Virtual Prototyping

Alexander Fasching

Vienna University of Technology
Matr. Nr.: 1225900
Email: alexander.fasching@student.tuwien.ac.at

HW/SW Codesign WS 2016/17
Overview

- What is Virtual Prototyping?
- Why are they used?
- Advantages compared to physical prototypes
- Abstraction levels
- Determinism and system state
- Tools, limits and requirements
What is virtual prototyping?

A virtual prototype is an executable software model of a hardware system that runs on a host computer.

Possible constraints on the definition:

- Binary compatible to physical hardware (instruction set, memory map, registers)
- Runs the full software stack, from boot-code to operating system and application
- Provides the same debugging utilities as the physical hardware (debugger, profiler, . . .)
- Ideally, testcases for functional verification can be executed on the virtual and the physical prototype.
Why the effort?

Usual development cycle:

- 12-15 months hardware development
- 9-12 months software development
- 6-9 months system integration, verification and validation
- Grand total of 36 months
Why the effort?

Usual development cycle:
- 12-15 months hardware development
- 9-12 months software development
- 6-9 months system integration, verification and validation
- Grand total of 36 months

Desired product window usually 24 months or less
The solution

- Develop HW and SW in parallel
- Verified software reduces integration time
Advantages to physical prototypes

- Shifts the start of software development "left": Development can start well before a physical prototype is available, even before RTL design is finished.
- Deterministic (e.g. simulate user input)
- Additional debugging and monitoring capabilities (fault injection, tracing, profiling)
- Verified firmware available before physical prototype reduces hardware bringup time
- Test special conditions without damaging hardware (short/open connections, memory corruption)
Abstraction

- Tradeoff between accuracy and simulation speed
- Consistency between VP and hardware
- Using formal specification as *golden reference*
- Proving equivalency of specification, VP and hardware
- Cycle-accurate
- Bit-exact
- Speed: 1/100000 realtime
- Interactive debugging possible, but very slow and not widely supported
TLM - Instruction set of target

- Firmware executed through instruction-set-simulator (ISS)
- Bit-exact
- Cycle-approximate
- Speed: 1/100 realtime
Source-level/host-compiled simulation

- Firmware and processor model executed as native binary on host
- Timing approximation through static analysis of source code and binary
- Original source code annotated with delays
- Speed: $\approx$ realtime
Source-level simulation stack
(Non)Determinism and system state

- Clock A and B asynchronous
- Timing between commands on Debug A and B unknown
(Non)Determinism and system state

- Subsystem B can’t be stopped (might be analog)
- Timeouts might occur
(Non)Determinism and system state

- Full control over clocks
- Synchronous debug commands to all subsystems
- Global time
- Global event ordering
Generic test-environment

- Same testsuite for physical and virtual prototype
- Abstraction layer for debugging tools and testsuite
- Interface to signal generators, digital I/O and measurement equipment, both physical and virtual
Drawbacks and limits

- Changes in development flow
- Additional costs
  - Tools
  - Developers
  - Computational resources
- Incompatible interfaces between different tools
- High effort to integrate models of different vendors
Available tools

- **Free**
  - SystemC
  - Qemu (allows the creation of virtual devices)
  - Android SDK

- **Commercial**
  - Matlab/Simulink for high level models
  - Synopsys: Virtualizer (ARM, Automotive and ARC HS38)
  - Mentor: Vista Virtual Prototyping (ARM)
  - Cadence: Virtual System Platform

T. De Schutter.
*Better Software. Faster!*
Synopsys, 2014.

W. Ecker, S. Heinen, and M. Velten.
Using a dataflow abstracted virtual prototype for HdS-design.

A. Garg.
Fast Virtual Prototyping for early software design and verification.
G. Hellestrand.  
How virtual prototypes aid SoC hardware design.  
http://www.embedded.com/design/virtual-prototyping/4024966/

D. Koivisto.  
Exploring Multicore Power Management with Modeling and Simulation.  
J. Lipman.
Taking System Design to a Higher Level.
http://www.embedded.com/design/programming-languages-and-tools/4018351/

E. Lumpkin and C. Alford.
How to make virtual prototyping better than designing with hardware: Part 1.
http://www.embedded.com/design/prototyping-and-development/4200573/
E. Lumpkin and C. Alford.
How to make virtual prototyping better than designing with hardware: Part 2.
http://www.embedded.com/design/real-time-and-performance/4200574/

S. Matalon.
Vista Virtual Prototyping.
B. Neifert.
Leveraging virtual hardware platforms for embedded software validation.
http://www.embedded.com/design/prototyping-and-development/4026870/
Leveraging-virtual-hardware-platforms-for-embedded-software-validation
2008.