Imperas Guide to using Virtual Platforms

Platform Specific Information for arc.ovpworld.org / BareMetalArcSingle_TLM2.0

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

Author	Imperas Software Limited
Version	20150901.0
Filename	Imperas_Platform_User_Guide_BareMetalArcSingle_TLM2.0.pdf
Created	26 August 2015
Status	OVP Standard Release

Copyright Notice

Right to Copy Documentation

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the readers responsibility to determine the applicable regulations and to comply with them.

Disclaimer

IMPERAS SOFTWARE LIMITED, AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Model Release Status

This model is released as part of OVP releases and is included in OVPworld packages. Please visit OVPworld.org.

Table Of Contents

1.0 Virtual Platform: BareMetalArcSingle_TLM2.0

This document provides the details of the usage of an Imperas OVP Virtual Platform. The first half of the document covers specifics of this particular virtual platform. For more information about Imperas OVP virtual platforms, how they are built and used, please see the later sections in this document.

1.1 Description

Bare Metal Platform for an ARC Processor.
 The bare metal platform instantiates a single ARC processor instance.
 The processor operates using little endian data ordering.
 It creates contiguous memory from 0x00000000 to 0xFFFFFFFF.
 The TLM2.0 platform can be passed any application compiled to an ARC elf format.
 ./platform.exe application.elf

1.2 Licensing

Open Source Apache 2.0

1.3 Limitations

BareMetal platform for execution of ARC binary files compiled with FOSS for Synopsys DesignWare ARC Processors CrossCompiler toolchain.

1.4 Location

The BareMetalArcSingle_TLM2.0 virtual platform is located in an Imperas/OVP installation at the VLNV: arc.ovpworld.org / platform / BareMetalArcSingle_TLM2.0 / 1.0.

2.0 Processor [arc.ovpworld.org/processor/arc/1.0] instance: cpu1

2.1 Processor model type: 'arc' variant '700' definition

Imperas OVP processor models support multiple variants and details of the variants implemented in this model can be found in:
- the Imperas installation located at ImperasLib/source/arc.ovpworld.org/processor/arc/1.0/doc
- the OVP website: OVP_Model_Specific_Information_arc_700.pdf

2.1.1 Description

ARC 700 processor model (ARCv1 architecture)

2.1.2 Licensing

Usage of binary model under license governing simulator usage. Source of model available under Imperas Software License Agreement.

2.1.3 Limitations

Instruction pipelines are not modeled in any way. All instructions are assumed to complete immediately.

Instruction and data caches are not modeled, except for the auxiliary register interface.
External host debug is not modeled, except for the auxiliary register interface.
Real-world timing effects are not modeled. All instructions are assumed to complete in a single cycle.
User extensions are not yet implemented, except for extension core registers.

2.1.4 Verification

Models have been validated correct in a cooperative project between Imperas and ARC

2.1.5 Reference

ARC Processor ARC6xx/ARC7xx Reference Documentation

2.1.6 Debugging

The model has been designed for debug using GNU gdb ARCompact/ARCv2 ISA elf32 version 7.5.1. To ensure correct behavior, enter the following command into gdb before attempting to connect to the processor:
set architecture ARC700
Failure to do this may cause the debugging session to fail because of g-packet size mismatch.

2.1.7 Features

The model implements the full ARCv1 instruction set.
The exact set of core instructions present can be configured by a number of parameters: see information for opt-swap, opt-bitscan, opt-extended-arith and opt-multiply in the table below.
Timer 0 and Timer 1 can be enabled using parameters opt-timer0 and opt-timer1, respectively.
The sizes of DCCM, ICCM0 and ICCM1 can be specified using parameters opt-dccm-size, opt-iccm0-size and opt-iccm1-size, respectively. Reset base addresses for the ICCMs can be specified using opt-iccm0-base and opt-iccm1-base. Note that the DCCM reset base address is architecturally defined (0x80000000) and not configurable. When CCMs are present, bus ports called DCCM0, ICCM0 and ICCM1 are created so that CCM contents may be viewed or modified externally by connecting to these ports. Parameter opt-internal-ccms specifies whether CCM memory is modeled internally or externally. If modeled externally, the CCMs must be implemented on a bus which is then connected to the CCM bus ports listed above (this parameter is ignored if CCM ports are unconnected; in that case, CCMs are always modeled internally). Parameter opt-reset-internal-ccms indicates that internally-modeled CCMs should be cleared to zero on a processor reset; if False, then internally-modeled CCMs retain their previous state after a reset.
The set of core registers can be specified using parameter opt-extension-core-regs. This is a 64-bit value in which a 1-bit implies the presence of that core extension register. For example, a value of 0xf00000000ULL implies that extension registers r32-r35 should be configured.
The reset value of the exception vector base register can be specified using parameter opt-intvbase-preset.

2.2 Instance Parameters

Several parameters can be specified when a processor is instanced in a platform. For this processor instance 'cpu1' it has been instanced with the following parameters:

Table 1. Processor Instance 'cpu1' Parameters (Configurations)

Parameter	Value	Description
endian	big	Select processor endian (big or little)
mips	100	The nominal MIPS for the processor
variant	700	The processor variant

2.3 Memory Map for processor 'cpu1' bus: 'bus1'

Processor instance 'cpu1' is connected to bus 'bus1' using master port 'INSTRUCTION'.
Processor instance 'cpu1' is connected to bus 'bus1' using master port 'DATA'.

Table 2. Memory Map ('cpu1' / 'bus1' [width: 32])

Lo Address	Hi Address	Instance	Component
0x0	0x1FFFFF	ram1	ram
0x3D000000	0x3DFFFFFF	ram2	ram

2.4 Net Connections to processor: 'cpu1'

There are no nets connected to this processor.

3.0 Overview of Imperas OVP Virtual Platforms

This document provides the details of the usage of an Imperas OVP Virtual Platform. The first half of the document covers specifics of this particular virtual platform.

This second part of the document, includes information about Imperas OVP virtual platforms, how they are built and used.

The Imperas virtual platforms are designed to provide a base for you to run high-speed software simulations of CPU-based SoCs and platforms on any suitable PC. They are typically based on the functionality of vendors fixed or evaluation platforms, enabling you to simulate software on these reference platforms. Typically virtual platforms are fixed and require the vendor to modify or extend them. Imperas virtual platforms are different in that they enable you to extend the functionality of the virtual platform, to closer reflect your own platform, by adding more component models, running different operating systems or adding additional applications.

Imperas virtual platforms are created using the Imperas iGen technology, allowing them to be used with Imperas OVP based simulators and also with Accellera/OSCI compliant SystemC simulators and commercial EDA System Design environments that use SystemC.

Virtual platforms include simulation models of the target devices, including the processor model(s) for the target device plus enough peripheral models to boot an operating system or run bare metal applications. The platform and the peripheral models used in most of the virtual platforms are open source, so that you can easily add new models to the platform as well as modify the existing models. Some models are only provided as binary, normally because the IP owner has restricted the release of the model source. In this case, please contact Imperas for more information.

There are typically several generic flavors of the virtual platforms for specific processor families, some targeting full operating systems, such as Linux, and some which focus on Real Time Operating Systems (RTOS) such as Mentor Nucleus or freeRTOS. OVP models of the processor cores are included in the virtual platforms, and for those processors which support mulitple cores SMP Linux is often supported for that virtual platform. For all of these virtual platforms, many of the peripheral components of the platform are modeled, often including the Ethernet and USB components. The semi-hosting capability of the Imperas virtual platform simulator products enables connection via the Ethernet and USB components from the virtual platform to the real world via the x86 host machine.

The Imperas OVP CPU models are written using the OVP Virtual Machine Interface (VMI) API that defines the behavior of the processor. The VMI API makes a clear line between model and simulator allowing very good optimization and world class high speed performance. The processor models are Instruction Accurate and do not model the detailed cycle timing of the processor and they implement functionality at the level of a Programmers View of the processor and peripherals and the software running on them does not know it is not running on hardware. Many models are provided as a binary shared object and also as source. This allows the download and use of the model binary or the use of the source to explore and modify the model. The models are run through an extensive QA and regression testing process and most processor model families are validated using technology provided by the processor IP owners. All the models in this platform are developed with the Open Virtual Platforms APIs and are implemented in C.

More information on modeling and APIs can be found on the www.OVPworld.org site.

4.0 Getting Started with Imperas OVP Virtual Platforms

Virtual platforms are downloadable from the OVPworld website OVPworld.org/downloads. You need to browse and look for '<platform processor name> Examples'. You do need to be registered and logged in on the OVP site to download. OVPworld currently provides 32 bit host versions of packages containing virtual platforms.

When downloading, choose, Linux or Windows host. 32 bit packages can be installed and executed on 32 bit or 64 bit hosts. If you require a 64 bit host version please contact Imperas.

For example, for the ARM Versatile Express platform booting Linux on Cortex-A15MP Single, Dual, and Quad core procesors, you would want the download package: 'OVPsim_demo_Linux_ArmVersatileExpress_arm_Cortex-A15MP'.

Most virtual platform packages contain the platform and all the processor and peripheral models needed. You will need to download a simulator to run the platform. You can use OVPsim, downloadable from OVPworld.org/downloads, or you can use one of the Imperas simulators (imperas.com/products) available commercially from Imperas.

5.0 Simulating Software

5.1 Getting a license key to run

After you have downloaded you will need a runtime license key before the simulators will run. For OVPsim please visit OVPworld.org/likey and provide the required information and an evaluation/demo license key will be automatically sent to you. If you are using Imperas, then please contact Imperas for a license key.

5.2 Normal runs

To run a platform, read the section below on command line control of the platform and the section on setting command line arguments.

5.3 Loading Software

For most virtual platforms the platform is already configured to run the default software application/program and there is normally a script to run that sets some arguments. You can then copy/edit this script to select your own applications etc.

The example application programs are typically .elf format files and are provided pre-compiled. There are normally makefiles and associated scripts to recompile the example applications.

To find more information about compiling and loading software, the following document should be looked at: Imperas_Installation_and_Getting_Started.pdf.

5.4 Semihosting

In a virtual platform, semihosting is not normally used as there is normally hardware that implements the appropriate functionality - for example I/O will be handled by UARTs etc.

5.5 Using a terminal (UART)

If the platform includes one or more UARTs you will need to connect a terminal program to it so that you can see output and type into the simulated program. Review the list of peripherals below and see what configuration options it has been set with. In most cases there is an option to set to instruct the simulator to 'pop up' a terminal window connected to the simulated UART.

5.6 Interacting with the simulation (keyboard and mouse)

If the platform has a simulated UART you can normally set a command to get the simulator to pop up a terminal window allowing you to see output from the simulated UART and also allowing you to type characters into the UART that can be processed by the simulated software.

If your simulated platform has an LCD device then you can often configure it to recognize mouse movements and mouse clicks - allowing full interaction.

To see these interactions in action, have a look at some of the available videos available at OVPworld.org/demosandvideos.

5.7 More Information (Documentation) on Simulation

To find more information about running simulations and more of the options the simulators provide, the following documents should be looked at:
Imperas_Installation_and_Getting_Started.pdf
OVPsim_and_CpuManager_User_Guide.pdf
OVP_Control_File_User_Guide.pdf

A full list of the currently available OVP documentation is available: OVPworld.org/documentation.

6.0 Debugging Software running on an Imperas OVP Virtual Platform

The Imperas and OVP simulators have several different interfaces to debuggers. These include several proprietary formats and also the standard GNU RSP format is supported allowing many compatible debuggers to be used. Below are some examples that Imperas directly support.

6.1 Debugging with GDB

A GNU debugger (GDB) can be connected to a processor in a platform using the RSP protocol. This allows the application program running on a processor to be debugged using a specific GDB for the processor selected. When using the Imperas Professional products many connections can be made allowing a GDB to be connected to all the processors in the platform.

The use of GDB is documented: OVPsim_Debugging_Applications_with_GDB_User_Guide.pdf.

*6.2 Debugging with Imperas M*DBG*

The Imperas multi-processor debugger can be connected to a platform and through this connection you can debug application programs running on all of the processors instanced within the platform. It is also capable, within this single unified environment, to debug peripheral model behavioral code in conjunction with the processor application programs.

For more information please see the Imperas M*DBG user guide.

The Imperas multi-processor debugger is also capable of controlling the Imperas Verification Analysis aand Profiling (VAP) tools in real time, making them invaluable to application program development, debugging and analysis.

For more information please see the Imperas VAP tools user guide.

6.3 Debugging with the Imperas iGui and GDB

Imperas iGui gives a GUI front end to the use of the GDB debugger. It allows use of all the features of GDB including source level application program debugging on processors.

*6.4 Debugging with the Imperas iGui and M*DBG*

Imperas iGui gives a GUI front end to the Imperas multi-processor debugger. It provides all the features of this debugger but does so with source level application program debugging on processors and source level debugging of the behavioral code on peripheral components in the platform. A context view shows all the processor and peripheral components within the platform and allows switching between them to examine the state of each at the event at which the simulation was stopped

Imperas iGui provides a menu from which the Imperas VAP tools can be controlled.

6.5 Debugging with Eclipse

A standard Eclipse CDT development environment can be connected to one or more processors in a platform (multiple processors require an Imperas professional product). The simulation platform is started remotely or using the external tool feature in Eclipse, opens a debug port and awaits the connection with Eclipse. All features provided by the Eclipse CDT development environment are available to be used to debug software applications executing on the processors in the platform.

The use of Eclipse is documented: OVPsim_Debugging_Applications_with_Eclipse_User_Guide.pdf.

6.6 Debugging applications running under a simulated operating system

If the simulated platform is running an Operating System and the platform has a UART or Ethernet etc connection then it is often possible to connect an external debugger and debug the applications running under the simulated operating system.

An example would be a simulated platform running the Linux operating system, such as the MIPS Malta, or ARM Versatile Express. Within the simulated Linux you can start a gdbserver that connects from within the simulation through a UART out to the host PC via a port. Within the host PC you start a terminal program and connect to the port with a debugger such as GDB and can then debug the simulated user application.

7.0 Modifying the Platform

7.1 Platforms use C/C++ and OVP APIs

The Imperas and OVP simulators execute a platform that is written in C/C++ and that makes function calls into the simulator's APIs. Thus the virtual platform is compiled from C/C++ into a binary shared object that the simulator loads and runs. OVP provides the definition and documentation that defines the C APIs for modeling the platforms, the peripherals and the processors. You can find more information about these APIs on the OVP website and in the OVP API documentation.

7.2 Platforms/Peripherals can be easily built with iGen from Imperas

Imperas provides a product 'iGen' that takes an input script file and creates the C/C++ files needed for platforms and peripherals - it creates the C/C++ file that is compiled into the platform or peripheral that is needed as an object file by the simulator. iGen creates the C/C++ files, you then need to add any necessary behaviors or further details etc. For platforms iGen creates either a C platform or a C++ SystemC TLM2 platform. For peripherals iGen creates the C files and also provides a native C++ SystemC TLM2 interface to allow the peripheral to be instantiated in SystemC TLM2 platforms.

Information on iGen is available from: imperas.com/products.

7.3 Re-configuring the platform

There will nornmally be several configuration options that you can set when running the platform without the neeed to change any source. Refer to the section above on command line arguments. If these do not allow you to make the changes you need, then you may need to edit and recompile the source of the platform.

The source of the platform and the source of the peripherals will be installed as part of the packages you are using. The sources are located in the Imperas/OVP installation VLNV source tree. The VLNV term refers to: Vendor (eg arm.ovpworld.org), Library (eg platform), Name, (eg ArmVersatileExpress-CA15), and Version (eg 1.0). To modify the platform, locate the platform source files.

If you are an Imperas user and have access to iGen, we recommend you modify the source script files and regenerate and recompile the C that makes up the platform. Refer to the Imperas iGen model generator guide and the Imperas platform generator guide.

If you are using the C or SystemC TLM2 platforms with OVPsim, then you can edit the C/C++ files, recompile the source directly using the supplied makefiles, and the run the simulator directly with the resultant shared object.

7.4 Replacing peripherals components

If you need to replace peripherals, find the appropriate place in the source of the platform, make the change you need, and recompile etc. Look in the library for documentation on available peripherals and their configuration options.

7.5 Adding new peripherals components

If you need to add peripherals, find the appropriate place in the source, make the additions you need, and recompile etc. Look in the library for documentation on available peripherals and their configuration options.

If you need to create new peripheral components then use iGen to very quickly create the necessary C/C++ files that get you started. With iGen you can create peripherals with register/memory state in a few lines of iGen source. When adding behavior to the peripherals refer to the OVP API documentation.

8.0 Available Virtual Platforms

Table 3. Imperas / OVP Extendable Platform Kits (17 available)

Platform Name	Vendor
AlteraCycloneIII_3c120	altera.ovpworld.org
AlteraCycloneV_HPS	altera.ovpworld.org
AlteraCycloneV_HPS_TLM2	altera.ovpworld.org
ARMv8-A-FMv1	arm.ovpworld.org
ArmIntegratorCP	arm.ovpworld.org
ArmIntegratorCP_TLM2.0	arm.ovpworld.org
ArmVersatileExpress	arm.ovpworld.org
ArmVersatileExpress-CA15	arm.ovpworld.org
ArmVersatileExpress-CA9	arm.ovpworld.org
ArmVersatileExpress_CA9_TLM2	arm.ovpworld.org
AtmelAT91SAM7	atmel.ovpworld.org
FreescaleKinetis60	freescale.ovpworld.org
FreescaleVybridVF5	freescale.ovpworld.org
MipsMalta	mips.ovpworld.org
MipsMaltaLinux_TLM2.0	mips.ovpworld.org
RenesasUPD70F3441	renesas.ovpworld.org
XilinxML505	xilinx.ovpworld.org

Table 4. Imperas General Virtual Platforms (6 available)

Platform Name	Vendor
arm-ti-eabi	arm.imperas.com
armm-ti-coff	arm.imperas.com
armm-ti-eabi	arm.imperas.com
HeteroAlteraCycloneV_HPS_CycloneIII_3c120	imperas.ovpworld.org
HeteroArmNucleusMIPSLinux	imperas.ovpworld.org
QuadArmVersatileExpress	imperas.ovpworld.org

Table 5. Imperas / OVP Bare Metal Virtual Platforms (39 available)

Platform Name	Vendor
BareMetalNios_IISingle	altera.ovpworld.org
BareMetalNios_IISingle_TLM2.0	altera.ovpworld.org
BareMetalArcSingle	arc.ovpworld.org
BareMetalArcSingle_TLM2.0	arc.ovpworld.org
BareMetalArm7Single	arm.ovpworld.org
BareMetalArm7Single_TLM2.0	arm.ovpworld.org
BareMetalArmAArch64Single_TLM2.0	arm.ovpworld.org
BareMetalArmCortexADual	arm.ovpworld.org
BareMetalArmCortexASingle	arm.ovpworld.org
BareMetalArmCortexASingleAngelTrap	arm.ovpworld.org
BareMetalArmCortexASingle_TLM2.0	arm.ovpworld.org
BareMetalArmCortexMSingle	arm.ovpworld.org
BareMetalArmCortexMSingle_TLM2.0	arm.ovpworld.org

ArmCortexMFreeRTOS	imperas.ovpworld.org
ArmCortexMuCOS-II	imperas.ovpworld.org
BareMetalArcManycore24_TLM2.0	imperas.ovpworld.org
BareMetalArm7Dual_TLM2.0	imperas.ovpworld.org
BareMetalArmx1Mips32x3	imperas.ovpworld.org
BareMetalMips32Multicore2_TLM2.0	imperas.ovpworld.org
Or1kUclinux_TLM2.0	imperas.ovpworld.org
BareMetalM14KSingle	mips.ovpworld.org
BareMetalM14KSingle_TLM2.0	mips.ovpworld.org
BareMetalMips32Dual	mips.ovpworld.org
BareMetalMips32Single	mips.ovpworld.org
BareMetalMips32Single_TLM2.0	mips.ovpworld.org
BareMetalMips64Single	mips.ovpworld.org
BareMetalMips64Single_TLM2.0	mips.ovpworld.org
BareMetalMipsDual	mips.ovpworld.org
BareMetalMipsSingle	mips.ovpworld.org
BareMetalMipsSingle_TLM2.0	mips.ovpworld.org
BareMetalOr1kSingle	ovpworld.org
BareMetalOr1kSingle_TLM2.0	ovpworld.org
BareMetalM16cSingle	posedgesoft.ovpworld.org
BareMetalPowerPc32Single	power.ovpworld.org
BareMetalPowerPc32Single_TLM2.0	power.ovpworld.org
BareMetalV850Single	renesas.ovpworld.org
BareMetalV850Single_TLM2.0	renesas.ovpworld.org
ghs-multi	renesas.ovpworld.org
BareMetalMicroBlazeSingle_TLM2.0	xilinx.ovpworld.org

#