Virtual PLATFORMS for complex IP within system context

VP Modeling Engineer/Pre-Silicon Platform Acceleration Group (PPA)

November, 12th, 2015

Rocco Jonack
Legal Notice

• This presentation is for informational purposes only. INTEL MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS SUMMARY.


• *Other names and brands may be claimed as the property of others.

• Microsoft, Windows, and the Windows logo are trademarks, or registered trademarks of Microsoft Corporation in the United States and/or other countries.

• Java and all Java based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

• Bluetooth is a trademark owned by its proprietor and used by Intel Corporation under license.

• Intel Corporation uses the Palm OS® Ready mark under license from Palm, Inc.

• Copyright © 2015, Intel Corporation. All rights reserved.
Agenda

- Problem Statement
- VPs for Different Integration Systems
- Development of Versatile VPs
- Validation of VPs
- Conclusions
PROBLEM STATEMENT
Problem Statement (1/2)

- SoC consists of many complex subsystems
- Tools for FW, driver and SW development on system level are needed
- Subsystems have to be integrated with various integration environments

Example for heterogeneous SoC architecture
Problem Statement (2/2)

• What is currently used for pre-silicon development and validation?
  ▪ Virtual Platforms in many flavors ➔ Does it support my use model?
  ▪ N-1 hardware ➔ Only works with minor HW changes
  ▪ FPGA ➔ Only available with stable RTL
  ▪ Hardware accelerator (SLE) ➔ Require significant setup

Versatile VPs, usable for all required scenarios, are needed
VPs FOR DIFFERENT INTEGRATION SYSTEMS
Early validation of FW

VP augments other methods for pre-silicon development

Focus on subsystem
  • Host and memory environment is emulated
VP for Driver/SW Development

- Development focus is guest OS
  - End user interacts with guest OS
  - Requires high responsiveness and error resilience
  - Simics platform is providing platform for OS

- OS interacts with subsystems
  - Boot flow, power scenarios, security, etc.

Simics platform model running an OS

Host

Memory

Fabrics

Device
Complex flows involving several subsystems

- For example: “listening to audio stream over bluetooth”

- Requires collaterals from FW, driver and SW team
VP as Part of Development Flow

Architecture Design

Executable Specification

Virtual Prototype

SW Development

Test SW Development

for System Verification

Chip Development & Verification

Test SW Development

for System Verification

Accellera

Systems Initiative
DEVELOPMENT OF VERSATILE VPs
Infrastructure Requirements

• Versatile VP development needs:
  ▪ A library of base components for common functionality
  ▪ Appropriate 3rd party models
  ▪ Automatic generation of formally specified parts of SoC
  ▪ Standard interfaces for easy integration
  ▪ Efficient and versatile mechanisms to test and exchange software components
  ▪ Unified build system ensures support for different environments
ISCTLM Generic VP Development Flow

- VP development has to be supported by a productivity library
  - ISCTLM contains registers, memories, interconnects, etc.
  - Code generation helps with productivity
  - Using 3rd party IP when applicable

DUT: Device Under Test
ISCTLM: Intel SystemC TLM Library – provided by ETS
TLM: Transaction Level Modeling
Simics Infrastructure and Platform

Simics @ Intel: Client, Server, SoC IA platforms simulators

- **Unmodified software**
  (BIOS, FW, drivers, OS, applications)

- **Complete platforms**
  (system level use cases)

- **“Black box” models**

- **Detailed models**

- **High performance**

- **Functionally accurate**

- **Full system simulation**

- **Standard tools support**
Generic SoC VP Setup
Using Host Based Emulation

• Alternative to running on ISS
  – ISS provides detailed model, but slow execution
  – ISS may not always be available

• Host based simulation
  – Same interface for register access and IRQ handling
  – Instructions are executed on host machine
VP for FW Development

- Model development starts with a simple setup where host CPUs are modeled simplified without OS.

- For host emulation a ISCTLM emuCPU was applied, running C code test scenarios based on adaptable bus access macros.
VP for Driver/SW Development

- Integration into the Simics Platform with OS
- The previous test cases can be reused through OS drivers with minimum functionality bridging between user and kernel space.
- Customer FW and drivers can be integrated
VP for Validation of System Level Flows

- Integration into Simics Platform with OS
- OS is interacting with multiple subsystems
- Complex synchronization scenarios can be tested
VALIDATION OF VPs
Requirements for Test Cases in Versatile Platforms

- Test cases have to be reused for all integration systems in the course of development
  - Some use cases might not be relevant, but status has to be clear
- Test case description and execution requires common layer
  - Using common formats and libraries as much as possible
- Execution as part of regression has to be possible in different environments
  - Integration servers often only run in one particular integration environment
Test Case Implementation by Unified Scenarios (1/2)

- Example of model types for host representation:
  - Windows on Simics / SVOS on Simics
  - emuCPU
  - VboxWithUbuntu

- OS/driver layer with common minimum functionality:
  - Support for physical memory access and ISR installation
  - Test cases run in User Space
Test Case Implementation by Unified Scenarios (2/2)

- FW code for test scenarios with hardware abstraction layer (HAL)
  - Interrupt installation, buffer allocation, device mapping into memory space (PNP), virtual-physical address mapping, etc.
  - Common file formats for any IO

- Test case description/execution:
  - XML format for test case description
  - Python parser used to derive build and run scripts for different integration systems
Use Cases as Driver of Test Case Development (1/2)

- Use cases are project specific
  - N-1 FW as basis for next project
    - Which components are planned to change during the project
    - Negotiation between groups about necessary features
  - High-Level Architectural Specification (HAS) identifies functionality that has to be supported
    - Description can be insufficient
    - More formal descriptions are needed
- Existing synthetic tests can become basis of use case definition
- Traces from existing models
Use Cases as Driver of Test Case Development (2/2)

- There has to be a clear understanding about the capabilities of the VP between user and developer
  - Aligning development plans around clearly specified use cases
- Use cases can and should evolve during the project duration
  - VP is software and can be updated relatively easily
CONCLUSIONS
Conclusions

- Various teams actively use VPs:
  - FW development $\rightarrow$ Early system integration
  - SoC integration $\rightarrow$ Validation of interaction with OS functionality
  - Full platform VP $\rightarrow$ System-level validation, debug and development

- A variety of integration systems is required
- Use cases align different integration systems
- VPs are an important strategy to parallelize FW and SW development with HW development
Next Steps

- Approach is currently applied by several IP and system level integration teams and found beneficial.
- We will extend it to more IP teams and also will address joint IP scenarios.
Acknowledgements

• Co-Authors
  – Grit Christange
  – Daniel Aarno
  – Josef Eckmueller
Questions