Open Source Hardware:
Simulation with Virtual Hardware

Mladen Berekovic, Computer Engineering, TU Braunschweig
Jan Haase, Computer Engineering, University of Lübeck
Design Methodologies Demands

- Design Tools & Methodologies have to reflect increased complexity
- Introduction of formal methods
  - Worst-Case Execution Time Analysis (WCET)
- Early System Validation of Hardware & Software Components / Integration
- Faster Simulation Methods

Virtual System Prototyping
What is a virtual prototype?

Demo System on a Virtual System Platform

Virtual System Platform (VP)

- Is a fast functional (SW-) Model of an embedded system (e.g. ECU – real or future)
- Consists of a (fast) Instruction-Set Simulator (ISS) and a set of models of chip- and board components (Bus, Memory, etc.)
- Executes the same Binaries as the (ECU) Hardware
- Runs on Standard Compute Platforms
- Uses Electronic-System-Level (ESL) Design-Techniques based on SystemC and TLM for a fast Hardware-Software Simulation

A Virtual System Platform looks to the OS / Applications like real hardware
Virtual System prototyping in the Automotive Development Process – OEM as architect and integrator

- Separation of technical and functional integration
  - Tier1: Secures Execution of SW (technical integration)
  - OEM: Tests Functionality of a subsystem (functional integration)
- **Virtual Prototyping**: Early Availability of an „executable spec“ within the supply chain
Electronic-System-Level (ESL) Design: Virtual System-Platform with SystemC/TLM

System-Design
Functional Design

Architecture-Design

Implementation

- Digital and Analog Simulation,
- Unified System-Environment for Mixed Signal
- VERY fast Digital-Simulators (real-time feasible).

Source: Denso
Concrete: ECU as Virtual System-Platform

- Complete Software-Simulation of the Hardware/Software-System
- Hardware-in-the-Loop (HILS) ECU replaced with Software-based “Virtual Hardware-in-the-Loop” (virtual HILS).
Virtual Hardware-in-the-Loop (vHILS)

ECU-Model
- Driver-Assistance-System ECU

Virtual System Platform

CAN Modell (Bus Surveillance)

Realizable through Virtual Prototype

Vehicle Model
- Motor
- Vehicle Electronics
- Human-Machine-Interface
- Sensors

Test Specification
- Displays
- Inputs

Event-Processing

Vehicle

Street Model

Virtual System Platform
ES Software Development with Virtual Prototypes

Model Based
- Model Level
  - Model
    - Model-in-the-loop
      - Abstract Model
      - Planned Model Or Stimulus
    - Software-in-the-loop
      - C Code on Host System
      - Planned Model Or Stimulus
  - Model
    - Model-in-the-loop
      - Abstract Model
      - Planned Model Or Stimulus
    - Software-in-the-loop
      - C Code on Host System
      - Planned Model Or Stimulus

Code-Based
- Untimed, Host-Data-Types
  - Software
    - C Code on Host System
    - Planned Model Or Stimulus
- Untimed, Target-Data-Types
  - Virtual Hardware
    - C Code on Platform
    - Planned Model
- Timed, Target-Data-Types
  - Virtual Hardware
    - C Code on Platform
    - Planned Model
- Real Hardware
  - Processor-in-the-loop
    - C Code on Target Platform
    - Planned Model

Source: Hitachi
Software-Design on Virtual System-Platforms (e.g., with Cadence VSP Tool)

Early Software- and System-Development
- High-Performance-Simulation, runs with Production-Software
- Higher SW-Design-Productivity with Multi-core-HW/SW-Debugging

Fast Platform Design, Debugging und Analysis
- HW/SW-Debugging
- Plattform- and Functional-Profiling, to localise Bottlenecks

Validation of Hardware/Software-Integration 1 year before Availability of Hardware
- Unified HW/SW-Simulator for System-Validation (e.g., with Cadence VSP)
- Enables finer Synchronisation between HW- & SW-Design in the Development cycle

Source: Cadence
Tool-Support for Embedded-Software-Design (ESD) on virtual Platforms

„Programmer’s View“ of Hardware models only those aspects of hardware that are needed for software debugging. All other Hardware Aspects are abstracted from the SW-developer.

- **Programmer’s View of Hardware**
  - Processor- und Peripheral-Registers „viewing“ and trace
  - Memory View
  - Interrupt/Reset Trace
  - “Waveform” type view of interrupts, resets, HW-Register-accesses
  - Peripherals accessible through a processor

- **Processor/Core Sensitive GUI Views**

- **“One-click” Launch & Debugging of several ESD-Executables on a single VP**

- **Full ESD-Debug of Multi-core und Multi-Processor**

- **Non-intrusive ESD Trace, Analysis & Profiling finds more problems than ESD-Debug**
  - Line, Function coverage, Memory Allocation Trace …

Source: Cadence
Tools for the ESD Design on virtual Platforms

Multi-Core ESW Debug GUI

VSP-Simulator

Common ESW Debug Command Interface

SW-Coverage and Analysis
Symbolic Debugging
Multi-core-Synchronization

Processor Access Layer (PAL)

PAL Client
- ARM Fast Model*

PAL Client
- Imperas OVP Processor Model*

PAL Client
- 3rd Party Processor Model (QEMU)

*Integrated by Cadence in VSP

Source: Cadence
Realization of Virtual System-Platforms with Electronic-System-Level Design Methodology (ESL)

- **ESL** is a standard that is based on **SystemC/TLM** (IEEE 1666-2005) (C/C++ based) for fast Hardware-Simulation.

- HW und SW are developed in a **single** language (C++) instead of C/C++/Java and Verilog.

- **ESL** is based on the simplified modeling of HW/SW-Systems from a communication perspective (**TLM**: Transaction-Level Modeling).

- A System-Modeling on Transaction Level (**TLM**), delivers a speedup of the HW/SW-System Simulator by a factor of of 1000-10000. This enables **virtual prototyping** on a workstation without real hardware („Executable Spec“).

- A System-Modeling on Transaction-Level (**TLM**), is possible on 3 levels of detail that can also be mixed (mixed-mode simulation) for verification and validation purposes
  - LT (loosely timed): Abstract model for fast software verification,
  - AT (approximately timed): 90% Cycle-true, 90% speed
  - CT (cycle-true): Fully detailed hardware model

- The high Abstraction level together with the high simulation speed enables prototyping and design space exploration of complex hardware/software systems such as ECUs.
HW/SW-Integration in the system design

Validation of the HW/SW-Integration as part of the development cycle

Source: Cadence
HW/SW-Design with Transaction-Level Modeling (TLM)

- **Separation** of Function und Communication (Orthogonality)
- Function und Communication each adequately abstract
- Refine Component, Communication as coarse transaction of abstract Data packages
- Abstract Component, but precise Bus-Protocol (e.g. Bus-data traffic estimation)
- **Virtual System-Platforms** (VP), Simulation of Function and Communication
- Virtual Processors are **binary- and register compatible**
- Hybrid (TLM/RTL) Model für cycle-true simulation possible (SystemC and Verilog joint simulations possible; hybrid mixed-mode-simulation)
System-Realization with SystemC and TLM

- Specification
- SW-Modeling
- Target-Code-Generation
- Virtual-Platform-Generation

Simulation
- Co-Simulation
- Co-Verification
- Performance estimation

Specifikation
- TLM-Model-Partitioning

TLM
- Software
- Communication
- Hardware

SW-Synthesys
- Comm-Synthesys
- HW-Synthesys

C++

Target System Implementation

Software-Implementation

Hardware-Implementation

Time, Data, ...

more detailed

more abstract
TLM-based Module-Design accelerates Time-To-Market

~3 x compact Design, efficiency increase

5–10 x faster Verification

SystemC → RTL in 10 days. vs. manual RTL in 3 month

FPGA – reconfig in 1-2 days. vs. manual in 2-3 weeks

Faster Design-Space-Exploration: “Do we reach this with 200 MHz?”

>10 x Productivity-increase

Source: Cadence
Conclusion

- New C++ based Modeling Technology that focuses on communication
- Single language for hardware and software eases hardware/software co-design
- Abstract Models offer faster path to modifications, easier maintainability
- Fully binary compatible VP models at different levels of granularity
- Simulation speeds close to real-time make use of VPs in design space exploration possible (“Virtual Hardware Platform in the Loop”)
- Enables replacement of ES Hardware System Design Platforms with pure Software Solutions
- New Methodologies for non-functional requirements validation, e.g. Fault Injection, reliability, power, ….