OVPsim Specification

Imperas Software Limited
Imperas Buildings, North Weston,
Thame, Oxfordshire, OX9 2HA, UK
docs@imperas.com

Author:	Imperas Software Limited
Version:	1.02
Filename:	OVP_OVPsim_Specification.doc
Project:	OVP OVPsim Specification
Last Saved:	Wednesday, 27 May 2020
Keywords:	

Table of Contents

Specification of OVPsim™

1 Overview

Open Virtual Platforms™ (OVP™) is the combination of the OVP standard APIs that enable a C/C++ model to be written, a collection of free open source processor and peripheral models, and OVPsim a fast, easy to download and use simulator that executes these models.

Imperas™ makes models of processors, platforms and peripherals available as freely available open source from the Open Virtual Platforms (www.OVPworld.org) web portal.

OVPsim is developed and supported by Imperas Software Ltd, and is provided in closed source binary form. OVPsim is available for download from the OVP portal and is free for non-commercial usage and licenses for commercial usage are obtained from Imperas.

Imperas also provides commercial licenses to higher performance multicore simulation, verification, debug, and analysis tools that are based on OVP and that are upwards compatible with OVPsim.

Imperas also engages in commercial relationships with other suppliers for them to provide 'OEM' versions of OVPsim that fits well with their market requirements and customer use model. These OEM versions will have restrictions in terms of functionality, capability and performance. Please refer to your supplier or Imperas for more information.

2 Supported Host Platforms

The OVPsim simulator is available on X86 PC Windows 7 SP1 and Linux
Fedora Core 12.

3 Delivery Package

OVPsim is provided as part of a package delivered as self installers for Windows and Linux that require click through license acceptance. The OVP package includes the OVPsim simulator provided as a binary shared object runtime, the user documentation in pdf format, the API documentation as html using Doxygen format, the API headers, and several examples in source form with detailed walkthroughs.

The OVP portal makes available many demonstration packages that include full virtual platforms for ARM and MIPS processors that run full operating systems such as Linux and Nucleus. These may include the source of models and binaries of the operating systems.

OVP open source models are available from the OVP portal. The model and platform sources use an Apache / Modified Apache 2.0 open source license.

The current version and previous versions of OVP and OVPsim are available from the OVP portal.

Releases take place every 60-90 days and there is no guarantee that components of the current release are compatible with components of previous releases. The recommendation is to always upgrade all components to the same release version.

Backwards compatibility of OVP APIs will be maintained, as far as possible. A model written for an older version of an OVP API can be re-compiled and used with the current version.

4 Standalone Usage (with C/C++ platforms)

To use OVPsim a user requires a host C/C++ compiler to compile a platform and a cross compiler to compile the target processors' application binary. A platform uses binary models of processors and peripherals.

Peripherals are written in C/C++ that make calls to OVP APIs and are compiled using the OVP Peripheral Simulation Engine compiler tool chain made available from the OVP portal.

Processors are written in C/C++ that make calls to OVP APIs and are compiled using a standard GNU GCC host compiler.

A platform is written in C/C++ that makes calls to the OVP APIs and is compiled using a standard GNU GCC host compiler. The platform calls the OVP simulation scheduler or can implement its own scheduling control over the platform components.

The resultant executable dynamically loads the OVPsim shared binary object and any processors and peripheral model shared object binaries, and then simulation proceeds with the processor model fetch and execute of target processor instructions.

5 Using OVPsim™ and OVP™ models with SystemC / TLM2.0

OVP models are used by platforms as shared objects dynamically loaded at simulation time by the OVPsim runtime. The models have SystemC/TLM2.0 interfaces and can be instanced/used in platforms written in SystemC TLM2.0. The SystemC simulator will be the main simulator and the OVP models will be executed as and when required by the platform.

The OVP processor models are optimized for best performance by making use of a negotiated DMI if the memory in the platform supports it.

Some other companies have solutions where you can 'co-simulate' a subsystem of the virtual platform in their simulator and rest in the SystemC simulator. Often this allows two partitions: theirs, and SystemC. This requires two islands of functionality that co-simulate. This is normally OK with one processor on one side, and the rest of the system in the other. However if you want several processors intermixed with bus logic and BeHavioral components it is very hard to split this into two.

The OVP approach is that if you want to use SystemC/TLM2.0 then that is what you use for your platform – and any OVP processors, peripherals, or BeHavioral components can be instanced within your SystemC/TLM2.0 platform – with no complex, or inefficient co-simulation

OVP works very well with SystemC / TLM2.0 platforms and simulators. OVPsim has been tested with Cadence, Mentor, and Synopsys SystemC simulation environments.

6 Using OVP™ with Verilog

OVP models can be interfaced to Verilog simulators by wrapping them in PLI C code and instancing that in the normal Verilog way of instancing C models. The platform is Verilog with calls into the PLI to activate the OVP models. Alternatively now that most HDL simulators can work with SystemC it is possible to instance OVP models as SystemC components with the appropriate additions of bus transactors / interfaces.

7 Debug interface

OVPsim allows one of the processors being simulated to connect via a port using RSP to a GNU GDB Debugger.

The single processor RSP GDB interface can be used when OVPsim is used standalone, as part of a SystemC TLM2.0 simulation or as part of a Verilog simulation.

8 Performance

OVPsim processor models run many 100's of millions of instructions per second on standard benchmarks. Max speed on a 3GHz single core x86 has been recorded above 4,000 mips for a 32bit RISC processor. With full virtual memory, many peripherals and a full operating system running then 100-500 mips is typical performance for a single core simulation. Attaching a debugger or lots of I/O, tracing etc, will slow this significantly.

Up to date performance data is available at the web site portal: www.ovpworld.org

9 Licensing

OVP requires user registration on the OVP portal before downloads can be made.

Each install requires a click through confirmation that the End User License Agreement is acceptable.

OVPsim usage is controlled by FlexLM licensing using an Imperas Daemon and node locked and floating licenses are available. A 90 day evaluation license is available automatically from the OVP portal and other licenses are available from Imperas.

Open source models are under Apache 2.0 / Modified Apache 2.0, or LGPL and some models are provided as binaries with FlexLM licensing.

10 Simulation Specifics

OVPsim executes 100,000 instructions per quanta by default[1]. The number of instructions per quanta can be set by the user in the platform using OVP API calls, enabling simulation of Coarse-Grain Multi-threading and Fine-Grained Multi-threading.

With instruction accurate processor models OVPsim is architecturally accurate enough to pass IP providers Architectural Verification Programs.

11 Modeling Specifics

IEEE-compliant FPU Behavior can be simulated, if modeled.

SIMD instructions such as those in the MIPS DSP ASE, or ARM NEON can be simulated, if modeled. SIMD instructions such as those in the FPU Dual-Single format can be simulated, if modeled.

Implementation-specific software visible registers can be simulated. For example, any registers related to branch-prediction/jump-return-stacks can be included in a model.

External events such as interrupts can be delivered to the processor models if the C, C++, SystemC, TLM2.0 platform uses the model in this way.

Platforms can be built that allow multiple processor cores to be simulated per simulation.

Processors can be of different instruction sets and configurations.

Users have the ability, through the platform API, to model software-managed memories in addition to regular (‘flat’) system memory. OVPsim components can be mapped to software managed memory spaces.

Components may signal asynchronous events, such as interrupts, via connection to other platform components such as an interrupt controller.

12 Restrictions

OVPsim does not implement all the API calls as defined in the OVP API documentation. OVPsim does not implement and allow all the functionality of the intercept technology. OVPsim does not allow heterogeneous processor platforms i.e. the use of processors from different vendors but does allow multiple different types of processors from a single vendor to be used.

The Imperas Professional simulator removes these restrictions.

[1] The default Quantum for a processor having the default MIPS rate of 100 MIPS is also equivalent to 1mS of simulation time

13 More information on OVP™ and OVPsim™

The OVP website provides more information about OVP and OVPsim.

Visit www.OVPworld.org and click on the appropriate title bar for information on:

OVP Technology:	Technology
OVPsim:	Technology → OVPsim
OVP APIs:	Technology → OVP APIs
OVP Models:	Technology → OVP Models
OVP and SystemC:	Technology → SystemC
OVP and SystemC TLM2.0:	Technology → TLM2.0
OVP and SPIRIT/IP-XACT:	Technology → Spirit

#