MPSoC 2004

An HdS Taxonomy as one of the first steps towards closing the gap between HW and SW design

Frank Pospiech

S.E.S.A. AG Berlin, Germany

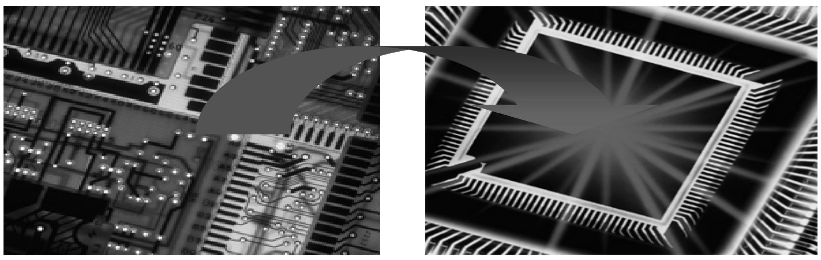
Frank.Pospiech@sesa.de

Outline

- ♦ **Motivation**
  - **Software Related issues in modern SoCs**
- ♦ HdS Concept
  - HdS
  - HdS-API
- ♦ HdS Related Standardization Efforts in VSIA
  - VSIA is Changing
  - Work on HdS in VSIA
  - HdS Taxonomy

Software Challenges for SoC Design (1)

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

Software Challenges for SoC Design (2)

- 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

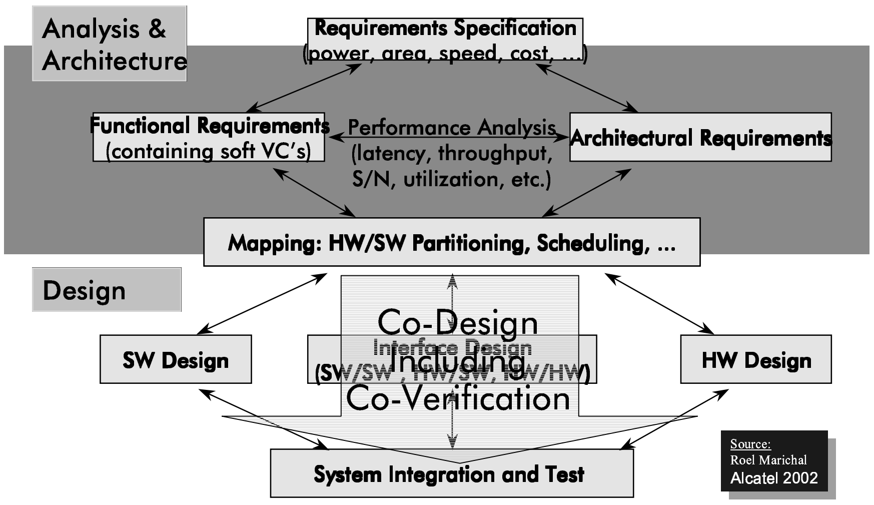
- It is useful to develop a standard abstraction for the software that "sits directly on top" of the hardware…

  Hardware Dependent Software (HdS)

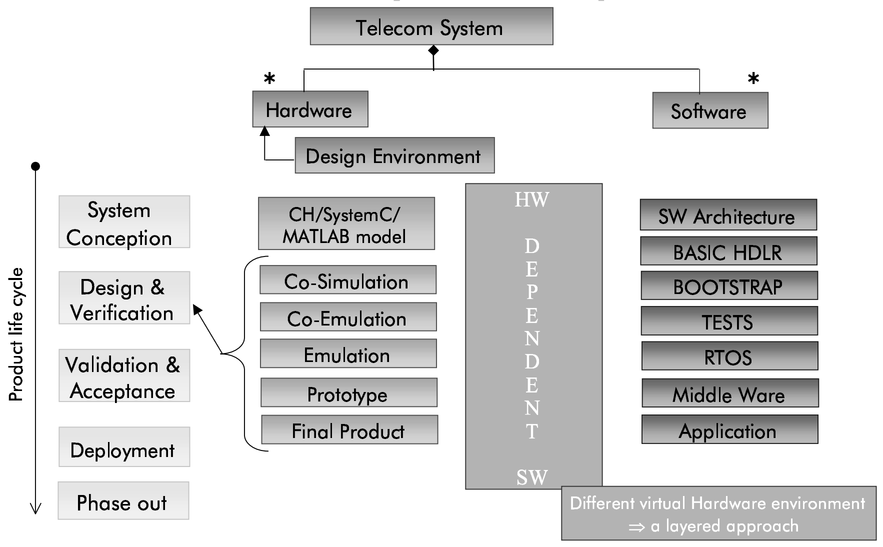
Outline

- Motivation
  - Software Related issues in modern SoCs
- **HdS Concept**
  - **HdS**
  - HdS-API
- HdS Related Standardization Efforts in VSIA
  - VSIA is Changing
  - Work on HdS in VSIA
  - HdS Taxonomy

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
Co-Design Including Co-Verification
Interface Design
(SW/SW , HW/SW, HW/HW)
SW Design
HW Design
System Integration and Test
Source:
Roel Marichal
Alcatel 2002

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



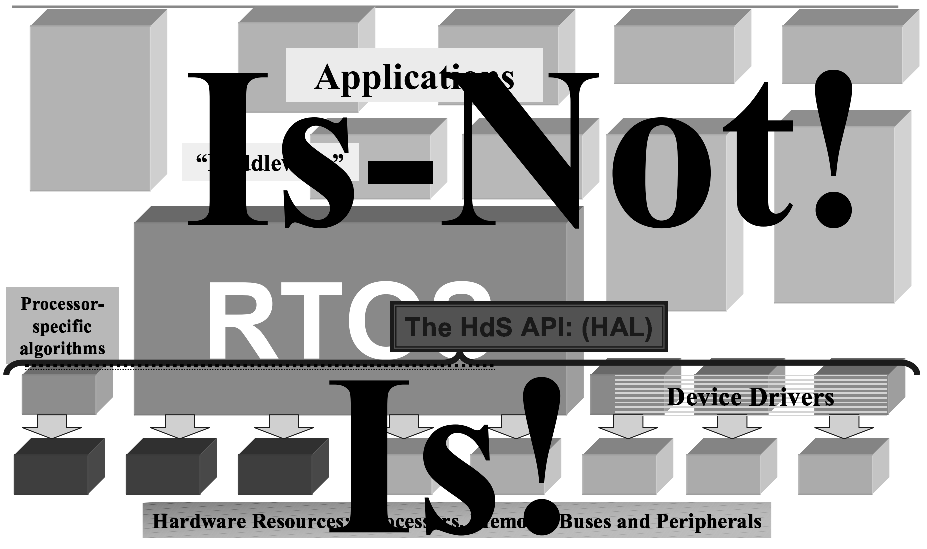
Hardware dependent SW (HdS)
Definition: Hardware-Dependent Software

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

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

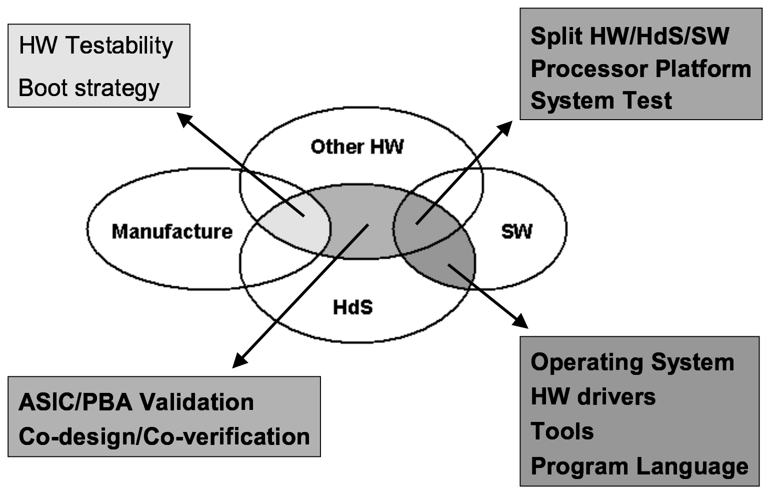


Applications
"Middleware"
RTOS
Processor-specific algorithms
The HdS API: (HAL)
Device Drivers
Hardware Resources: Processors, Memory, Buses and Peripherals
Is-Not!
Is!

MPSoC 2004

Slide 9 of 41 / Frank Pospiech

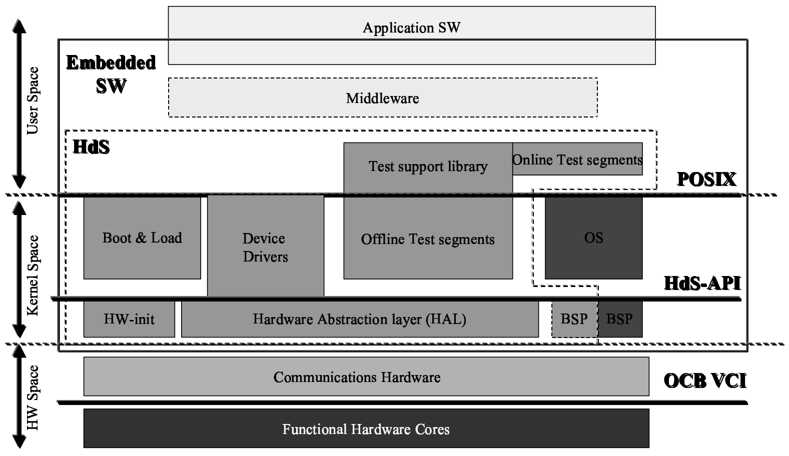
Hardware dependent SW (HdS)
HdS Links to other disciplines



MPSoC 2004

Slide 10 of 41 / Frank Pospiech

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



SoC Design needs a New Standard

Standardization

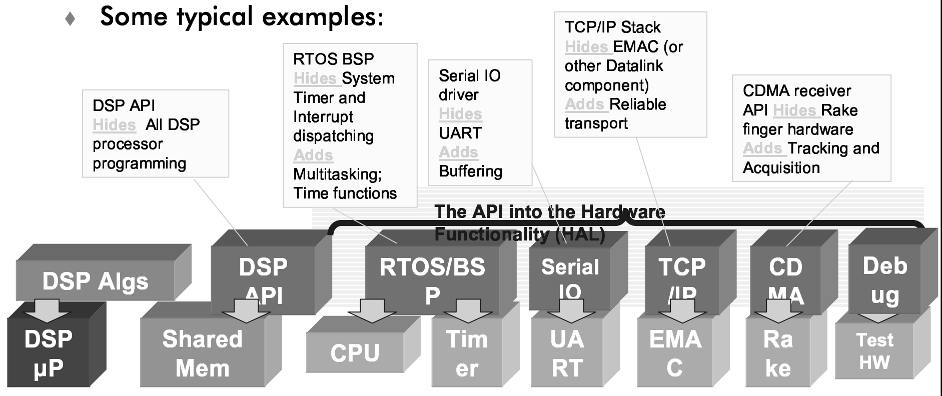
- Many standards are defined for higher level SW
  - Communication (CORBA,...)
  - High level languages, specification languages (ADA, Modula, UML, SDL,...)
  - OS Interface (POSIX, OSDL-CGL,...)
- In the HdS domain, standardization is still quite poor
  - Activities started (VSIA, see later in this presentation)
  - Others in other domains (Automotive,...)

Outline

- Motivation
  - Software Related issues in modern SoCs
- HdS Concept
  - HdS
  - **HdS-API**
- HdS Related Standardization Efforts in VSIA
  - VSIA is Changing
  - Work on HdS in VSIA
  - HdS Taxonomy

Hardware dependent SW (HdS)
The SoC as an API

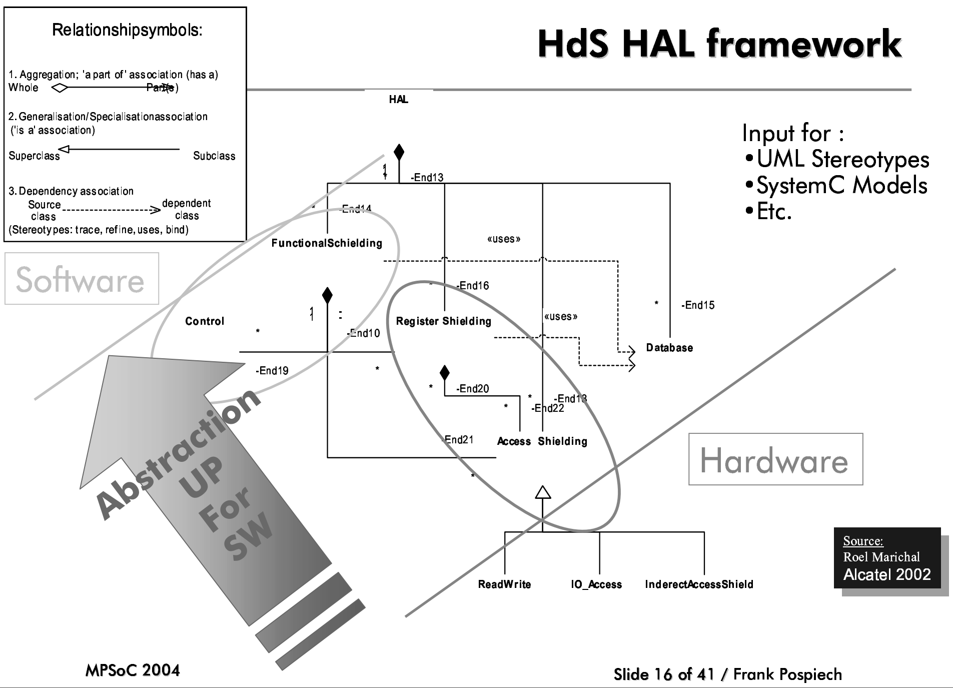
- An SoC looks like an API to most embedded SW developers
- SoC functionality needs an abstraction layer, which may also extend the capabilities, the **Hardware Abstraction Layer (HAL)**
- Some typical examples:


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
CDMA
Debug
DSP µP
Shared Mem
CPU
Timer
UART
EMAC
Rake
Test HW

Hardware dependent SW (HdS)
HAL and shielding

- The **Hardware Abstraction layer (HAL)** is the **ONLY** gate towards the HW, shielding register access from application providing a SW interface.
- HAL is based on:
  - structures and/or arrays of basic data types
    ---> **Register Shielding** ≡ first degree of abstraction.
  - Allow SW clients to use the device, without the need of in-depth knowledge of the device.
    ---> **Functional Shielding** ≡ 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.

HdS HAL framework



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

Very Basic Example

```
/* A function of the functional shielding layer*/
Xxx_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 */
```

Functional Shielding

Register Shielding

Access Shielding

Hardware dependent SW (HdS)

HdS API, What and why

- 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.

Outline

- Motivation
  - Software Related issues in modern SoCs
- HdS Concept
  - HdS
  - HdS-API
- **HdS Related Standardization Efforts in VSIA**
  - **VSIA is Changing**
  - Work on HdS in VSIA
  - HdS Taxonomy

VSIA
Mission of the "old" 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

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

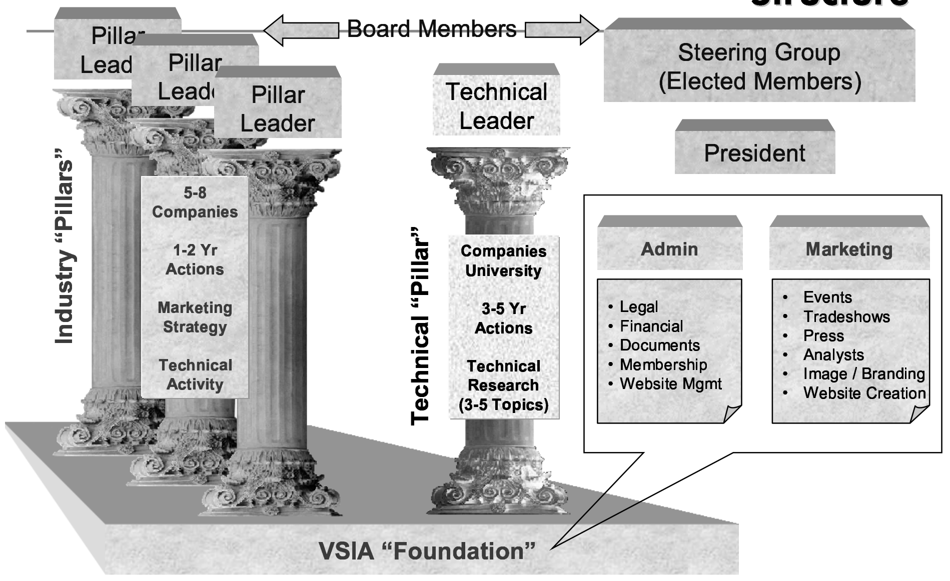
VSIA
VSIA's New Mission 2004

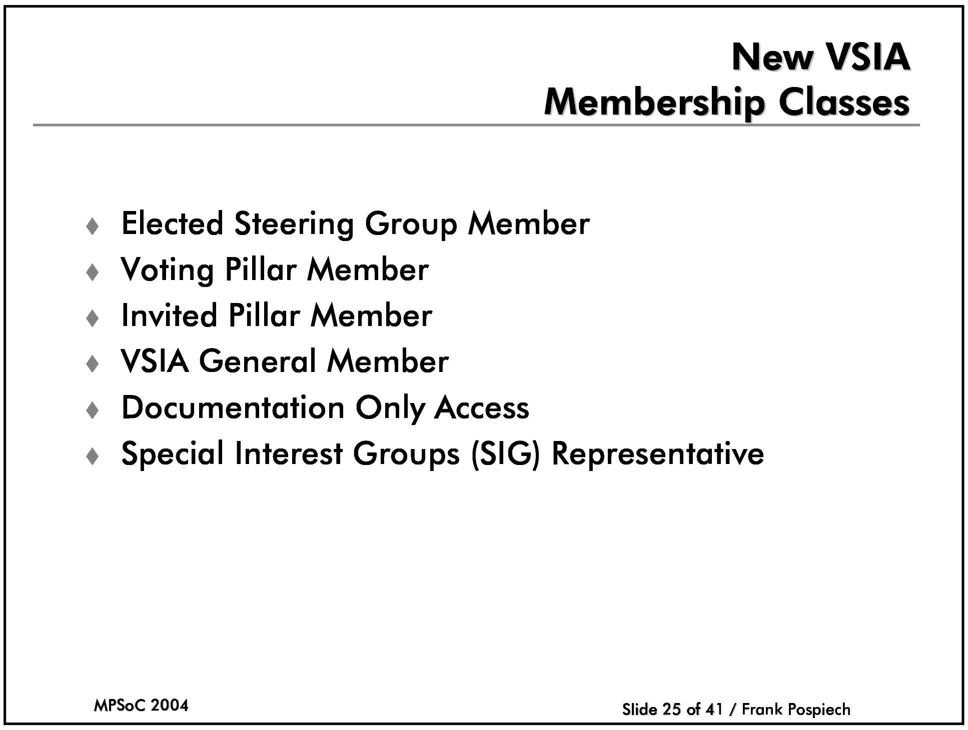
"Dramatically enhance the productivity of the SoC design community by providing leading edge business and technical solutions and insight into the development, integration and reuse of IP"

New VSIA
Statement of VSIA Value Proposition

- Focus on issues related to IP reuse
  - Business to Business (as opposed to "tool to tool")
  - Platforms
  - Hardware Integration
  - Quality
  - Protection
  - Hard and soft reuse
- Incubator for new standards
  - Sufficient (full service)
  - Efficient (timely)
  - Proficient (expertise)
- Output has substantive impact within 1-2 years
- Investigate emerging issues (3-5 years)
  - Work with R&D Departments
  - Universities
- Host libraries and information exchange related to IP reuse

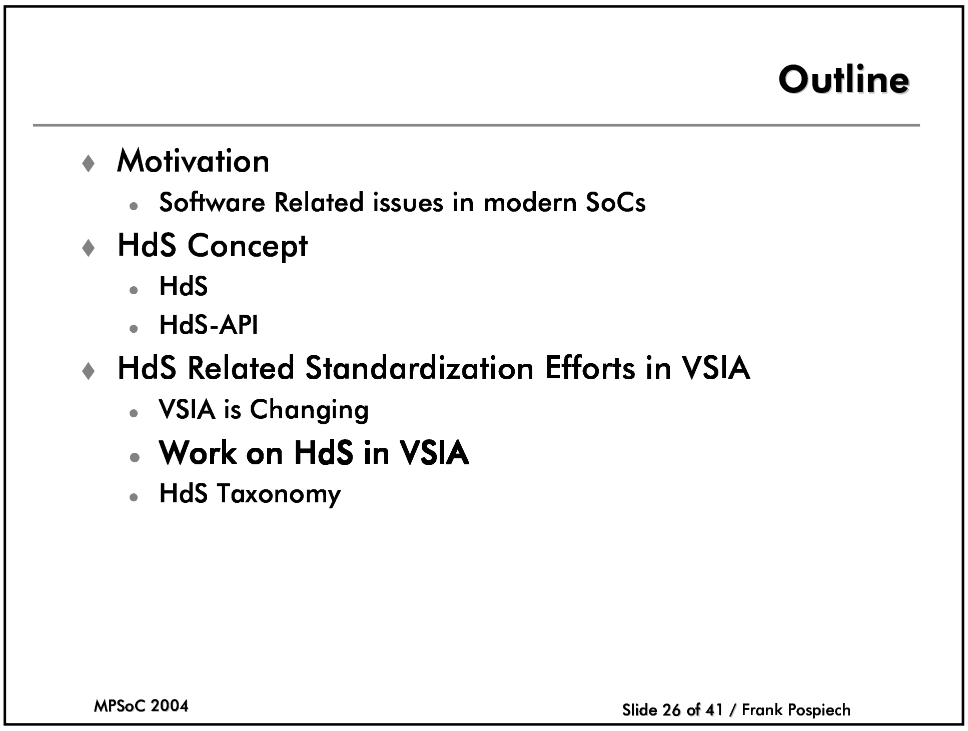
New VSIA
Structure





New VSIA Membership Classes

- Elected Steering Group Member
- Voting Pillar Member
- Invited Pillar Member
- VSIA General Member
- Documentation Only Access
- Special Interest Groups (SIG) Representative



Outline

- Motivation
  - Software Related issues in modern SoCs
- HdS Concept
  - HdS
  - HdS-API
- HdS Related Standardization Efforts in VSIA
  - VSIA is Changing
  - **Work on HdS in VSIA**
  - HdS Taxonomy

VSIA and HdS
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 up to 26 members from 15 companies, and 5 indiviudal members
- Chaired by
  - Michael Kaskowitz / 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.

VSIA and HdS
HdS-DWG - Main activities

- HdS Taxonomy:
  - Clarifies basic concepts in the HdS domain
  - Defines the vocabulary and language needed for clear communications between groups, especially between HW and SW designers
  - Puts HdS into its context of life cycle, software and hardware architecture, and platform design
  - Allows HW and SW experts to understand the specifics of Hardware dependent Software
- 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.

VSIA's HdS Taxonomy
HdS Taxonomy Motivation

- Main benefit expected from **fixing the HdS-API**
- **However**:
  - Still missing awareness on HdS issue as method to close the gap between HW and SW design
  - HW designers and SW designers speak different languages
  - Common architecture, common design flow perception

→ 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

MPSoC 2004

Slide 29 of 41 / Frank Pospiech

VSIA's HdS Taxonomy
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 2004

Slide 30 of 41 / Frank Pospiech

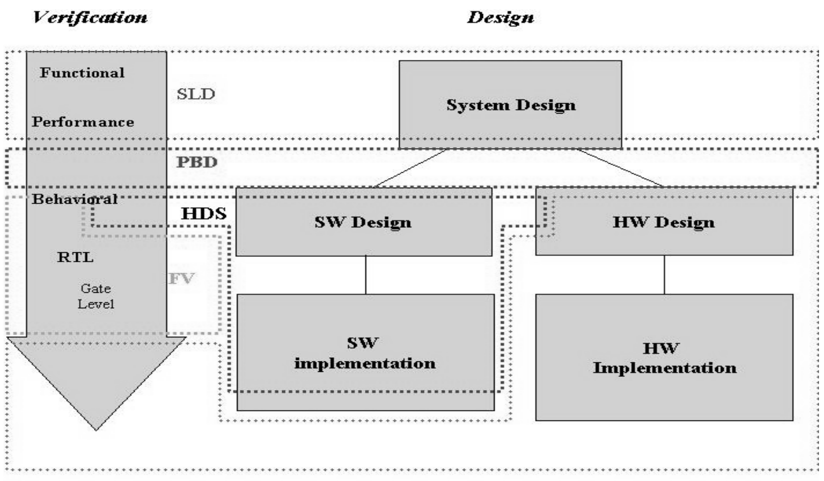
VSIA's HdS Taxonomy
HdS Taxonomy. Intended Audience

- **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.

VSIA's HdS Taxonomy
Structure

- HdS Terms and Abbreviations
- HdS Basic Concepts
- HdS Taxonomy Axes
  - Life Cycle Axis
  - Run time and Real-time axis
  - HW Architecture axis
  - Software layering architecture axis

VSIA's HdS Taxonomy
HdS Relationship to other Disciplines



VSIA's HdS Taxonomy
HdS Terms

- Example for Basic HdS Terms (Chapter 3.1.2):

The **Hardware Abstraction Layer (HAL)** is a software layer that interfaces to the underlying hardware. This layer provides **shielding** of the hardware access and functionality through simplified or standard interfaces that isolate the hardware complexity from the software developer. The HAL consists of three sub-layers: the access shielding, register shielding and functional shielding layer. The **access shielding layer** contains the mapping between the variable names and the addresses of the memory locations; macros enable the use of the HAL in simulations. The **register shielding layer** enables access to the memory locations via read and write functions. The **functional shielding layer** allows higher layers to use a device, without knowing all details of the device.

- + Dictionary of ca. 70 HdS related terms with definition.

VSIA's HdS Taxonomy
HdS Taxonomy Axis (1)

- **Life Cycle Axis**: Relationship/Importance of HdS during different life cycle phases like:
  - System Development
  - SW/HW Co-development
  - Debug/Optimisation
  - Use
  - Retargeting
  - Variant or Derivative development
  - Re-use
- **Run Time and Real Time Axis**: Typical HdS modules responsible for
  - Boot, Configure, Execute, Load,...
  - Scheduling and interface timing characteristics,...
  - Communication mechanisms

VSIA's HdS Taxonomy
HdS Taxonomy Axis (2)

- **HW Architecture Axis:** different aspects of HW platform, seen from the software point of view, e.g.
  - Architecture Synopsis
  - Code levels (Micro code, Instruction set code, ...)
  - OS requirements (Memory, interrupts,...)
  - Architecture of Software defined by Hardware (CPU Subsystem, Memory subsystem,...)
  - Multiprocessor Architectures, and SW aspects

VSIA's HdS Taxonomy
HdS Taxonomy Axis (3)

- **Software Layering Axis:** layered model for embedded software, as well as the HdS API as one of the key concepts in the HdS domain (VSIA reference model, see earlier in this presentation)
- Different views:
  - **VSIA reference model** with
    - *Hardware Layer*
    - *Primitive Function Layer*
    - *Inter-API Communications Layer*
    - *Interface Access Layer*
    - *OS Layer*
    - *Application Layer*
  - **Control, Data, Hardware and Software Layering** (classification according to modelling and implementation means)
  - **HdS API**

VSIA's HdS Activities
Status

- HdS Taxonomy released to VSIA in 09/2003
- HdS Taxonomy released to Public in 02/2004

- HdS API – First draft existing, still much work necessary

- Due to VSIA restructuring, there are no more DWGs
- So HdS-DWG as own group does no more exist
- Investigation, if HdS work can be continued in VSIA's R&D Pillar

- Activities to be continued potentially in other initiatives

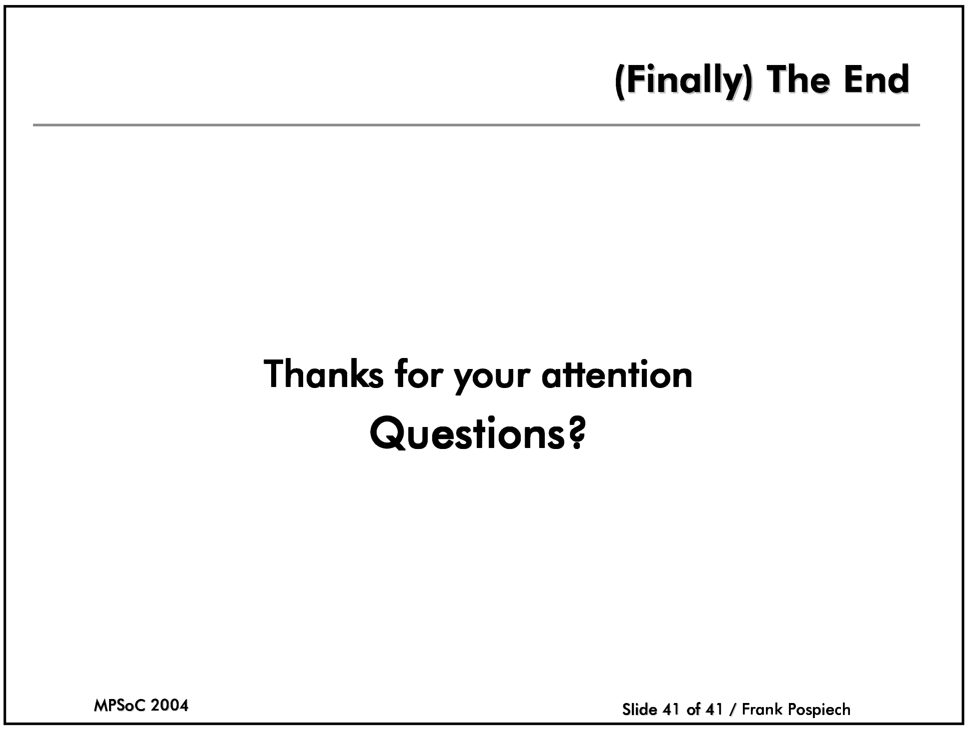
Conclusion

- HdS-API useful for improvement of Re-use (across life-cycle, across platforms)
- With a defined HdS-API, potential for design and test automation
- Work started in VSIA, comprehensive taxonomy achieved
- Still work open for fixing the HdS-API
- With moving VSIA focus, currently few activity in this context

→ New roof for this important work to be investigated.

VSIA and HdS
Public Information Sources

- HdS DWG presentation on DAC 2002:
  http://www.vsi.org/events/dac02/cooke/slide01.htm
- 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)
- HdS presentation on MPSoC 2003:
  HdS For Multiprocessor SoCs. An Introduction. http://tima.imag.fr/mpsoc/2003/lectures.html - Pospiech
- HdS DWG presentation on DesignCon 2004:
  9-TP1: A Primer to the World of Hardware-Dependent Software



(Finally) The End

Thanks for your attention

Questions?