Imperas Guide to using Virtual Platforms

Platform / Module Specific Information for imperas.ovpworld.org / HeteroArmNucleusMIPSLinux

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

Author	Imperas Software Limited
Version	20211118.0
Filename	Imperas_Platform_User_Guide_HeteroArmNucleusMIPSLinux.pdf
Created	31 December 2021
Status	OVP Standard Release

Copyright Notice

Right to Copy Documentation

Destination Control Statement

Disclaimer

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 Platform / Module: HeteroArmNucleusMIPSLinux

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

1.1 Virtual Platform / Module Type

Hardware described using OVP can either be a platform, module, processor, or peripheral.
This is a fixed virtual platform.

1.2 Licensing

Open Source Apache 2.0

1.3 Description

This heterogeneous platform combines MIPS Malta and ARM Integrator platforms.

MIPS Malta executes a Linux Kernel Operating System.

ARM Integrator executes the Nucleus Operating System.

1.4 Limitations

Peripherals are modeled only to the extent required to boot and run Operating Systems.

1.5 Reference

OVP Heterogeneous platforms. MIPS Malta and ARM IntgratorCP reference guides.

1.6 Location

The HeteroArmNucleusMIPSLinux virtual platform / module is located in an Imperas/OVP installation at the VLNV: imperas.ovpworld.org / platform / HeteroArmNucleusMIPSLinux / 1.0.

2.0 Command Line Control of the Platform

2.1 Built-in Arguments

Table 1. Platform Built-in Arguments

Attribute	Value	Description
allargs	allargs	The Command line parser will accept the complete imperas argument set. Note that this option is ignored in some Imperas products

When running a platform in a Windows or Linux shell several command arguments can be specified. Typically there is a '-help' command which lists the commands available in the platforms. For example:
myplatform.exe -help

Some command line arguments require a value to be provided. For example:
myplatform.exe -program myimagefile.elf

2.2 Platform Specific Command Line Arguments

Table 2. Platform Command Line Arguments

Name	Type	Description
uartconsole	boolvar	open a console terminal on the ARM IntegratorCP UART
uartport	uns32var	set the base port number to open on the ARM IntegratorCP UART
nographics	boolvar	disable the MIPS Malta graphics window

3.0 Processor [mips.ovpworld.org/processor/mips32_r1r5/1.0] instance: mipsle1

3.1 Processor model type: 'mips32_r1r5' variant '34Kf' 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/mips.ovpworld.org/processor/mips32_r1r5/1.0/doc
- the OVP website: OVP_Model_Specific_Information_mips32_r1r5_34Kf.pdf

3.1.1 Description

MIPS32 Configurable Processor Model

3.1.2 Licensing

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

3.1.3 Limitations

If this model is not part of your installation, then it is available for download from www.OVPworld.org/ip-vendor-mips.

3.1.4 Verification

Models have been validated correct as part of the MIPS Verified program and run through the MIPS AVP test programs

3.1.5 Features

only MIPS32 Instruction set implemented
MMU Type: Standard TLB
FPU implemented
L1 I and D cache model in either full or tag-only mode implemented (disabled by default)
Vectored interrupts implemented
MIPS16e ASE implemented
MT ASE implemented

DSP ASE implemented

3.2 Instance Parameters

Several parameters can be specified when a processor is instanced in a platform. For this processor instance 'mipsle1' it has been instanced with the following parameters:

Table 3. Processor Instance 'mipsle1' Parameters (Configurations)

Parameter	Value	Description
endian	little	Select processor endian (big or little)
simulateexceptions	simulateexceptions	Causes the processor simulate exceptions instead of halting
mips	100	The nominal MIPS for the processor
imagefile	mips.vmlinux	The image file to load onto the processor

Table 4. Processor Instance 'mipsle1' Parameters (Attributes)

Parameter Name	Value	Type
variant	34Kf	enum
vectoredinterrupt	0	bool
config1MMUSizeM1	63	uns32

3.3 Memory Map for processor 'mipsle1' bus: 'busmipsMain'

Processor instance 'mipsle1' is connected to bus 'busmipsMain' using master port 'INSTRUCTION'.
Processor instance 'mipsle1' is connected to bus 'busmipsMain' using master port 'DATA'.

Table 5. Memory Map ('mipsle1' / 'busmipsMain' [width: 32])

Lo Address	Hi Address	Instance	Component
0x0	0x7FFFFFF	Core_Board_SDRAM	ram
remappable	remappable	PCI_NET	79C970
remappable	remappable	PCI_PM	PciPM
remappable	remappable	PCI_USB	PciUSB
remappable	remappable	PIIX4-IDE	PciIDE
remappable	remappable	sysControl	GT6412x
remappable	remappable	vga	GD5446
0x8000000	0x800FFFF	busBridgeM1	bridge
0x18000020	0x18000021	intCtrlMaster	8259A
0x18000040	0x18000043	mpit	8253
0x18000060	0x18000067	Ps2Control	Ps2Control
0x18000070	0x18000071	mrtc	MC146818
0x180000A0	0x180000A1	intCtrlSlave	8259A
0x180002F8	0x180002FF	uartTTY1	16550
0x180003B0	0x180003DF	vga	GD5446
0x180003F0	0x180003F7	fd0	82077AA
0x180003F8	0x180003FF	uartTTY0	16550
0x180004D0	0x180004D0	intCtrlMaster	8259A
0x180004D1	0x180004D1	intCtrlSlave	8259A
0x1E000000	0x1E3FFFFF	map	bridge
0x1F000000	0x1F0008FF	maltaFpga	MaltaFPGA

0x1F000900	0x1F00093F	uartCBUS	16450C
0x1F000A00	0x1F000FFF	maltaFpga	MaltaFPGA
0x1FC00000	0x1FC0000F	remap1	bridge
0x1FC00010	0x1FC00013	Core_Board_SDRAM_promInit	SmartLoaderLinux
0x1FC00014	0x1FFFFFFF	remap2	bridge

Table 6. Bridged Memory Map ('mipsle1' / 'busBridgeM1' / 'busShared' [width: 32])

Lo Address	Hi Address	Instance	Component
0x0	0xFFFF	sharedRAM	ram

Table 7. Bridged Memory Map ('mipsle1' / 'map' / 'flashBus' [width: 32])

Lo Address	Hi Address	Instance	Component
0x1E000000	0x1E3FFFFF	Monitor_Flash	ram

Table 8. Bridged Memory Map ('mipsle1' / 'remap1' / 'flashBus' [width: 32])

Lo Address	Hi Address	Instance	Component
0x1E000000	0x1E3FFFFF	Monitor_Flash	ram

Table 9. Bridged Memory Map ('mipsle1' / 'remap2' / 'flashBus' [width: 32])

Lo Address	Hi Address	Instance	Component
0x1E000000	0x1E3FFFFF	Monitor_Flash	ram

3.4 Net Connections to processor: 'mipsle1'

Table 10. Processor Net Connections ('mipsle1')

Net Port	Net	Instance	Component
hwint0	i8259Int	intCtrlMaster	8259A

4.0 Processor [arm.ovpworld.org/processor/arm/1.0] instance: armSub1arm1

4.1 Processor model type: 'arm' variant 'ARM920T' 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/arm.ovpworld.org/processor/arm/1.0/doc
- the OVP website: OVP_Model_Specific_Information_arm_ARM920T.pdf

4.1.1 Description

ARM Processor Model

4.1.2 Licensing

Usage of binary model under license governing simulator usage.

Note that for models of ARM CPUs the license includes the following terms:
Licensee is granted a non-exclusive, worldwide, non-transferable, revocable licence to:
If no source is being provided to the Licensee: use and copy only (no modifications rights are granted) the model for the sole purpose of designing, developing, analyzing, debugging, testing, verifying, validating and optimizing software which: (a) (i) is for ARM based systems; and (ii) does not incorporate the ARM Models or any part thereof; and (b) such ARM Models may not be used to emulate an ARM based system to run application software in a production or live environment.
If source code is being provided to the Licensee: use, copy and modify the model for the sole purpose of designing, developing, analyzing, debugging, testing, verifying, validating and optimizing software which: (a) (i) is for ARM based systems; and (ii) does not incorporate the ARM Models or any part thereof; and (b) such ARM Models may not be used to emulate an ARM based system to run application software in a production or live environment.
In the case of any Licensee who is either or both an academic or educational institution the purposes shall be limited to internal use.
Except to the extent that such activity is permitted by applicable law, Licensee shall not reverse engineer, decompile, or disassemble this model. If this model was provided to Licensee in Europe, Licensee shall not reverse engineer, decompile or disassemble the Model for the purposes of error correction.
The License agreement does not entitle Licensee to manufacture in silicon any product based on this model.
The License agreement does not entitle Licensee to use this model for evaluating the validity of any ARM patent.
Source of model available under separate Imperas Software License Agreement.

4.1.3 Limitations

Instruction pipelines are not modeled in any way. All instructions are assumed to complete immediately. This means that instruction barrier instructions (e.g. ISB, CP15ISB) are treated as NOPs, with the exception of any undefined instruction behavior, which is modeled. The model does not implement speculative fetch behavior. The branch cache is not modeled.
Caches and write buffers are not modeled in any way. All loads, fetches and stores complete immediately and in order, and are fully synchronous (as if the memory was of Strongly Ordered or Device-nGnRnE type). Data barrier instructions (e.g. DSB, CP15DSB) are treated as NOPs, with the exception of any undefined instruction behavior, which is modeled. Cache manipulation instructions are implemented as NOPs, with the exception of any undefined instruction behavior, which is modeled.
Real-world timing effects are not modeled: all instructions are assumed to complete in a single cycle.
TLBs are architecturally-accurate but not device accurate. This means that all TLB maintenance and address translation operations are fully implemented but the cache is larger than in the real device.

4.1.4 Verification

Models have been extensively tested by Imperas. ARM9 models have been successfully used by customers to simulate Linux and Nucleus on ArmIntegrator virtual platforms.

4.1.5 Core Features

Thumb instructions are supported.

4.1.6 Memory System

FCSE extension is implemented.
VMSA address translation is implemented.
TLB behavior is controlled by parameter ASIDCacheSize. If this parameter is 0, then an unlimited number of TLB entries will be maintained concurrently. If this parameter is non-zero, then only TLB entries for up to ASIDCacheSize different ASIDs will be maintained concurrently initially; as new ASIDs are used, TLB entries for less-recently used ASIDs are deleted, which improves model performance in some cases (especially when 16-bit ASIDs are in use). If the model detects that the TLB entry cache is too small (entry ejections are very frequent), it will increase the cache size automatically. In this variant, ASIDCacheSize is 8

4.1.7 Debug Mask

It is possible to enable model debug features in various categories. This can be done statically using the "override_debugMask" parameter, or dynamically using the "debugflags" command. Enabled debug features are specified using a bitmask value, as follows:
Value 0x004: enable debugging of MMU/MPU mappings.
Value 0x080: enable debugging of all system register accesses.
Value 0x100: enable debugging of all traps of system register accesses.
Value 0x200: enable verbose debugging of other miscellaneous behavior (for example, the reason why a particular instruction is undefined).
Value 0x800: enable dynamic validation of TLB entries against in-memory page table contents (finds some classes of error where page table entries are updated without a subsequent flush of affected TLB entries).
All other bits in the debug bitmask are reserved and must not be set to non-zero values.

4.1.8 AArch32 Unpredictable Behavior

Many AArch32 instruction behaviors are described in the ARM ARM as CONSTRAINED UNPREDICTABLE. This section describes how such situations are handled by this model.

4.1.9 Equal Target Registers

Some instructions allow the specification of two target registers (for example, double-width SMULL, or some VMOV variants), and such instructions are CONSTRAINED UNPREDICTABLE if the same target register is specified in both positions. In this model, such instructions are treated as UNDEFINED.

4.1.10 Floating Point Load/Store Multiple Lists

Instructions that load or store a list of floating point registers (e.g. VSTM, VLDM, VPUSH, VPOP) are CONSTRAINED UNPREDICTABLE if either the uppermost register in the specified range is greater than 32 or (for 64-bit registers) if more than 16 registers are specified. In this model, such instructions are treated as UNDEFINED.

4.1.11 Floating Point VLD[2-4]/VST[2-4] Range Overflow

Instructions that load or store a fixed number of floating point registers (e.g. VST2, VLD2) are CONSTRAINED UNPREDICTABLE if the upper register bound exceeds the number of implemented floating point registers. In this model, these instructions load and store using modulo 32 indexing (consistent with AArch64 instructions with similar behavior).

4.1.12 If-Then (IT) Block Constraints

Where the behavior of an instruction in an if-then (IT) block is described as CONSTRAINED UNPREDICTABLE, this model treats that instruction as UNDEFINED.

4.1.13 Use of R13

In architecture variants before ARMv8, use of R13 was described as CONSTRAINED UNPREDICTABLE in many circumstances. From ARMv8, most of these situations are no longer considered unpredictable. This model allows R13 to be used like any other GPR, consistent with the ARMv8 specification.

4.1.14 Use of R15

Use of R15 is described as CONSTRAINED UNPREDICTABLE in many circumstances. This model allows such use to be configured using the parameter "unpredictableR15" as follows:
Value "undefined": any reference to R15 in such a situation is treated as UNDEFINED;
Value "nop": any reference to R15 in such a situation causes the instruction to be treated as a NOP;
Value "raz_wi": any reference to R15 in such a situation causes the instruction to be treated as a RAZ/WI (that is, R15 is read as zero and write-ignored);
Value "execute": any reference to R15 in such a situation is executed using the current value of R15 on read, and writes to R15 are allowed (but are not interworking).
Value "assert": any reference to R15 in such a situation causes the simulation to halt with an assertion message (allowing any such unpredictable uses to be easily identified).
In this variant, the default value of "unpredictableR15" is "execute".

4.1.15 Unpredictable Instructions in Some Modes

Some instructions are described as CONSTRAINED UNPREDICTABLE in some modes only (for example, MSR accessing SPSR is CONSTRAINED UNPREDICTABLE in User and System modes). This model allows such use to be configured using the parameter "unpredictableModal", which can have values "undefined" or "nop". See the previous section for more information about the meaning of these values.
In this variant, the default value of "unpredictableModal" is "nop".

4.1.16 Integration Support

This model implements a number of non-architectural pseudo-registers and other features to facilitate integration.

4.1.17 Memory Transaction Query

Two registers are intended for use within memory callback functions to provide additional information about the current memory access. Register transactPL indicates the processor execution level of the current access (0-3). Note that for load/store translate instructions (e.g. LDRT, STRT) the reported execution level will be 0, indicating an EL0 access. Register transactAT indicates the type of memory access: 0 for a normal read or write; and 1 for a physical access resulting from a page table walk.

4.1.18 Page Table Walk Query

A banked set of registers provides information about the most recently completed page table walk. There are up to six banks of registers: bank 0 is for stage 1 walks, bank 1 is for stage 2 walks, and banks 2-5 are for stage 2 walks initiated by stage 1 level 0-3 entry lookups, respectively. Banks 1-5 are present only for processors with virtualization extensions. The currently active bank can be set using register PTWBankSelect. Register PTWBankValid is a bitmask indicating which banks contain valid data: for example, the value 0xb indicates that banks 0, 1 and 3 contain valid data.
Within each bank, there are registers that record addresses and values read during that page table walk. Register PTWBase records the table base address, register PTWInput contains the input address that starts a walk, register PTWOutput contains the result address and register PTWPgSize contains the page size (PTWOutput and PTWPgSize are valid only if the page table walk completes). Registers PTWAddressL0-PTWAddressL3 record the addresses of level 0 to level 3 entries read, respectively. Register PTWAddressValid is a bitmask indicating which address registers contain valid data: bits 0-3 indicate PTWAddressL0-PTWAddressL3, respectively, bit 4 indicates PTWBase, bit 5 indicates PTWInput, bit 6 indicates both PTWOutput and PTWPgSize. For example, the value 0x73 indicates that PTWBase, PTWInput, PTWOutput, PTWPgSize and PTWAddressL0-L1 are valid but PTWAddressL2-L3 are not. Register PTWAddressNS is a bitmask indicating whether an address is in non-secure memory: bits 0-3 indicate PTWAddressL0-PTWAddressL3, respectively, bit 4 indicates PTWBase, bit 6 indicates PTWOutput (PTWInput is a VA and thus has no secure/non-secure info). Registers PTWValueL0-PTWValueL3 contain page table entry values read at level 0 to level 3. Register PTWValueValid is a bitmask indicating which value registers contain valid data: bits 0-3 indicate PTWValueL0-PTWValueL3, respectively.

4.1.19 Artifact Page Table Walks

Registers are also available to enable a simulation environment to initiate an artifact page table walk (for example, to determine the ultimate PA corresponding to a given VA). Register PTWI_EL1S initiates a secure EL1 table walk for a fetch. Register PTWD_EL1S initiates a secure EL1 table walk for a load or store (note that current ARM processors have unified TLBs, so these registers are synonymous). Registers PTW[ID]_EL1NS initiate walks for non-secure EL1 accesses. Registers PTW[ID]_EL2 initiate EL2 walks. Registers PTW[ID]_S2 initiate stage 2 walks. Registers PTW[ID]_EL3 initiate AArch64 EL3 walks. Finally, registers PTW[ID]_current initiate current-mode walks (useful in a memory callback context). Each walk fills the query registers described above.

4.1.20 MMU and Page Table Walk Events

Two events are available that allow a simulation environment to be notified on MMU and page table walk actions. Event mmuEnable triggers when any MMU is enabled or disabled. Event pageTableWalk triggers on completion of any page table walk (including artifact walks).

4.1.21 Artifact Address Translations

A simulation environment can trigger an artifact address translation operation by writing to the architectural address translation registers (e.g. ATS1CPR). The results of such translations are written to an integration support register artifactPAR, instead of the architectural PAR register. This means that such artifact writes will not perturb architectural state.

4.1.22 TLB Invalidation

A simulation environment can cause TLB state for one or more address translation regimes in the processor to be flushed by writing to the artifact register ResetTLBs. The argument is a bitmask value, in which non-zero bits select the TLBs to be flushed, as follows:
Bit 1: EL0/EL1 stage 1 non-secure TLB

4.1.23 Halt Reason Introspection

An artifact register HaltReason can be read to determine the reason or reasons that a processor is halted. This register is a bitfield, with the following encoding: bit 0 indicates the processor has executed a wait-for-event (WFE) instruction; bit 1 indicates the processor has executed a wait-for-interrupt (WFI) instruction; and bit 2 indicates the processor is held in reset.

4.1.24 System Register Access Monitor

If parameter "enableSystemMonitorBus" is True, an artifact 32-bit bus "SystemMonitor" is enabled for each PE. Every system register read or write by that PE is then visible as a read or write on this artifact bus, and can therefore be monitored using callbacks installed in the client environment (use opBusReadMonitorAdd/opBusWriteMonitorAdd or icmAddBusReadCallback/icmAddBusWriteCallback, depending on the client API). The format of the address on the bus is as follows:
bits 31:26 - zero
bit 25 - 1 if AArch64 access, 0 if AArch32 access
bit 24 - 1 if non-secure access, 0 if secure access
bits 23:20 - CRm value
bits 19:16 - CRn value
bits 15:12 - op2 value
bits 11:8 - op1 value
bits 7:4 - op0 value (AArch64) or coprocessor number (AArch32)
bits 3:0 - zero
As an example, to view non-secure writes to writes to CNTFRQ_EL0 in AArch64 state, install a write monitor on address range 0x020e0330:0x020e0333.

4.1.25 System Register Implementation

If parameter "enableSystemBus" is True, an artifact 32-bit bus "System" is enabled for each PE. Slave callbacks installed on this bus can be used to implement modified system register behavior (use opBusSlaveNew or icmMapExternalMemory, depending on the client API). The format of the address on the bus is the same as for the system monitor bus, described above.

4.2 Instance Parameters

Several parameters can be specified when a processor is instanced in a platform. For this processor instance 'armSub1arm1' it has been instanced with the following parameters:

Table 11. Processor Instance 'armSub1arm1' Parameters (Configurations)

Parameter	Value	Description
endian	little	Select processor endian (big or little)

simulateexceptions	simulateexceptions	Causes the processor simulate exceptions instead of halting
mips	200.0	The nominal MIPS for the processor
imagefile	arm.plus_demo.out	The image file to load onto the processor

Table 12. Processor Instance 'armSub1arm1' Parameters (Attributes)

Parameter Name	Value	Type
variant	ARM920T	enum
compatibility	ISA	enum
showHiddenRegs	0	bool

4.3 Memory Map for processor 'armSub1arm1' bus: 'busarmSub1'

Processor instance 'armSub1arm1' is connected to bus 'busarmSub1' using master port 'INSTRUCTION'.
Processor instance 'armSub1arm1' is connected to bus 'busarmSub1' using master port 'DATA'.

Table 13. Memory Map ('armSub1arm1' / 'busarmSub1' [width: 32])

Lo Address	Hi Address	Instance	Component
0x0	0x7FFFFFF	armSub1ram1Bridge	bridge
remappable	remappable	armSub1lcd	LcdPL110
0xA000000	0xA00FFFF	busBridgeA1	bridge
0x10000000	0x10000FFF	armSub1cm	CoreModule9x6
0x13000000	0x13000FFF	armSub1pit	IcpCounterTimer
0x14000000	0x14000FFF	armSub1pic1	IntICP
0x15000000	0x15000FFF	armSub1rtc	RtcPL031
0x16000000	0x16000FFF	armSub1uart1	UartPL011
0x17000000	0x17000FFF	armSub1uart2	UartPL011
0x18000000	0x18000FFF	armSub1kb1	KbPL050
0x19000000	0x19000FFF	armSub1ms1	KbPL050
0x1A000000	0x1A000FFF	armSub1ld1	DebugLedAndDipSwitch
0x1C000000	0x1C000FFF	armSub1mmci	MmciPL181
0x1D000000	0x1D000FFF	armSub1ambaDummy	ram
0x80000000	0x87FFFFFF	armSub1ram1	ram
0xC0000000	0xC0000FFF	armSub1lcd	LcdPL110
0xCA000000	0xCA000FFF	armSub1pic2	IntICP
0xCB000000	0xCB00000F	armSub1icp	IcpControl

Table 14. Bridged Memory Map ('armSub1arm1' / 'armSub1ram1Bridge' / 'busarmSub1' [width: 32])

Lo Address	Hi Address	Instance	Component
remappable	remappable	armSub1lcd	LcdPL110
0x80000000	0x87FFFFFF	armSub1ram1	ram

Table 15. Bridged Memory Map ('armSub1arm1' / 'busBridgeA1' / 'busShared' [width: 32])

Lo Address	Hi Address	Instance	Component
0x0	0xFFFF	sharedRAM	ram

4.4 Net Connections to processor: 'armSub1arm1'

Table 16. Processor Net Connections ('armSub1arm1')

Net Port	Net	Instance	Component
irq	armSub1irq	armSub1pic1	IntICP
fiq	armSub1fiq	armSub1pic1	IntICP

5.0 Peripheral Instances

5.1 Peripheral [mips.ovpworld.org/peripheral/SmartLoaderLinux/1.0] instance: Core_Board_SDRAM_promInit

5.1.1 Licensing

Open Source Apache 2.0

5.1.2 Description

Smart peripheral creates memory initialisation for a MIPS32 based Linux kernel boot. Performs the generation of boot code at the reset vector (virtual address 0xbfc00000) of the MIPS32 processor. Loads both the linux kernel and initial ramdisk into memory and patches the boot code to jump to the kernel start. Initialises the MIPS32 registers and Linux command line.

5.1.3 Reference

MIPS Malta User Manual. MIPS Boot code reference.

5.1.4 Limitations

None

Table 17. Configuration options (attributes) set for instance 'Core_Board_SDRAM_promInit'

Attribute	Value	Type	Expression
kernel	mips.vmlinux	string	
boardid	0x00000420	uns32	
initrd	mips.initrd.gz	string	
command	console=tty0	string	

5.2 Peripheral [marvell.ovpworld.org/peripheral/GT6412x/1.0] instance: sysControl

5.2.1 Licensing

Open Source Apache 2.0

5.2.2 Description

GT64120 System Controller.
Diagnostic levels:
PCI_SLAVE 0x03
PCI_CONFIG_MASTER 0x04

PCI_EMPTY 0x08
INT_ACK 0x10
MAIN_BUS 0x20
INFO 0x40

5.2.3 Limitations

This model has sufficient functionality to allow a Linux Kernel to Boot on the MIPS:MALTA platform.

5.2.4 Reference

GT64121A System Controller for RC4650/4700/5000 and RM526X/527X/7000 CPUs Datasheet Revision 1.0 MAR 14, 2000

There are no configuration options set for this peripheral instance.

5.3 Peripheral [intel.ovpworld.org/peripheral/82371EB/1.0] instance: PIIX4

5.3.1 Licensing

Open Source Apache 2.0

5.3.2 Description

PIIX4 PCI configuration controller.

5.3.3 Limitations

This model has sufficient functionality to allow a Linux Kernel to Boot on the MIPS:MALTA platform.

5.3.4 Reference

Intel 82371EB South Bridge Chipset Datasheet

Table 18. Configuration options (attributes) set for instance 'PIIX4'

Attribute	Value	Type	Expression
PCIslot	10	uns32	

5.4 Peripheral [intel.ovpworld.org/peripheral/PciIDE/1.0] instance: PIIX4-IDE

5.4.1 Licensing

Open Source Apache 2.0

5.4.2 Description

PCI:IDE interface. This forms part of the 82371 PIIX4 chip. It implements 4 IDE interfaces and 2 DMA controllers.

5.4.3 Limitations

This model has sufficient functionality to allow a Linux Kernel to Boot on the MIPS:MALTA platform.

5.4.4 Reference
Intel 82371EB South Bridge Chipset Datasheet

Table 19. Configuration options (attributes) set for instance 'PIIX4-IDE'

Attribute	Value	Type	Expression
PCIslot	10	uns32	
PCIfunction	1	uns32	
Drive0Name	mipsel_hda	string	
Drive1Name	mipsel_hdb	string	
Drive2Name	mipsel_cd	string	

5.5 Peripheral [intel.ovpworld.org/peripheral/PciUSB/1.0] instance: PCI_USB

5.5.1 Licensing
Open Source Apache 2.0

5.5.2 Description
PCI USB Interface

5.5.3 Limitations
This model has sufficient functionality to allow a Linux Kernel to Boot on the MIPS:MALTA platform.

5.5.4 Reference
Intel 82371EB South Bridge Chipset Datasheet

Table 20. Configuration options (attributes) set for instance 'PCI_USB'

Attribute	Value	Type	Expression
PCIslot	10	uns32	
PCIfunction	2	uns32	

5.6 Peripheral [intel.ovpworld.org/peripheral/PciPM/1.0] instance: PCI_PM

5.6.1 Licensing
Open Source Apache 2.0

5.6.2 Description
PCI Power Manager.

5.6.3 Limitations
This model has sufficient functionality to allow a Linux Kernel to Boot on the MIPS:MALTA platform.

5.6.4 Reference
Intel 82371EB South Bridge Chipset Datasheet

Table 21. Configuration options (attributes) set for instance 'PCI_PM'

Attribute	Value	Type	Expression
PCIslot	10	uns32	
PCIfunction	3	uns32	

5.7 Peripheral [amd.ovpworld.org/peripheral/79C970/1.0] instance: PCI_NET

5.7.1 Licensing

Open Source Apache 2.0

5.7.2 Description

Implements part of the AMD AM79C97xx series Ethernet devices.
diagnosticlevel: bits 0:1 give levels for the network hardware. bits 2:3 give levels for the user:mode SLIRP interface.

5.7.3 Limitations

Sufficient is implemented to Boot MIPS Linux and support ethernet TCP/IP services.

5.7.4 Reference

AMD Am79C973/Am79C975 PCnet-FAST III Single-Chip 10/100 Mbps PCI Ethernet Controller with Integrated PHY Datasheet

Table 22. Configuration options (attributes) set for instance 'PCI_NET'

Attribute	Value	Type	Expression
PCIslot	11	uns32	
PCIfunction	0	uns32	

5.8 Peripheral [intel.ovpworld.org/peripheral/8259A/1.0] instance: intCtrlMaster

5.8.1 Licensing

Open Source Apache 2.0

5.8.2 Description

Intel 8259A Programmable Interrupt Controller (PIT).

5.8.3 Limitations

This model has sufficient functionality to allow a Linux Kernel to Boot on the MIPS:MALTA platform.

5.8.4 Reference

Intel 8259A Datasheet. MIPS Malta Platform Reference Guide.

Table 23. Configuration options (attributes) set for instance 'intCtrlMaster'

Attribute	Value	Type	Expression
spen	master	enum	

5.9 Peripheral [intel.ovpworld.org/peripheral/8259A/1.0] instance: intCtrlSlave

5.9.1 Licensing

Open Source Apache 2.0

5.9.2 Description

Intel 8259A Programmable Interrupt Controller (PIT).

5.9.3 Limitations

This model has sufficient functionality to allow a Linux Kernel to Boot on the MIPS:MALTA platform.

5.9.4 Reference

Intel 8259A Datasheet. MIPS Malta Platform Reference Guide.

Table 24. Configuration options (attributes) set for instance 'intCtrlSlave'

Attribute	Value	Type	Expression
spen	slave	enum	

5.10 Peripheral [cirrus.ovpworld.org/peripheral/GD5446/1.0] instance: vga

5.10.1 Licensing

Open Source Apache 2.0

5.10.2 Description

Cirrus CL GD5446 VGA controller.

5.10.3 Limitations

This model has sufficient functionality to allow a Linux Kernel to Boot on the MIPS:MALTA platform. The VGA peripheral utilises memory mapping. This requires the use of ICM memory for the frame buffers, which currently may stop its use in SystemC TLM2 platforms.

5.10.4 Reference

CL-GD5446 Preliminary Databook, Version 2.0, November 1996

Table 25. Configuration options (attributes) set for instance 'vga'

Attribute	Value	Type	Expression
scanDelay	50000	uns32	
PCIslot	18	uns32	
title	Imperas MIPS32 Malta	string	
noGraphics		bool	

5.11 Peripheral [intel.ovpworld.org/peripheral/Ps2Control/1.0] instance: Ps2Control

5.11.1 Licensing

Open Source Apache 2.0

5.11.2 Description

PS2 Keyboard/Mouse Controller.

5.11.3 Limitations

This is a preliminary model with sufficient functionality to enable Linux to Boot on the MIPS:MALTA platform. Mouse functions are currently turned off.

5.11.4 Reference

SMsC FDC37M817 Super I/O Controller Datasheet

Table 26. Configuration options (attributes) set for instance 'Ps2Control'

Attribute	Value	Type	Expression
pollPeriod	5000	uns32	
grabDisable	1	bool	

5.12 Peripheral [intel.ovpworld.org/peripheral/8253/1.0] instance: mpit

5.12.1 Description

Intel 8253 Programmable Interval Timer (PIT)

5.12.2 Limitations

This model has sufficient functionality to allow a Linux Kernel to Boot on the MIPS:MALTA platform. Not all modes are supported.

5.12.3 Licensing

Open Source Apache 2.0

5.12.4 Reference

Intel 8253 Datasheet. MIPS Malta Platform Reference Guide.

There are no configuration options set for this peripheral instance.

5.13 Peripheral [motorola.ovpworld.org/peripheral/MC146818/1.0] instance: mrtc

5.13.1 Licensing

Open Source Apache 2.0

5.13.2 Description

MC146818 Real:time clock.

5.13.3 Limitations
This model has sufficient functionality to allow a Linux Kernel to Boot on the MIPS:MALTA platform.

5.13.4 Reference
Motorola MC146818AS Datasheet

There are no configuration options set for this peripheral instance.

5.14 Peripheral [national.ovpworld.org/peripheral/16550/1.0] instance: uartTTY0

5.14.1 Licensing
Open Source Apache 2.0

5.14.2 Description
16550 UART model
The serial input/output from the simulator is implemented using the Serial Device Support described in OVP BHM and PPM API Functions Reference, which describes the parameters that control how the model interacts with the host computer.
Interrupts and FIFOs are supported.
Registers are aligned on 1 byte boundaries.

5.14.3 Limitations
Resolution of the baud rate is limited to the simulation time slice (aka quantum) size.
Values written to the MCR are ignored. Loopback mode is not supported.
The LSR is read-only. The model never sets the LSR 'Parity Error', 'Framing Error', 'Break Interrupt' or 'Error in RCVR FIFO' bits.
The MSR 'Data Set Ready' and 'Clear To Send' bits are set at reset and all other MSR bits are cleared. MSR bits will only be changed by writes to the MSR and values written to the Modem Status Register do not effect the operation of the model.

5.14.4 Reference
PC16550D Universal Asynchronous Receiver/Transmitter with FIFOs datasheet (http://www.ti.com/lit/ds/symlink/pc16550d.pdf)

There are no configuration options set for this peripheral instance.

5.15 Peripheral [national.ovpworld.org/peripheral/16550/1.0] instance: uartTTY1

5.15.1 Licensing
Open Source Apache 2.0

5.15.2 Description

16550 UART model
The serial input/output from the simulator is implemented using the Serial Device Support described in OVP BHM and PPM API Functions Reference, which describes the parameters that control how the model interacts with the host computer.
Interrupts and FIFOs are supported.
Registers are aligned on 1 byte boundaries.

5.15.3 Limitations

Resolution of the baud rate is limited to the simulation time slice (aka quantum) size.
Values written to the MCR are ignored. Loopback mode is not supported.
The LSR is read-only. The model never sets the LSR 'Parity Error', 'Framing Error', 'Break Interrupt' or 'Error in RCVR FIFO' bits.
The MSR 'Data Set Ready' and 'Clear To Send' bits are set at reset and all other MSR bits are cleared. MSR bits will only be changed by writes to the MSR and values written to the Modem Status Register do not effect the operation of the model.

5.15.4 Reference

PC16550D Universal Asynchronous Receiver/Transmitter with FIFOs datasheet (http://www.ti.com/lit/ds/symlink/pc16550d.pdf)

There are no configuration options set for this peripheral instance.

5.16 Peripheral [mips.ovpworld.org/peripheral/16450C/1.0] instance: uartCBUS

5.16.1 Licensing

Open Source Apache 2.0

5.16.2 Description

Model of 16550/16450 UART.
Special version with register addresses for MIPS MALTA C-BUS.
Connects to a bus by a slave port and optionally to a processor by an interrupt signal.
The serial input/output ports are modelled by socket connection which must be attached to a process outside the simulation environment.
Note that on start:up, the UART model will block the simulator, pending a connection to the socket.

5.16.3 Limitations

No modelling of baud:rate.
No modem support (DTR etc).
No support for parity.
No means to simulate errors.

5.16.4 Reference

MIPS Malta Datasheet

There are no configuration options set for this peripheral instance.

5.17 Peripheral [intel.ovpworld.org/peripheral/82077AA/1.0] instance: fd0

5.17.1 Licensing

Open Source Apache 2.0

5.17.2 Description

Dummy Floppy Disc Controller.

5.17.3 Limitations

Register stubs only.

5.17.4 Reference

http://www.buchty.net/casio/files/82077.pdf http://www.alldatasheet.com/datesheet-pdf/pdf/167793/INTEL/82077AA.html

There are no configuration options set for this peripheral instance.

5.18 Peripheral [mips.ovpworld.org/peripheral/MaltaFPGA/1.0] instance: maltaFpga

5.18.1 Licensing

Open Source Apache 2.0

5.18.2 Description

MIPS MALTA FPGA. Drives Development board functions.

5.18.3 Limitations

This model has sufficient functionality to allow a Linux Kernel to Boot on the MIPS:MALTA platform.

5.18.4 Reference

MIPS Malta User Manual.

Table 27. Configuration options (attributes) set for instance 'maltaFpga'

Attribute	Value	Type	Expression
stoponsoftreset	1	bool	

5.19 Peripheral [arm.ovpworld.org/peripheral/CoreModule9x6/1.0] instance: armSub1cm

5.19.1 Description

ARM Integrator Board 9x6 Core Module Registers

5.19.2 Limitations
none

5.19.3 Reference
ARM Integrator CM926EJ-S, CM946E-S, CM966E-S, CM1026EJ-S, and CM1136JF-S User Guide (DUI 0138)

5.19.4 Licensing
Open Source Apache 2.0

There are no configuration options set for this peripheral instance.

5.20 Peripheral [arm.ovpworld.org/peripheral/IntICP/1.0] instance: armSub1pic1

5.20.1 Description
ARM Integrator Board interrupt controller

5.20.2 Limitations
none

5.20.3 Reference
Integrator User Guide Compact Platform Baseboard HBI-0086 (DUI 0159B)

5.20.4 Licensing
Open Source Apache 2.0

There are no configuration options set for this peripheral instance.

5.21 Peripheral [arm.ovpworld.org/peripheral/IntICP/1.0] instance: armSub1pic2

5.21.1 Description
ARM Integrator Board interrupt controller

5.21.2 Limitations
none

5.21.3 Reference
Integrator User Guide Compact Platform Baseboard HBI-0086 (DUI 0159B)

5.21.4 Licensing
Open Source Apache 2.0

There are no configuration options set for this peripheral instance.

5.22 Peripheral [arm.ovpworld.org/peripheral/IcpCounterTimer/1.0] instance: armSub1pit

5.22.1 Description
ARM Integrator Board Counter/Timer Module

5.22.2 Limitations
none

5.22.3 Reference
Integrator User Guide Compact Platform Baseboard HBI-0086 (DUI 0159B)

5.22.4 Licensing
Open Source Apache 2.0

There are no configuration options set for this peripheral instance.

5.23 Peripheral [arm.ovpworld.org/peripheral/IcpControl/1.0] instance: armSub1icp

5.23.1 Description
ARM Integrator Board Controller Module

5.23.2 Limitations
none

5.23.3 Reference
Integrator User Guide Compact Platform Baseboard HBI-0086 (DUI 0159B)

5.23.4 Licensing
Open Source Apache 2.0

There are no configuration options set for this peripheral instance.

5.24 Peripheral [arm.ovpworld.org/peripheral/DebugLedAndDipSwitch/1.0] instance: armSub1ld1

5.24.1 Description
ARM Integrator Board Debug LEDs and DIP Switch Interface

5.24.2 Limitations
none

5.24.3 Reference
Integrator User Guide Compact Platform Baseboard HBI-0086 (DUI 0159B)

5.24.4 Licensing
Open Source Apache 2.0

There are no configuration options set for this peripheral instance.

5.25 Peripheral [arm.ovpworld.org/peripheral/KbPL050/1.0] instance: armSub1kb1

5.25.1 Description
ARM PL050 PS2 Keyboard or mouse controller

5.25.2 Limitations
None

5.25.3 Reference
ARM PrimeCell PS2 Keyboard/Mouse Interface (PL050) Technical Reference Manual (ARM DDI 0143)

5.25.4 Licensing
Open Source Apache 2.0

Table 28. Configuration options (attributes) set for instance 'armSub1kb1'

Attribute	Value	Type	Expression
isMouse	0	bool	
grabDisable	0	bool	

5.26 Peripheral [arm.ovpworld.org/peripheral/KbPL050/1.0] instance: armSub1ms1

5.26.1 Description
ARM PL050 PS2 Keyboard or mouse controller

5.26.2 Limitations
None

5.26.3 Reference
ARM PrimeCell PS2 Keyboard/Mouse Interface (PL050) Technical Reference Manual (ARM DDI 0143)

5.26.4 Licensing
Open Source Apache 2.0

Table 29. Configuration options (attributes) set for instance 'armSub1ms1'

Attribute	Value	Type	Expression
isMouse	1	bool	
grabDisable	1	bool	

5.27 Peripheral [arm.ovpworld.org/peripheral/RtcPL031/1.0] instance: armSub1rtc

5.27.1 Description

ARM PL031 Real Time Clock (RTC)

5.27.2 Limitations

none

5.27.3 Reference

ARM PrimeCell Real Time Clock (PL031) Technical Reference Manual (ARM DDI 0224)

5.27.4 Licensing

Open Source Apache 2.0

There are no configuration options set for this peripheral instance.

5.28 Peripheral [arm.ovpworld.org/peripheral/UartPL011/1.0] instance: armSub1uart1

5.28.1 Description

ARM PL011 UART

5.28.2 Limitations

This is not a complete model of the PL011. There is no modeling of physical aspects of the UART, such as baud rates etc.

5.28.3 Reference

ARM PrimeCell UART (PL011) Technical Reference Manual (ARM DDI 0183)

5.28.4 Licensing

Open Source Apache 2.0

Table 30. Configuration options (attributes) set for instance 'armSub1uart1'

Attribute	Value	Type	Expression
variant	ARM	enum	
outfile	uart1.log	string	
finishOnDisconnect	1	bool	

5.29 Peripheral [arm.ovpworld.org/peripheral/UartPL011/1.0] instance: armSub1uart2

5.29.1 Description

ARM PL011 UART

5.29.2 Limitations

This is not a complete model of the PL011. There is no modeling of physical aspects of the UART, such as baud rates etc.

5.29.3 Reference

ARM PrimeCell UART (PL011) Technical Reference Manual (ARM DDI 0183)

5.29.4 Licensing

Open Source Apache 2.0

Table 31. Configuration options (attributes) set for instance 'armSub1uart2'

Attribute	Value	Type	Expression
variant	ARM	enum	
finishOnDisconnect	1	bool	

5.30 Peripheral [arm.ovpworld.org/peripheral/MmciPL181/1.0] instance: armSub1mmci

5.30.1 Description

ARM PrimeCell Multimedia Card Interface (MMCI)

5.30.2 Limitations

None

5.30.3 Licensing

Open Source Apache 2.0

5.30.4 Reference

ARM PrimeCell Multimedia Card Interface (Pl180) Technical Reference Manual (ARM DDI 0172)

There are no configuration options set for this peripheral instance.

5.31 Peripheral [arm.ovpworld.org/peripheral/LcdPL110/1.0] instance: armSub1lcd

5.31.1 Description

ARM PL110 LCD Controller

5.31.2 Limitations

The VGA display refresh is not optimised resulting in the VGA peripheral causing a limit on the maximum performance of a platform it contains to be around 300 MIPS (actual dependent upon refresh rate of LCD).
The LCD peripheral utilises memory watchpoints to optimise display refresh. This requires the use of ICM memory for the frame buffers, which currently may stop its use in SystemC TLM2 platforms.
Interrupts are not supported

5.31.3 Reference

ARM PrimeCell Color LCD Controller (PL111) Technical Reference Manual (ARM DDI 0293)

5.31.4 Licensing

Open Source Apache 2.0

Table 32. Configuration options (attributes) set for instance 'armSub1lcd'

Attribute	Value	Type	Expression
noGraphics	1	bool	

5.32 Peripheral [arm.ovpworld.org/peripheral/SmartLoaderArmLinux/1.0] instance: armSub1smartLoader

5.32.1 Licensing

Open Source Apache 2.0

5.32.2 Description

Psuedo-peripheral to perform memory initialisation for an ARM based Linux kernel boot: Loads Linux kernel image file and (optional) initial ram disk image into memory. Writes ATAG data into memory. Writes tiny boot code at physical memory base that configures the registers as expected by Linux Kernel and then jumps to boot address (image load address by default).

5.32.3 Limitations

Only supports little endian

5.32.4 Reference

See ARM Linux boot requirements in Linux source tree at documentation/arm/Booting

Table 33. Configuration options (attributes) set for instance 'armSub1smartLoader'

Attribute	Value	Type	Expression
disable	1	bool	

6.0 Overview of Imperas OVP Virtual Platforms

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

This second part of the document, includes information about Imperas OVP virtual platforms and modules, 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. A platform can be modeled as different levels of hierarchy using separately describable and compilable modules.

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

7.0 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.

8.0 Simulating Software

8.1 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.

8.2 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.

8.3 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.

8.4 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.

8.5 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.

8.6 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.

8.7 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
Simulation_Control_of_Platforms_and_Modules_User_Guide.pdf
Advanced_Simulation_Control_of_Platforms_and_Modules_User_Guide.pdf
OVP_Control_File_User_Guide.pdf

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

9.0 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.

9.1 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.

*9.2 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.

9.3 Debugging with the Imperas eGui and GDB

Imperas eGui 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.

*9.4 Debugging with the Imperas eGui and M*DBG*

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

9.5 Debugging with Imperas eGui and Eclipse

Imperas provide a GUI based on Eclipse called eGui. This provides a GUI front end to use with a standard GDB or the Imperas MPD (Multi-Processor Debugger).

The use of eGui is documented: eGui_Eclipse_User_Guide.pdf.

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.

9.6 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.

10.0 Modifying the Platform / Module

10.1 Platforms / Modules use C/C++ and OVP APIs

The Imperas and OVP simulators execute a platform / module that is written in C/C++ and that makes function calls into the simulator's APIs. Thus the virtual platform / module 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, modules, the peripherals, and the processors. You can find more information about these APIs on the OVP website and in the OVP API documentation.

10.2 Platforms/Modules/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, modules, and peripherals - it creates the C/C++ file that is compiled into the platform, module 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 or modules iGen creates the C files and also provides a native C++ SystemC TLM2 interface to allow the peripheral/module to be instantiated in SystemC TLM2 platforms.

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

10.3 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, modules, 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.

10.4 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.

10.5 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.

11.0 Available Virtual Platforms

Table 34. Imperas / OVP Extendable Platform Kits (13 available)

Name	Vendor
AlteraCycloneIII_3c120	altera.ovpworld.org
AlteraCycloneV_HPS	altera.ovpworld.org
ArmIntegratorCP	arm.ovpworld.org
ArmVersatileExpress	arm.ovpworld.org
ArmVersatileExpress-CA15	arm.ovpworld.org
ArmVersatileExpress-CA9	arm.ovpworld.org
AtmelAT91SAM7	atmel.ovpworld.org
FreescaleKinetis60	freescale.ovpworld.org
FreescaleKinetis64	freescale.ovpworld.org
FreescaleVybridVFxx	freescale.ovpworld.org
MipsMalta	mips.ovpworld.org
RenesasUPD70F3441	renesas.ovpworld.org
XilinxML505	xilinx.ovpworld.org

Table 35. Imperas General Virtual Platforms (6 available)

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
SiFiveFU540	imperas.ovpworld.org

Table 36. Imperas Modules (component of other platforms) (55 available)

Name	Vendor
AlteraCycloneIII_3c120	altera.ovpworld.org
AlteraCycloneV_HPS	altera.ovpworld.org
AE350	andes.ovpworld.org
ARMv8-A-FMv1	arm.ovpworld.org
ArmIntegratorCP	arm.ovpworld.org
ArmVersatileExpress	arm.ovpworld.org
ArmVersatileExpress-CA15	arm.ovpworld.org
ArmVersatileExpress-CA9	arm.ovpworld.org
AtmelAT91SAM7	atmel.ovpworld.org
ArmCortexMFreeRTOS	imperas.ovpworld.org
ArmCortexMuCOS-II	imperas.ovpworld.org
ArmuKernel	imperas.ovpworld.org
ArmuKernelDual	imperas.ovpworld.org
BareMetalMIPS	imperas.ovpworld.org
Dual_ARMv8-A-FMv1_VLAN	imperas.ovpworld.org
Hetero_1xArm_3xMips32	imperas.ovpworld.org
Hetero_ARM_RISCV_NeuralNetwork	imperas.ovpworld.org

Hetero_ARMv8-A-FMv1_Cortex-M3	imperas.ovpworld.org
Hetero_ARMv8-A-FMv1_MIPS_microAptiv	imperas.ovpworld.org
Hetero_AlteraCycloneV_HPS_AlteraCycloneIII_3c120	imperas.ovpworld.org
Hetero_ArmIntegratorCP_XilinxMicroBlaze	imperas.ovpworld.org
Hetero_ArmVersatileExpress_MipsMalta	imperas.ovpworld.org
Hetero_ArmVersatileExpress_XilinxMicroBlaze	imperas.ovpworld.org
Quad_ArmVersatileExpress-CA15	imperas.ovpworld.org
RiscvRV32FreeRTOS	imperas.ovpworld.org
MipsMalta	mips.ovpworld.org
iMX6S	nxp.ovpworld.org
RenesasUPD70F3441	renesas.ovpworld.org
ghs-multi	renesas.ovpworld.org
virtio	riscv.ovpworld.org
FaultInjection	safepower.ovpworld.org
PublicDemonstrator	safepower.ovpworld.org
Zynq_PL_DualMicroblaze	safepower.ovpworld.org
Zynq_PL_NoC	safepower.ovpworld.org
Zynq_PL_NoC_node	safepower.ovpworld.org
Zynq_PL_NostrumNoC	safepower.ovpworld.org
Zynq_PL_NostrumNoC_node	safepower.ovpworld.org
Zynq_PL_RO	safepower.ovpworld.org
Zynq_PL_SingleMicroblaze	safepower.ovpworld.org
Zynq_PL_TTELNoC	safepower.ovpworld.org
Zynq_PL_TTELNoC_node	safepower.ovpworld.org
Zynq_PL_TTELNoC_processing_node_public_demonstrator	safepower.ovpworld.org
Zynq_PL_TTELNoC_public_demonstrator	safepower.ovpworld.org
Zynq_PL_TTELNoC_sensor_actor_node_public_demonstrator	safepower.ovpworld.org
FU540	sifive.ovpworld.org
S51CC	sifive.ovpworld.org
coreip-s51-arty	sifive.ovpworld.org
coreip-s51-rtl	sifive.ovpworld.org
dualFifo	vendor.com
XilinxML505	xilinx.ovpworld.org
Zynq	xilinx.ovpworld.org
Zynq_PL_Default	xilinx.ovpworld.org
Zynq_PS	xilinx.ovpworld.org
zc702	xilinx.ovpworld.org
zc706	xilinx.ovpworld.org

Table 37. Imperas / OVP Bare Metal Virtual Platforms (22 available)

Name	Vendor
BareMetalNios_IISingle	altera.ovpworld.org
BareMetalArcSingle	arc.ovpworld.org
BareMetalArm7Single	arm.ovpworld.org
BareMetalArmCortexADual	arm.ovpworld.org
BareMetalArmCortexASingle	arm.ovpworld.org
BareMetalArmCortexASingleAngelTrap	arm.ovpworld.org
BareMetalArmCortexMSingle	arm.ovpworld.org

ArmCortexMFreeRTOS	imperas.ovpworld.org
ArmCortexMuCOS-II	imperas.ovpworld.org
BareMetalArmx1Mips32x3	imperas.ovpworld.org
Or1kUclinux	imperas.ovpworld.org
BareMetalM14KSingle	mips.ovpworld.org
BareMetalMips32Dual	mips.ovpworld.org
BareMetalMips32Single	mips.ovpworld.org
BareMetalMips64Single	mips.ovpworld.org
BareMetalMipsDual	mips.ovpworld.org
BareMetalMipsSingle	mips.ovpworld.org
BareMetalOr1kSingle	ovpworld.org
BareMetalM16cSingle	posedgesoft.ovpworld.org
BareMetalPowerPc32Single	power.ovpworld.org
BareMetalV850Single	renesas.ovpworld.org
ghs-multi	renesas.ovpworld.org

#