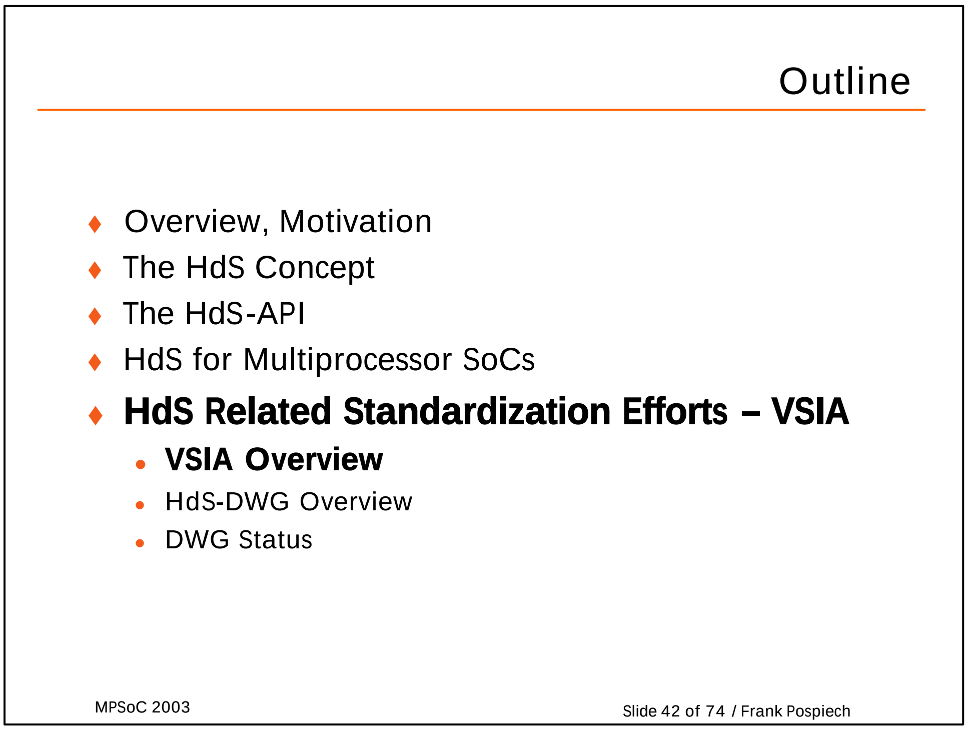
- Overview, Motivation
- The HdS Concept
- **HdS for Multiprocessor SoCs**
  - MPSoC System
  - HdS Communication System
  - Distributed CORBA Application Management
- HdS Related Standardization Efforts - VSIA



MPSoC HdS

- The material presented on MPSoC was derived from a project running in Alcatel, in the MESA context
- This project concentrated on investigating the impact of MPSoC architectures on HdS regarding
  - Distribution of boot and load
  - Core-to-core and processor-to-processor communication
  - Requirements on RTOS
- Besides some design guidelines for HdS in MPSoC, a tool to practically demonstrating CORBA based inter-process communication was developed
- For the ideas on how Alcatel approaches this kind of problems, please contact the author. Detailed results will be finally published in 2004.

MPSoC 2003 Slide 41 of 74 / Frank Pospiech



Outline

- Overview, Motivation
- The HdS Concept
- The HdS-API
- HdS for Multiprocessor SoCs
- **HdS Related Standardization Efforts – VSIA**
  - **VSIA Overview**
  - HdS-DWG Overview
  - DWG Status

MPSoC 2003 Slide 42 of 74 / Frank Pospiech

Ongoing Activities. VSIA
Why VSIA?

- The System on a Chip Revolution:
  - dilemma of exponentially growing density and more functionality in competition with the need for shorter development cycles
- Design Reuse
  - solution for this dilemma is to design with pre-designed blocks, much as is now done using off-the-shelf IC's on printed circuit boards
  - form of Intellectual Property (IP) - Virtual Components (VCs – the VSIA term for these elements)
- System-on-Chip Industry
  - Roadblocks to efficient and economical use: lack of open VC-to-VC interface standards upon which VC development and use can be based
  - system-chip industry:
    - who develop the infrastructure for system-chip designs (tools, services, and fabrication)
    - who design the system-chip devices themselves

Ongoing Activities. VSIA
Who is VSIA?

- Virtual Socket Interface Alliance (VSIA)
  - formed in September 1996
  - goal of establishing a unifying vision for the system-chip industry and the technical standards required to enable the most critical component of the vision: the mix and match of Virtual Components (IP) from multiple sources.
- VSIA Vision:
  - dramatically accelerate system chip development by specifying open standards
- VSIA Standards Philosophy:
  - "open" interface standards, which will allow Virtual Components to fit quickly into "Virtual Sockets", at both the functional level (e.g., interface protocols) and the physical level (e.g., clock, test, and power structures)
  - VSIA specifies existing de facto, or open, or proprietary (reasonable fee and non-discriminatory terms) data formats
  - VSIA **does not**: product development, price or business strategy decisions for individual members

Ongoing Activities. VSIA
VSIA Members

Among others, the following companies are VSIA members:

- SoC and system providers:
  - Alcatel, Analog Devices, ARM, Canon, Fujitsu, HP, IBM, Infineon, Intel
  - LSI Logic, Mitsubishi, Motorola, NEC, NOKIA, Nortel, Philips,…
- Tool providers:
  - Beach Solutions, Cadence, CoWare, Improv, LogicVision
  - Mentor Graphics, Synopsys, TransEDA, Verisity Design,…
- Research Institutes:
  - ECSI, Fraunhofer Institute, Forschungszentrum Informatik Karlsruhe,
  - IMEC, ITRI, Kyoto University, …
- RTOS Vendors
  - QNX,…

Ongoing Activities. VSIA
Embedded SW Study Group results (09/01)

- Large companies working in the SOC area have said VSIA should work in the Embedded Systems and Software areas! (Alcatel, Infineon, ST Microelectronics, ARM, Nokia, Philips, Motorola, …)
- Platforms are the reuse paradigm for the next level of issues VSIA is facing, with important interoperability issues and time-to-market reductions expected.
- Software reuse, for software that is intimate with hardware, is much more costly than it needs to be - there is a passionate desire for it, but the economic argument is weak due to poor interoperability
- Although we have identified many possible collaborators, there is no other standards body focused on this area

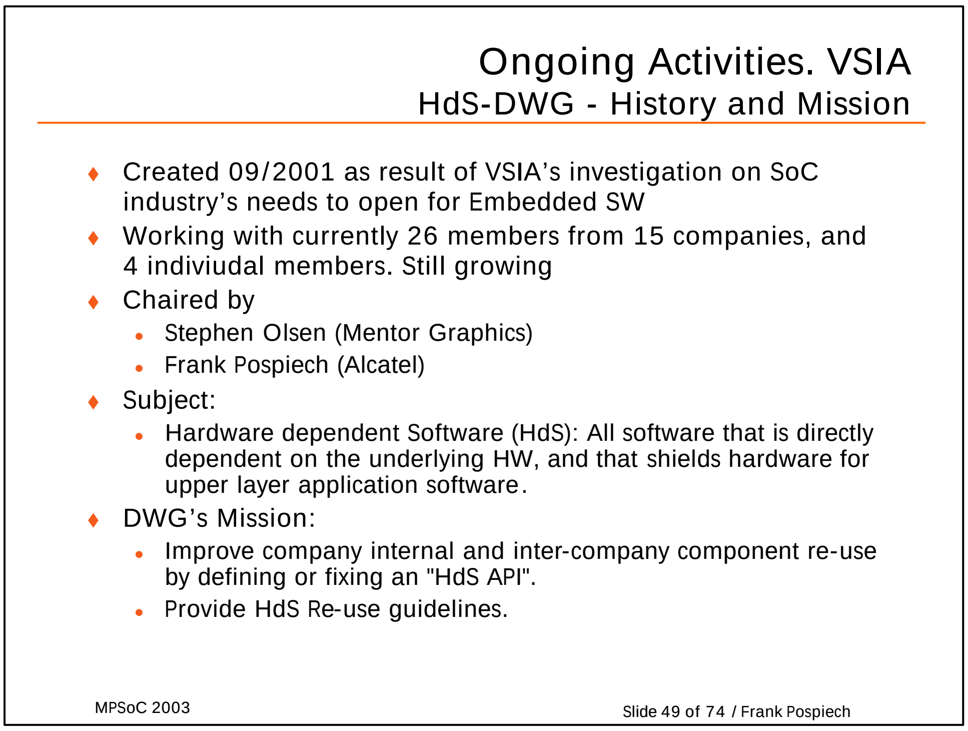
Ongoing Activities. VSIA

HdS-DWG: Action Plan: Suggested HdS Deliverables

- Taxonomy of HdS components
  - Precise definitions for HdS
  - Attributes and examples
  - Discovery process may lead to SoC API definition
- Deliverables for each class of HdS component
  - Including guidelines to maximise reuse or minimise porting ⇦ ***Critical!***
    - Guidelines for HdS components including design patterns and interfaces, and rules for reuse; define minimal interfaces and extensions
    - Language-independent; basic API possible for any language (C/C++/Java etc.). Include possibility of language extensions for optimality (like OCB)
- SoC API (HAL)
  - Definition and Examples
  - Processor, platform, language, RTOS-independent
    - Notion of Minimal or Basic API Plus extension templates (like OCB)
    - Scope out which part of RTOS covered (HW-dependent) and which not

Outline

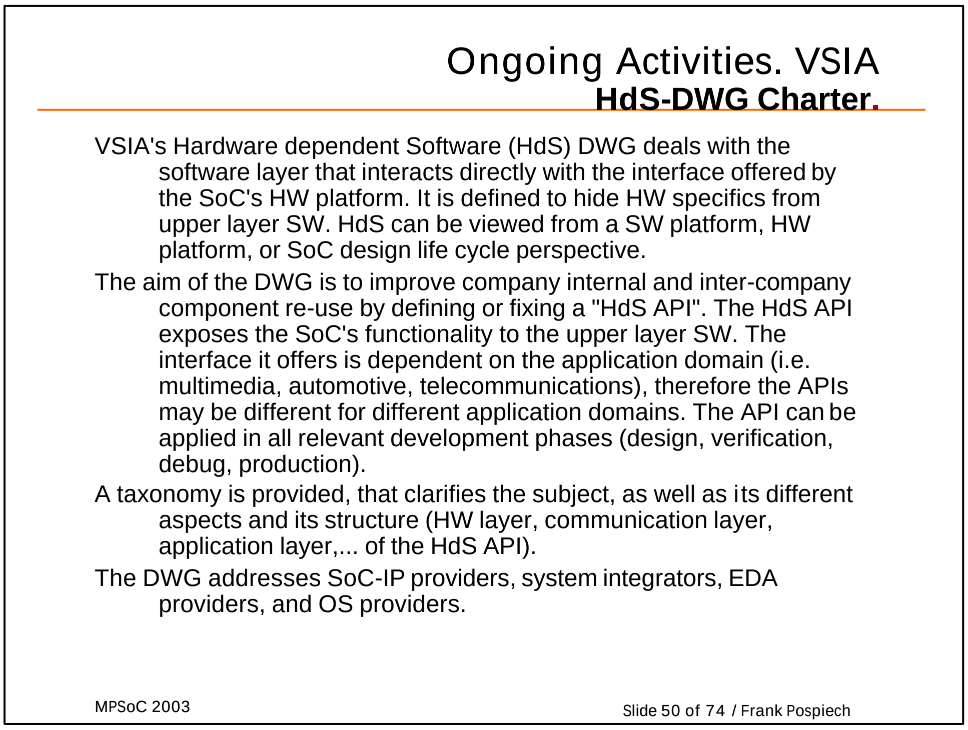
- Overview, Motivation
- The HdS Concept
- The HdS-API
- HdS for Multiprocessor SoCs
- **HdS Related Standardization Efforts – VSIA**
  - VSIA Overview
  - **HdS-DWG Overview**
  - DWG Status



Ongoing Activities. VSIA
HdS-DWG - History and Mission

- Created 09/2001 as result of VSIA's investigation on SoC industry's needs to open for Embedded SW
- Working with currently 26 members from 15 companies, and 4 indiviudal members. Still growing
- Chaired by
  - Stephen Olsen (Mentor Graphics)
  - Frank Pospiech (Alcatel)
- Subject:
  - Hardware dependent Software (HdS): All software that is directly dependent on the underlying HW, and that shields hardware for upper layer application software.
- DWG's Mission:
  - Improve company internal and inter-company component re-use by defining or fixing an "HdS API".
  - Provide HdS Re-use guidelines.

MPSoC 2003 Slide 49 of 74 / Frank Pospiech



Ongoing Activities. VSIA
HdS-DWG Charter.

VSIA's Hardware dependent Software (HdS) DWG deals with the software layer that interacts directly with the interface offered by the SoC's HW platform. It is defined to hide HW specifics from upper layer SW. HdS can be viewed from a SW platform, HW platform, or SoC design life cycle perspective.

The aim of the DWG is to improve company internal and inter-company component re-use by defining or fixing a "HdS API". The HdS API exposes the SoC's functionality to the upper layer SW. The interface it offers is dependent on the application domain (i.e. multimedia, automotive, telecommunications), therefore the APIs may be different for different application domains. The API can be applied in all relevant development phases (design, verification, debug, production).

A taxonomy is provided, that clarifies the subject, as well as its different aspects and its structure (HW layer, communication layer, application layer,... of the HdS API).

The DWG addresses SoC-IP providers, system integrators, EDA providers, and OS providers.

MPSoC 2003 Slide 50 of 74 / Frank Pospiech

Ongoing Activities. VSIA
DWG Member Companies

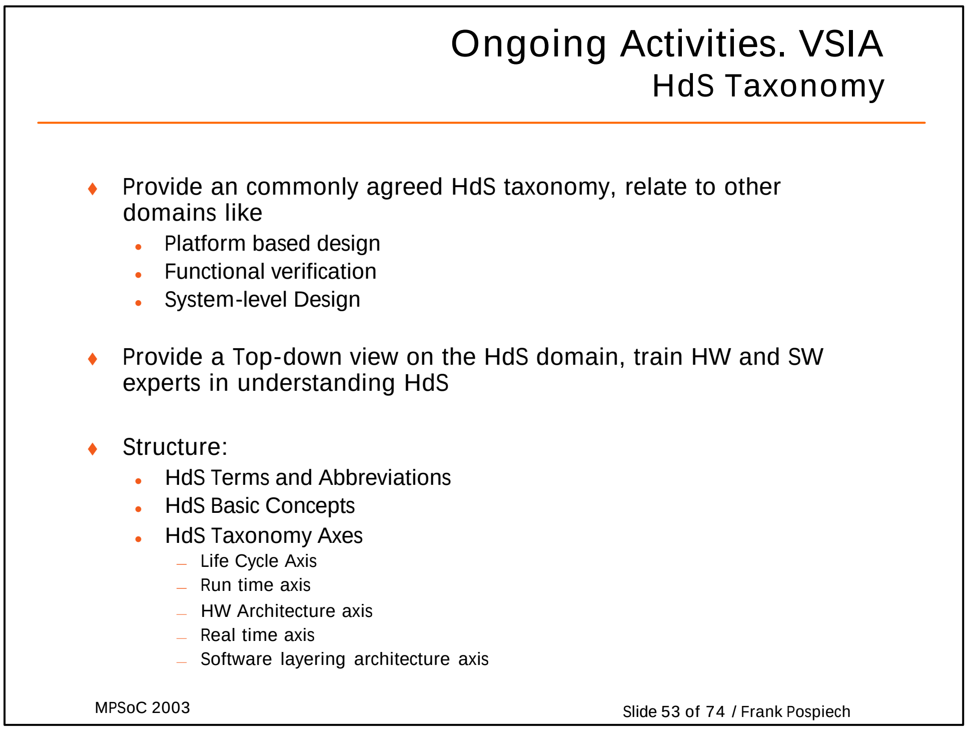
Member Companies:

- Alcatel
- Analog Devices
- ARM
- Beach Solutions
- Cadence Design Systems
- IBM
- Infineon
- Intel
- Mentor Graphics
- Nokia
- QNX
- ST Microelectronics
- Synopsys
- Toshiba

Ongoing Activities. VSIA
HdS-DWG - Current Main activities

♦ HdS Taxonomy:
- Consolidating terms from the SoC and the SW world, that need to be agreed upon in the HdS context
- Definition of HdS specific terms (HdS, HdS-API, kernel space, user space, driver, access shielding,…)
- Relate HdS concepts to different aspects, like life cycle, HW platform, SW platform, Real-time
- Determine HdS' architectural relation to HW, middleware and application software layers
- Provide a tutorial to train both HW and SW experts on HdS needs

♦ HdS-API:
- Define the characteristics of a common API, that defines the way how to provide SoC IP's functionality to upper layer Software.
- Define HdS-APIs for different application domains (telecommunications, automotive, multimedia,…)
- Define the HdS-API for single-processor and for multi-processor architectures.

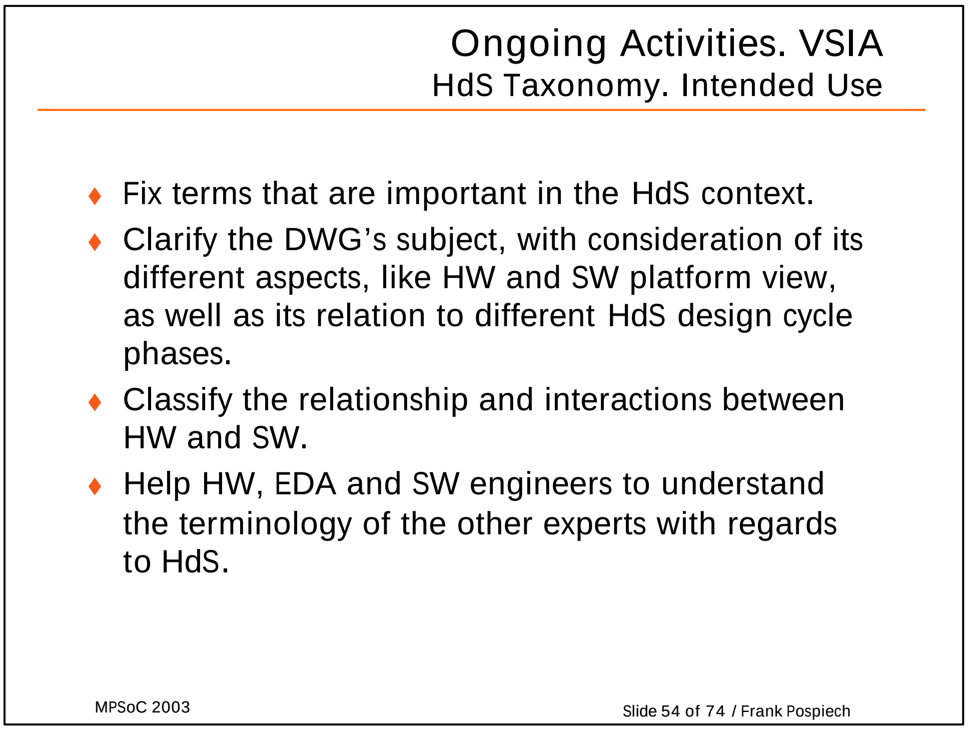


Ongoing Activities. VSIA
HdS Taxonomy

- Provide an commonly agreed HdS taxonomy, relate to other domains like
  - Platform based design
  - Functional verification
  - System-level Design
- Provide a Top-down view on the HdS domain, train HW and SW experts in understanding HdS
- Structure:
  - HdS Terms and Abbreviations
  - HdS Basic Concepts
  - HdS Taxonomy Axes
    - Life Cycle Axis
    - Run time axis
    - HW Architecture axis
    - Real time axis
    - Software layering architecture axis

MPSoC 2003

Slide 53 of 74 / Frank Pospiech



Ongoing Activities. VSIA
HdS Taxonomy. Intended Use

- Fix terms that are important in the HdS context.
- Clarify the DWG’s subject, with consideration of its different aspects, like HW and SW platform view, as well as its relation to different HdS design cycle phases.
- Classify the relationship and interactions between HW and SW.
- Help HW, EDA and SW engineers to understand the terminology of the other experts with regards to HdS.

MPSoC 2003

Slide 54 of 74 / Frank Pospiech

Ongoing Activities. VSIA
HdS Taxonomy. Intended Audience

- **HdS designers/engineers**. The taxonomy offers a vocabulary in the HdS domain and provides a means of unambiguous communication. It facilitates the definition of other related topics like the HdS API
- **HW designers and SW engineers**. For them, this taxonomy provides mainly a tool to understand the specifics of HdS, and to make efficient use of the HdS concept in their domain.
- **System architects/integrators/testers**. As primary users of the HdS-API, they need to understand the different aspects (life cycle, HW platform, runtime aspects) of the HdS concept.
- **EDA/IP Developers**: As tool and IP developers explore and automate functions for both Hardware and Software design and development, this taxonomy provides a structure for common understanding across both the users and the developers of these automated functions.

Ongoing Activities. VSIA
HdS-API

- Working on API description methods
  - UML notation based
  - Definition of
    - HdS Framework: HAL architecture
    - API mechanism: **How** to specify an HdS API
    - Elementary Services: Which are the basic services either to be offered directly to the SW, or of which services are composed of
- Mapping of the current proposals to the members' actual situation (telecommunications, multimedia, automotion,…)
- Stepwise refinement, starting with most common APIs (read, write, init,…)
- For some examples, see the HAL definition earlier in this presentation

Outline

- Overview, Motivation
- The HdS Concept
- The HdS-API
- HdS for Multiprocessor SoCs
- **HdS Related Standardization Efforts – VSIA**
  - VSIA Overview
  - HdS-DWG Overview
  - **DWG Status**

Ongoing Activities. VSIA
HdS-DWG - Objectives for 2003

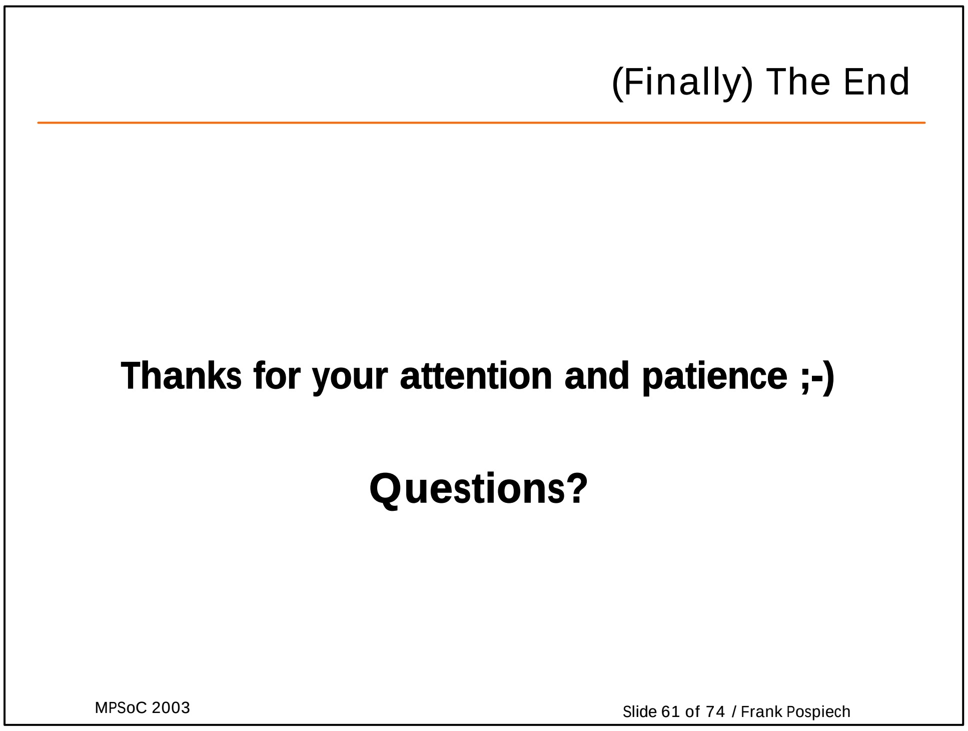
- Two major efforts are underway:
  - Taxonomy
  - Hardware Abstraction layer, HdS-API
- 2003 Goals:
  - Release taxonomy: 03/2003
    - Taxonomy is now in VSIA member review (till 06/2003)
  - HdS-API
    - Contents, syntax, general architecture: 2002
    - Example definition, first standards: 09/2003
- Software Virtual Component Specification / Rules to follow
- Currently: Concentration on Single-processor systems, MPSoC related HdS-API to follow.

Ongoing Activities. VSIA
HdS-DWG – Public Information Sources

- HdS DWG presentation on DAC 2002:
  http://www.vsi.org/events/dac02/cooke/slide01.htm
- HdS DWG presentation on DATE 2002:
  http://www.vsi.org/events/esc02/images/tiletkas.jpg
- HdS DWG presentation on DATE 2003:
  Hardware dependent Software Mastering the Gap between HW and SW Design
  (http://www.vsi.org/events/date03/date03hds.pdf)

Ongoing Activities. VSIA
HdS-DWG – You are Welcome to join us!

- Success of the HdS DWG depends largely on the broad competence of its members, and on the adoption by
  - Silicon vendors
  - RTOS providers
  - IP providers
  - EDA providers
  - System integrators
- Everyone, who shares our view of the need of a (more) standardized HdS-API to enable IP re-use is VERY welcome in our DWG.
- Please contact the author or http://www.vsi.org



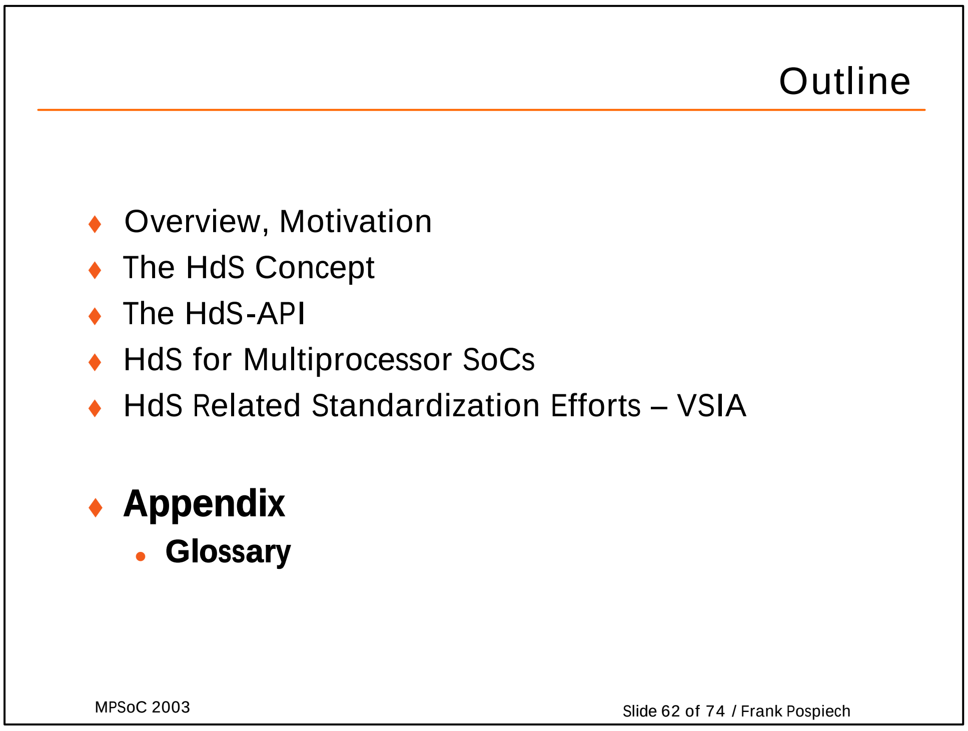
(Finally) The End

Thanks for your attention and patience ;-)

Questions?

MPSoC 2003

Slide 61 of 74 / Frank Pospiech



Outline

- ♦ Overview, Motivation
- ♦ The HdS Concept
- ♦ The HdS-API
- ♦ HdS for Multiprocessor SoCs
- ♦ HdS Related Standardization Efforts – VSIA

- ♦ **Appendix**
  - **Glossary**

MPSoC 2003

Slide 62 of 74 / Frank Pospiech

Appendix. Abbreviations (1/2)

API	Application programming interface
ASIP	Application Specific Integrated Processor
CORBA	Common Object Request Broker Architecture
CPU	Central Processing Unit
DAC	Design Automation Conference
DATE	Design Automation and Test Conference Europe
DSP	Digital Signal Processor
DWG	Development Working Group (VSIA)
GPP	General-Purpose Processor
HAL	Hardware Abstraction Layer
HdS	Hardware dependent Software
HW	Hardware
IP	Intellectual Property

Appendix. Abbreviations (2/2)

IPC	Inter-process Communication
MPSoC	Multi-processor SoC
OFLT	Offline Test
OS	Operating System
RISC	Reduced Instruction Set Computer
RPC	Remote Procedure Call
RTOS	Real-time Operating System
SoC	System on Chip
SUD	System under Design
SW	Software
VC	Virtual Component
VSIA	Virtual Socket Interface Alliance