



OVP Guide to Using Processor Models

Model specific information for riscv_RV64GCV

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



| | |
|----------|--|
| Author | Imperas Software Limited |
| Version | 20200114.0 |
| Filename | OVP_Model.Specific.Information_riscv_RV64GCV.pdf |
| Created | 14 January 2020 |
| Status | OVP Standard Release |

Copyright Notice

Copyright (c) 2020 Imperas Software Limited. All rights reserved. This software and documentation contain information that is the property of Imperas Software Limited. The software and documentation are furnished under a license agreement and may be used or copied only in accordance with the terms of the license agreement. No part of the software and documentation may be reproduced, transmitted, or translated, in any form or by any means, electronic, mechanical, manual, optical, or otherwise, without prior written permission of Imperas Software Limited, or as expressly provided by the license agreement.

Right to Copy Documentation

The license agreement with Imperas permits licensee to make copies of the documentation for its internal use only. Each copy shall include all copyrights, trademarks, service marks, and proprietary rights notices, if any.

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.

Contents

| | | |
|----------|--|-----------|
| 1 | Overview | 1 |
| 1.1 | Description | 1 |
| 1.2 | Licensing | 1 |
| 1.3 | Extensions | 1 |
| 1.3.1 | Available (But Not Enabled) Extensions | 2 |
| 1.4 | General Features | 2 |
| 1.5 | Floating Point Features | 4 |
| 1.6 | Vector Extension | 4 |
| 1.6.1 | Vector Extension Parameters | 4 |
| 1.6.2 | Vector Extension Features | 5 |
| 1.6.3 | Vector Extension Versions | 5 |
| 1.6.4 | Version 0.7.1-draft-20190605 | 5 |
| 1.6.5 | Version 0.7.1-draft-20190605+ | 5 |
| 1.6.6 | Version 0.8-draft-20190906 | 6 |
| 1.6.7 | Version 0.8-draft-20191004 | 6 |
| 1.6.8 | Version 0.8-draft-20191117 | 6 |
| 1.6.9 | Version 0.8-draft-20191118 | 7 |
| 1.6.10 | Version 0.8 | 7 |
| 1.6.11 | Version master | 7 |
| 1.7 | Interrupts | 7 |
| 1.8 | Debug Mask | 7 |
| 1.9 | Integration Support | 8 |
| 1.9.1 | CSR Register External Implementation | 8 |
| 1.9.2 | LR/SC Active Address | 8 |
| 1.10 | Limitations | 8 |
| 1.11 | Verification | 9 |
| 1.12 | References | 9 |
| 2 | Configuration | 10 |
| 2.1 | Location | 10 |
| 2.2 | GDB Path | 10 |
| 2.3 | Semi-Host Library | 10 |
| 2.4 | Processor Endian-ness | 10 |
| 2.5 | QuantumLeap Support | 10 |
| 2.6 | Processor ELF code | 10 |
| 3 | All Variants in this model | 11 |

| | | |
|-----------|--|-----------|
| 4 | Bus Master Ports | 12 |
| 5 | Bus Slave Ports | 13 |
| 6 | Net Ports | 14 |
| 7 | FIFO Ports | 15 |
| 8 | Formal Parameters | 16 |
| 8.1 | Parameters with enumerated types | 17 |
| 8.1.1 | Parameter user_version | 17 |
| 8.1.2 | Parameter priv_version | 17 |
| 8.1.3 | Parameter vector_version | 17 |
| 8.1.4 | Parameter fp16_version | 18 |
| 8.1.5 | Parameter mstatus_fs_mode | 18 |
| 9 | Execution Modes | 19 |
| 10 | Exceptions | 20 |
| 11 | Hierarchy of the model | 21 |
| 11.1 | Level 1: Hart | 21 |
| 12 | Model Commands | 22 |
| 12.1 | Level 1: Hart | 22 |
| 12.1.1 | dumpTLB | 22 |
| 12.1.1.1 | Argument description | 22 |
| 12.1.2 | isync | 22 |
| 12.1.3 | itrace | 22 |
| 13 | Registers | 23 |
| 13.1 | Level 1: Hart | 23 |
| 13.1.1 | Core | 23 |
| 13.1.2 | Floating_point | 24 |
| 13.1.3 | Vector | 24 |
| 13.1.4 | User_Control_and_Status | 25 |
| 13.1.5 | Supervisor_Control_and_Status | 26 |
| 13.1.6 | Machine_Control_and_Status | 26 |
| 13.1.7 | Integration_support | 28 |

Chapter 1

Overview

This document provides the details of an OVP Fast Processor Model variant.

OVP Fast Processor Models are written in C and provide a C API for use in C based platforms. The models also provide a native interface for use in SystemC TLM2 platforms.

The models are written using the OVP VMI API that provides a Virtual Machine Interface 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. Most 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 model families are validated using technology provided by the processor IP owners. There is a companion document (OVP Guide to Using Processor Models) which explains the general concepts of OVP Fast Processor Models and their use. It is downloadable from the OVPworld website documentation pages.

1.1 Description

RISC-V RV64GCV 64-bit processor model

1.2 Licensing

This Model is released under the Open Source Apache 2.0

1.3 Extensions

The model has the following architectural extensions enabled, and the following bits in the misa CSR Extensions field will be set upon reset:

misa bit 0: extension A (atomic instructions)

misa bit 2: extension C (compressed instructions)

misa bit 3: extension D (double-precision floating point)

misa bit 5: extension F (single-precision floating point)

misa bit 8: RV32I/64I/128I base ISA

misa bit 12: extension M (integer multiply/divide instructions)

misa bit 18: extension S (Supervisor mode)

misa bit 20: extension U (User mode)

misa bit 21: extension V (vector instructions)

To specify features that can be dynamically enabled or disabled by writes to the misa register in addition to those listed above, use parameter “add_Extensions_mask”. This is a string parameter containing the feature letters to add; for example, value “DV” indicates that double-precision floating point and the Vector Extension can be enabled or disabled by writes to the misa register.

Legacy parameter “misa_Extensions_mask” can also be used. This Uns32-valued parameter specifies all writable bits in the misa Extensions field, replacing any value defined in the base variant.

Note that any features that are indicated as present in the misa mask but absent in the misa will be ignored. See the next section.

1.3.1 Available (But Not Enabled) Extensions

The following extensions are supported by the model, but not enabled by default in this variant:

misa bit 4: RV32E base ISA (NOT ENABLED)

misa bit 13: extension N (user-level interrupts) (NOT ENABLED)

misa bit 23: extension X (non-standard extensions present) (NOT ENABLED)

To add features from this list to the base variant, use parameter “add_Extensions”. This is a string parameter containing the feature letters to add; for example, value “DV” indicates that double-precision floating point and the Vector Extension should be enabled, if they are absent.

Legacy parameter “misa_Extensions” can also be used. This Uns32-valued parameter specifies the reset value for the misa CSR Extensions field, replacing any value defined in the base variant.

1.4 General Features

On this variant, the Machine trap-vector base-address register (mtvec) is writable. It can instead be configured as read-only using parameter “mtvec_is_ro”.

Values written to “mtvec” are masked using the value 0xffffffffffffd. A different mask of writable bits may be specified using parameter “mtvec_mask” if required. In addition, when Vectored interrupt mode is enabled, parameter “tvec_align” may be used to specify additional hardware-enforced base address alignment. In this variant, “tvec_align” defaults to 0, implying no alignment

constraint.

The initial value of “mtvec” is 0x0. A different value may be specified using parameter “mtvec” if required.

Values written to “stvec” are masked using the value 0xffffffffffffd. A different mask of writable bits may be specified using parameter “stvec_mask” if required. parameter “tvec_align” may be used to specify additional hardware-enforced base address alignment in the same manner as for the “mtvec” register, described above.

On reset, the model will restart at address 0x0. A different reset address may be specified using parameter “reset_address” if required.

On an NMI, the model will restart at address 0x0. A different NMI address may be specified using parameter “nmi_address” if required.

WFI will halt the processor until an interrupt occurs. It can instead be configured as a NOP using parameter “wfi_is_nop”. WFI timeout wait is implemented with a time limit of 0 (i.e. WFI causes an Illegal Instruction trap in Supervisor mode when mstatus.TW=1).

The “cycle” CSR is implemented in this variant. Set parameter “cycle_undefined” to True to instead specify that “cycle” is unimplemented and reads of it should trap to Machine mode.

The “time” CSR is implemented in this variant. Set parameter “time_undefined” to True to instead specify that “time” is unimplemented and reads of it should trap to Machine mode. Usually, the value of the “time” CSR should be provided by the platform - see notes below about the artifact “CSR” bus for information about how this is done.

The “instret” CSR is implemented in this variant. Set parameter “instret_undefined” to True to instead specify that “instret” is unimplemented and reads of it should trap to Machine mode.

A 16-bit ASID is implemented. Use parameter “ASID_bits” to specify a different implemented ASID size if required.

This variant supports address translation modes 0, 8 and 9. Use parameter “Sv_modes” to specify a bit mask of different modes if required.

Unaligned memory accesses are not supported by this variant. Set parameter “unaligned” to “T” to enable such accesses.

Unaligned memory accesses are not supported for AMO instructions by this variant. Set parameter “unalignedAMO” to “T” to enable such accesses.

16 PMP entries are implemented by this variant. Use parameter “PMP_registers” to specify a different number of PMP entries; set the parameter to 0 to disable the PMP unit. The PMP grain size (G) is 0, meaning that PMP regions as small as 4 bytes are implemented. Use parameter “PMP_grain” to specify a different grain size if required.

LR/SC instructions are implemented with a 1-byte reservation granule. A different granule size may be specified using parameter “lr_sc_grain”.

1.5 Floating Point Features

The D extension is enabled in this variant independently of the F extension. Set parameter “d_requires.f” to “T” to specify that the D extension requires the F extension to be enabled.

By default, the processor starts with floating-point instructions disabled (mstatus.FS=0). Use parameter “mstatus_FS” to force mstatus.FS to a non-zero value for floating-point to be enabled from the start.

The specification is imprecise regarding the conditions under which mstatus.FS is set to Dirty state (3). Parameter “mstatus_fs_mode” can be used to specify the required behavior in this model, as described below.

If “mstatus_fs_mode” is set to “always_dirty” then the model implements a simplified floating point status view in which mstatus.FS holds values 0 (Off) and 3 (Dirty) only; any write of values 1 (Initial) or 2 (Clean) from privileged code behave as if value 3 was written.

If “mstatus_fs_mode” is set to “write_1” then mstatus.FS will be set to 3 (Dirty) by any explicit write to the fflags, frm or fcsr control registers, or by any executed instruction that writes an FPR, or by any executed floating point compare or conversion to integer/unsigned that signals a floating point exception. Floating point compare or conversion to integer/unsigned instructions that do not signal an exception will not set mstatus.FS.

If “mstatus_fs_mode” is set to “write_any” then mstatus.FS will be set to 3 (Dirty) by any explicit write to the fflags, frm or fcsr control registers, or by any executed instruction that writes an FPR, or by any executed floating point compare or conversion even if those instructions do not signal a floating point exception.

In this variant, “mstatus_fs_mode” is set to “write_1”.

1.6 Vector Extension

This variant implements the RISC-V base vector extension with version specified in the References section of this document. Note that parameter “vector_version” can be used to select the required version, including the unstable “master” version corresponding to the active specification. See section “Vector Extension Versions” for detailed information about differences between each supported version.

1.6.1 Vector Extension Parameters

Parameter ELEN is used to specify the maximum size of a single vector element in bits (32 or 64). By default, ELEN is set to 64 in this variant.

Parameter VLEN is used to specify the number of bits in a vector register (a power of two in the range 32 to 2048). By default, VLEN is set to 512 in this variant.

Parameter SLEN is used to specify the striping distance (a power of two in the range 32 to 2048). By default, SLEN is set to 64 in this variant.

Parameter Zvlssseg is used to specify whether the Zvlssseg extension is implemented. By default,

Zvlsseg is set to 1 in this variant.

Parameter Zvamo is used to specify whether the Zvamo extension is implemented. By default, Zvamo is set to 1 in this variant.

Parameter Zvediv will be used to specify whether the Zvediv extension is implemented. This is not currently supported.

Parameter Zvqmac is used to specify whether the Zvqmac extension is implemented (from version 0.8-draft-20191117 only). By default, Zvqmac is set to 1 in this variant.

Parameter require_vstart0 is used to specify whether non-interruptible vector instructions require vstart=0. By default, require_vstart0 is set to 0 in this variant.

1.6.2 Vector Extension Features

The model implements the base vector extension with a maximum ELEN of 64. Striping, masking and polymorphism are all fully supported. Zvlsseg and Zvamo extensions are fully supported. The Zvediv extension specification is subject to change and therefore not yet supported.

Single precision and double precision floating point types are supported if those types are also supported in the base architecture (i.e. the corresponding D and F features must be present and enabled). Presently, the interaction of vector floating point with the Privileged Architecture is not well defined; this model assumes that vector floating point operations may only be executed if the base floating point unit is also enabled (i.e. mstatus.FS must be non-zero). Attempting to execute vector floating point instructions when mstatus.FS is 0 will cause an Illegal Instruction exception.

The model assumes that all vector memory operations must be aligned to the memory element size. Unaligned accesses will cause a Load/Store Address Alignment exception.

1.6.3 Vector Extension Versions

The Vector Extension specification has been under active development. To enable simulation of hardware that may be based on an older version of the specification, the model implements behavior for a number of previous versions of the specification. The differing features of these are listed below, in chronological order.

1.6.4 Version 0.7.1-draft-20190605

Stable 0.7.1 version of June 10 2019.

1.6.5 Version 0.7.1-draft-20190605+

Version 0.7.1, with some 0.8 and custom features. Not intended for general use.

1.6.6 Version 0.8-draft-20190906

Stable 0.8 draft of September 6 2019, with these changes compared to version 0.7.1-draft-20190605:

- tail vector and scalar elements preserved, not zeroed;
- vext.s.v, vmford.vv and vmford.vf instructions removed;
- encodings for vfmv.f.s, vfmv.s.f, vmv.s.x, vpopc.m, vfirst.m, vmsbf.m, vmsif.m, vmsof.m, viota.m and vid.v instructions changed;
- overlap constraints for slideup and slidedown instructions relaxed to allow overlap of destination and mask when SEW=1;
- 64-bit vector AMO operations replaced with SEW-width vector AMO operations;
- vsetvl and vsetvli instructions when rs1 = x0 preserve the current vl instead of selecting the maximum possible vl;
- instruction vfnvct.rod.f.f.w added (to allow narrowing floating point conversions with jamming semantics).

1.6.7 Version 0.8-draft-20191004

Stable 0.8 draft of October 4 2019, with these changes compared to version 0.8-draft-20190906:

- vwmaccsu and vwmaccus instruction encodings exchanged;
- vwsmaccsu and vwsmaccus instruction encodings exchanged.

1.6.8 Version 0.8-draft-20191117

Stable 0.8 draft of November 17 2019, with these changes compared to version 0.8-draft-20191004:

- indexed load/store instructions zero-extend offsets (previously, they were sign-extended);
- vslide1up/vslide1down instructions sign-extend XLEN values to SEW length (previously, they were zero-extended);
- vadc/vsbc instruction encodings require vm=0 (previously, they required vm=1);
- vmadc/vmsbc instruction encodings allow both vm=0, implying carry input is used, and vm=1, implying carry input is zero (previously, only vm=1 was permitted, implying carry input is used);
- vaaddu.vv, vaaddu.vx, vasubu.vv and vasubu.vx instructions added;
- vaadd.vv and vaadd.vx, instruction encodings changed;
- vaadd.vi instruction removed;
- all widening saturating scaled multiply-add instructions removed;
- vqmaccu.vv, vqmaccu.vx, vqmacc.vv, vqmacc.vx, vqmacc.vx, vqmaccsu.vx and vqmaccus.vx instructions added;

- CSR `vlenb` added (vector register length in bytes);
- load/store whole register instructions added;
- whole register move instructions added.

1.6.9 Version 0.8-draft-20191118

Stable 0.8 draft of November 18 2019, with these changes compared to version 0.8-draft-20191117:

- `vsetvl/vsetvli` with `rd!=zero` and `rs1=zero` sets `vl` to the maximum vector length.

1.6.10 Version 0.8

Stable 0.8 official release (commit 9a65519), with these changes compared to version 0.8-draft-20191118:

- vector context status in `mstatus` register is now implemented;
- whole register load and store operations have been restricted to a single register only;
- whole register move operations have been restricted to aligned groups of 1, 2, 4 or 8 registers only.

1.6.11 Version master

Unstable master version as of 14 December 2019 (commit 9a65519), currently identical to version 0.8.

1.7 Interrupts

The “reset” port is an active-high reset input. The processor is halted when “reset” goes high and resumes execution from the reset address specified using the “reset_address” parameter when the signal goes low. The “mcause” register is cleared to zero.

The “nmi” port is an active-high NMI input. The processor is halted when “nmi” goes high and resumes execution from the address specified using the “nmi_address” parameter when the signal goes low. The “mcause” register is cleared to zero.

All other interrupt ports are active high.

1.8 Debug Mask

It is possible to enable model debug messages in various categories. This can be done statically using the “override_debugMask” parameter, or dynamically using the “debugflags” command. Enabled messages are specified using a bitmask value, as follows:

Value 0x002: enable debugging of PMP and virtual memory state;

Value 0x004: enable debugging of interrupt state.

All other bits in the debug bitmask are reserved and must not be set to non-zero values.

1.9 Integration Support

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

1.9.1 CSR Register External Implementation

If parameter “enable_CSR_bus” is True, an artifact 16-bit bus “CSR” is enabled. Slave callbacks installed on this bus can be used to implement modified CSR behavior (use opBusSlaveNew or icmMapExternalMemory, depending on the client API). A CSR with index 0xABC is mapped on the bus at address 0xABC0; as a concrete example, implementing CSR “time” (number 0xC01) externally requires installation of callbacks at address 0xC010 on the CSR bus.

1.9.2 LR/SC Active Address

Artifact register “LRSCAddress” shows the active LR/SC lock address. The register holds all-ones if there is no LR/SC operation active.

1.10 Limitations

Instruction pipelines are not modeled in any way. All instructions are assumed to complete immediately. This means that instruction barrier instructions (e.g. fence.i) are treated as NOPs, with the exception of any Illegal Instruction behavior, which is 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. Data barrier instructions (e.g. fence) are treated as NOPs, with the exception of any Illegal Instruction behavior, which is modeled.

Real-world timing effects are not modeled: all instructions are assumed to complete in a single cycle.

The processor fully supports the architecturally-specified floating-point instructions.

Hardware Performance Monitor and Debug registers are not implemented and hardwired to zero.

The TLB is 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.

1.11 Verification

All instructions have been extensively tested by Imperas, using tests generated specifically for this model and also reference tests from <https://github.com/riscv/riscv-tests>.

Also reference tests have been used from various sources including:

<https://github.com/riscv/riscv-tests>

<https://github.com/ucb-bar/riscv-torture>

The Imperas OVPsim RISC-V models are used in the RISC-V Foundations Compliance Framework as a functional Golden Reference:

<https://github.com/riscv/riscv-compliance>

where the simulated model is used to provide the reference signatures for compliance testing. The Imperas OVPsim RISC-V models are used as reference in both open source and commercial instruction stream test generators for hardware design verification, for example:

<http://valtrix.in/sting/> from Valtrix

<https://github.com/google/riscv-dv> from Google

The Imperas OVPsim RISC-V models are also used by commercial and open source RISC-V Core RTL developers as a reference to ensure correct functionality of their IP.

1.12 References

The Model details are based upon the following specifications:

RISC-V Instruction Set Manual, Volume I: User-Level ISA (User Architecture Version 20190305-Base-Ratification)

RISC-V Instruction Set Manual, Volume II: Privileged Architecture (Privileged Architecture Version 20190405-Priv-MSU-Ratification)

RISC-V “V” Vector Extension (Vector Architecture Version 0.8)

Chapter 2

Configuration

2.1 Location

This model's VLNv is riscv.ovpworld.org/processor/riscv/1.0.

The model source is usually at:

`$IMPERAS_HOME/ImperasLib/source/riscv.ovpworld.org/processor/riscv/1.0`

The model binary is usually at:

`$IMPERAS_HOME/lib/$IMPERAS_ARCH/ImperasLib/riscv.ovpworld.org/processor/riscv/1.0`

2.2 GDB Path

The default GDB for this model is: `$IMPERAS_HOME/lib/$IMPERAS_ARCH/gdb/riscv-none-embed-gdb`.

2.3 Semi-Host Library

The default semi-host library file is riscv.ovpworld.org/semihosting/pk/1.0

2.4 Processor Endian-ness

This is a LITTLE endian model.

2.5 QuantumLeap Support

This processor is qualified to run in a QuantumLeap enabled simulator.

2.6 Processor ELF code

The ELF code supported by this model is: 0xf3.

Chapter 3

All Variants in this model

This model has these variants

| Variant | Description |
|----------|------------------------------|
| RV32I | |
| RV32IM | |
| RV32IMC | |
| RV32IMAC | |
| RV32G | |
| RV32GC | |
| RV32GCN | |
| RV32GCV | |
| RV32E | |
| RV32EC | |
| RV64I | |
| RV64IM | |
| RV64IMC | |
| RV64IMAC | |
| RV64G | |
| RV64GC | |
| RV64GCN | |
| RV64GCV | (described in this document) |
| RVB32I | |
| RVB32E | |
| RVB64I | |

Table 3.1: All Variants in this model

Chapter 4

Bus Master Ports

This model has these bus master ports.

| Name | min | max | Connect? | Description |
|-------------|-----|-----|-----------|-----------------|
| INSTRUCTION | 32 | 56 | mandatory | Instruction bus |
| DATA | 32 | 56 | optional | Data bus |

Table 4.1: Bus Master Ports

Chapter 5

Bus Slave Ports

This model has no bus slave ports.

Chapter 6

Net Ports

This model has these net ports.

| Name | Type | Connect? | Description |
|--------------------|-------|----------|-------------------------------|
| reset | input | optional | Reset |
| nmi | input | optional | NMI |
| SSWInterrupt | input | optional | Supervisor software interrupt |
| MSWInterrupt | input | optional | Machine software interrupt |
| STimerInterrupt | input | optional | Supervisor timer interrupt |
| MTimerInterrupt | input | optional | Machine timer interrupt |
| SExternalInterrupt | input | optional | Supervisor external interrupt |
| MExternalInterrupt | input | optional | Machine external interrupt |

Table 6.1: Net Ports

Chapter 7

FIFO Ports

This model has no FIFO ports.

Chapter 8

Formal Parameters

| Name | Type | Description |
|-------------------|-------------|---|
| variant | Enumeration | Selects variant (either a generic UISA or a specific model) |
| user_version | Enumeration | Specify required User Architecture version (2.2, 2.3 or 20190305) |
| priv_version | Enumeration | Specify required Privileged Architecture version (1.10, 1.11 or 20190405) |
| vector_version | Enumeration | Specify required Vector Architecture version (0.7.1-draft-20190605, 0.7.1-draft-20190605+, 0.8-draft-20190906, 0.8-draft-20191004, 0.8-draft-20191117, 0.8-draft-20191118, 0.8 or master) |
| fp16_version | Enumeration | Specify required 16-bit floating point format (none, IEEE754 or BFLOAT16) |
| mstatus_fs_mode | Enumeration | Specify conditions causing update of mstatus.FS to dirty (write_1, write_any or always_dirty) |
| verbose | Boolean | Specify verbose output messages |
| updatePTEA | Boolean | Specify whether hardware update of PTE A bit is supported |
| updatePTED | Boolean | Specify whether hardware update of PTE D bit is supported |
| unaligned | Boolean | Specify whether the processor supports unaligned memory accesses |
| unalignedAMO | Boolean | Specify whether the processor supports unaligned memory accesses for AMO instructions |
| wfi_is_nop | Boolean | Specify whether WFI should be treated as a NOP (if not, halt while waiting for interrupts) |
| mtvec_is_ro | Boolean | Specify whether mtvec CSR is read-only |
| tvec_align | Uns32 | Specify hardware-enforced alignment of mtvec/stvec/utvec when Vectored interrupt mode enabled |
| mtvec_mask | Uns64 | Specify hardware-enforced mask of writable bits in mtvec register |
| stvec_mask | Uns64 | Specify hardware-enforced mask of writable bits in stvec register |
| tval_ii_code | Boolean | Specify whether mtval/stval contain faulting instruction bits on illegal instruction exception |
| cycle_undefined | Boolean | Specify that the cycle CSR is undefined (reads to it are emulated by a Machine mode trap) |
| time_undefined | Boolean | Specify that the time CSR is undefined (reads to it are emulated by a Machine mode trap) |
| instret_undefined | Boolean | Specify that the instret CSR is undefined (reads to it are emulated by a Machine mode trap) |
| enable_CSR_bus | Boolean | Add artifact CSR bus port, allowing CSR registers to be externally implemented |
| d_requires_f | Boolean | If D and F extensions are separately enabled in the misa CSR, whether D is enabled only if F is enabled |
| xret_preserves_lr | Boolean | Whether an xRET instruction preserves the value of LR |
| require_vstart0 | Boolean | Whether CSR vstart must be 0 for non-interruptible vector instructions |
| ASID.bits | Uns32 | Specify the number of implemented ASID bits |
| lr_sc_grain | Uns32 | Specify byte granularity of ll/sc lock region (constrained to a power of two) |
| reset_address | Uns64 | Override reset vector address |
| nmi_address | Uns64 | Override NMI vector address |

| | | |
|----------------------|---------|---|
| PMP_grain | Uns32 | Specify PMP region granularity, G (0 =>4 bytes, 1 =>8 bytes, etc) |
| PMP_registers | Uns32 | Specify the number of implemented PMP address registers |
| Sv_modes | Uns32 | Specify bit mask of implemented Sv modes (e.g. 1<<8 is Sv39) |
| local_int_num | Uns32 | Specify number of supplemental local interrupts |
| endian | Endian | Model endian |
| misa_MXL | Uns32 | Override default value of misa.MXL |
| misa_MXL_mask | Uns32 | Override mask of writable bits in misa.MXL |
| misa_Extensions | Uns32 | Override default value of misa.Extensions |
| add_Extensions | String | Add extensions specified by letters to misa.Extensions (for example, specify “VD” to add V and D features) |
| misa_Extensions_mask | Uns32 | Override mask of writable bits in misa.Extensions |
| add_Extensions_mask | String | Add extensions specified by letters to mask of writable bits in misa.Extensions (for example, specify “VD” to add V and D features) |
| mvendorid | Uns64 | Override mvendorid register |
| marchid | Uns64 | Override marchid register |
| mimpid | Uns64 | Override mimpid register |
| mhartid | Uns64 | Override mhartid register |
| mtvec | Uns64 | Override mtvec register |
| mstatus_FS | Uns32 | Override default value of mstatus.FS (initial state of floating point unit) |
| ELEN | Uns32 | Override ELEN (vector extension) |
| SLEN | Uns32 | Override SLEN (vector extension) |
| VLEN | Uns32 | Override VLEN (vector extension) |
| Zvlsseg | Boolean | Specify that Zvlsseg is implemented (vector extension) |
| Zvamo | Boolean | Specify that Zvamo is implemented (vector extension) |
| Zvediv | Boolean | Specify that Zvediv is implemented (vector extension) |
| Zvqmac | Boolean | Specify that Zvqmac is implemented (vector extension) |

Table 8.1: Parameters that can be set in: Hart

8.1 Parameters with enumerated types

8.1.1 Parameter user_version

| Set to this value | Description |
|-------------------|--|
| 2.2 | User Architecture Version 2.2 |
| 2.3 | Deprecated and equivalent to 20190305 |
| 20190305 | User Architecture Version 20190305-Base-Ratification |

Table 8.2: Values for Parameter user_version

8.1.2 Parameter priv_version

| Set to this value | Description |
|-------------------|--|
| 1.10 | Privileged Architecture Version 1.10 |
| 1.11 | Deprecated and equivalent to 20190405 |
| 20190405 | Privileged Architecture Version 20190405-Priv-MSU-Ratification |

Table 8.3: Values for Parameter priv_version

8.1.3 Parameter vector_version

| Set to this value | Description |
|----------------------|--|
| 0.7.1-draft-20190605 | Vector Architecture Version 0.7.1-draft-20190605 |

| | |
|-----------------------|---|
| 0.7.1-draft-20190605+ | Vector Architecture Version 0.7.1-draft-20190605 with custom features (not for general use) |
| 0.8-draft-20190906 | Vector Architecture Version 0.8-draft-20190906 |
| 0.8-draft-20191004 | Vector Architecture Version 0.8-draft-20191004 |
| 0.8-draft-20191117 | Vector Architecture Version 0.8-draft-20191117 |
| 0.8-draft-20191118 | Vector Architecture Version 0.8-draft-20191118 |
| 0.8 | Vector Architecture Version 0.8 |
| master | Vector Architecture Master Branch as of commit 9a65519 (this is subject to change) |

Table 8.4: Values for Parameter vector_version

8.1.4 Parameter fp16_version

| Set to this value | Description |
|-------------------|--------------------------------------|
| none | No 16-bit floating point implemented |
| IEEE754 | IEEE 754 half precision implemented |
| BFLOAT16 | BFLOAT16 implemented |

Table 8.5: Values for Parameter fp16_version

8.1.5 Parameter mstatus_fs_mode

| Set to this value | Description |
|-------------------|--|
| write_1 | Any non-zero flag result sets mstatus.fs dirty |
| write_any | Any write of flags sets mstatus.fs dirty |
| always_dirty | mstatus.fs is either off or dirty |

Table 8.6: Values for Parameter mstatus_fs_mode

Chapter 9

Execution Modes

| Mode | Code | Description |
|------------|------|-----------------|
| User | 0 | User mode |
| Supervisor | 1 | Supervisor mode |
| Machine | 3 | Machine mode |

Table 9.1: Modes implemented in: Hart

Chapter 10

Exceptions

| Exception | Code | Description |
|------------------------------|------|--|
| InstructionAddressMisaligned | 0 | Fetch from unaligned address |
| InstructionAccessFault | 1 | No access permission for fetch |
| IllegalInstruction | 2 | Undecoded, unimplemented or disabled instruction |
| Breakpoint | 3 | EBREAK instruction executed |
| LoadAddressMisaligned | 4 | Load from unaligned address |
| LoadAccessFault | 5 | No access permission for load |
| StoreAMOAddressMisaligned | 6 | Store/atomic memory operation at unaligned address |
| StoreAMOAccessFault | 7 | No access permission for store/atomic memory operation |
| EnvironmentCallFromUMode | 8 | ECALL instruction executed in User mode |
| EnvironmentCallFromSMode | 9 | ECALL instruction executed in Supervisor mode |
| EnvironmentCallFromMMode | 11 | ECALL instruction executed in Machine mode |
| InstructionPageFault | 12 | Page fault at fetch address |
| LoadPageFault | 13 | Page fault at load address |
| StoreAMOPageFault | 15 | Page fault at store/atomic memory operation address |
| SSWInterrupt | 65 | Supervisor software interrupt |
| MSWInterrupt | 67 | Machine software interrupt |
| STimerInterrupt | 69 | Supervisor timer interrupt |
| MTimerInterrupt | 71 | Machine timer interrupt |
| SExternalInterrupt | 73 | Supervisor external interrupt |
| MExternalInterrupt | 75 | Machine external interrupt |

Table 10.1: Exceptions implemented in: Hart

Chapter 11

Hierarchy of the model

A CPU core may be configured to instance many processors of a Symmetrical Multi Processor (SMP). A CPU core may also have sub elements within a processor, for example hardware threading blocks.

OVP processor models can be written to include SMP blocks and to have many levels of hierarchy. Some OVP CPU models may have a fixed hierarchy, and some may be configured by settings in a configuration register. Please see the register definitions of this model.

This model documentation shows the settings and hierarchy of the default settings for this model variant.

11.1 Level 1: Hart

This level in the model hierarchy has 3 commands.

This level in the model hierarchy has 7 register groups:

| Group name | Registers |
|-------------------------------|-----------|
| Core | 33 |
| Floating_point | 32 |
| Vector | 32 |
| User_Control_and_Status | 41 |
| Supervisor_Control_and_Status | 10 |
| Machine_Control_and_Status | 102 |
| Integration_support | 1 |

Table 11.1: Register groups

This level in the model hierarchy has no children.

Chapter 12

Model Commands

A Processor model can implement one or more **Model Commands** available to be invoked from the simulator command line, from the OP API or from the Imperas Multiprocessor Debugger.

12.1 Level 1: Hart

12.1.1 dumpTLB

12.1.1.1 Argument description

show TLB contents

12.1.2 isync

specify instruction address range for synchronous execution

| Argument | Type | Description |
|------------|-------|--|
| -addresshi | Uns64 | end address of synchronous execution range |
| -addresslo | Uns64 | start address of synchronous execution range |

Table 12.1: isync command arguments

12.1.3 itrace

enable or disable instruction tracing

| Argument | Type | Description |
|-------------------|---------|--|
| -after | Uns64 | apply after this many instructions |
| -enable | Boolean | enable instruction tracing |
| -instructioncount | Boolean | include the instruction number in each trace |
| -off | Boolean | disable instruction tracing |
| -on | Boolean | enable instruction tracing |
| -registerchange | Boolean | show registers changed by this instruction |
| -registers | Boolean | show registers after each trace |

Table 12.2: itrace command arguments

Chapter 13

Registers

13.1 Level 1: Hart

13.1.1 Core

Registers at level:1, type:Hart group:Core

| Name | Bits | Initial-Hex | RW | Description |
|------|------|-------------|----|-----------------|
| zero | 64 | 0 | r- | |
| ra | 64 | 0 | rw | |
| sp | 64 | 0 | rw | stack pointer |
| gp | 64 | 0 | rw | |
| tp | 64 | 0 | rw | |
| t0 | 64 | 0 | rw | |
| t1 | 64 | 0 | rw | |
| t2 | 64 | 0 | rw | |
| s0 | 64 | 0 | rw | |
| s1 | 64 | 0 | rw | |
| a0 | 64 | 0 | rw | |
| a1 | 64 | 0 | rw | |
| a2 | 64 | 0 | rw | |
| a3 | 64 | 0 | rw | |
| a4 | 64 | 0 | rw | |
| a5 | 64 | 0 | rw | |
| a6 | 64 | 0 | rw | |
| a7 | 64 | 0 | rw | |
| s2 | 64 | 0 | rw | |
| s3 | 64 | 0 | rw | |
| s4 | 64 | 0 | rw | |
| s5 | 64 | 0 | rw | |
| s6 | 64 | 0 | rw | |
| s7 | 64 | 0 | rw | |
| s8 | 64 | 0 | rw | |
| s9 | 64 | 0 | rw | |
| s10 | 64 | 0 | rw | |
| s11 | 64 | 0 | rw | |
| t3 | 64 | 0 | rw | |
| t4 | 64 | 0 | rw | |
| t5 | 64 | 0 | rw | |
| t6 | 64 | 0 | rw | |
| pc | 64 | 0 | rw | program counter |

Table 13.1: Registers at level 1, type:Hart group:Core

13.1.2 Floating_point

Registers at level:1, type:Hart group:Floating_point

| Name | Bits | Initial-Hex | RW | Description |
|------|------|-------------|----|-------------|
| ft0 | 64 | 0 | rw | |
| ft1 | 64 | 0 | rw | |
| ft2 | 64 | 0 | rw | |
| ft3 | 64 | 0 | rw | |
| ft4 | 64 | 0 | rw | |
| ft5 | 64 | 0 | rw | |
| ft6 | 64 | 0 | rw | |
| ft7 | 64 | 0 | rw | |
| fs0 | 64 | 0 | rw | |
| fs1 | 64 | 0 | rw | |
| fa0 | 64 | 0 | rw | |
| fa1 | 64 | 0 | rw | |
| fa2 | 64 | 0 | rw | |
| fa3 | 64 | 0 | rw | |
| fa4 | 64 | 0 | rw | |
| fa5 | 64 | 0 | rw | |
| fa6 | 64 | 0 | rw | |
| fa7 | 64 | 0 | rw | |
| fs2 | 64 | 0 | rw | |
| fs3 | 64 | 0 | rw | |
| fs4 | 64 | 0 | rw | |
| fs5 | 64 | 0 | rw | |
| fs6 | 64 | 0 | rw | |
| fs7 | 64 | 0 | rw | |
| fs8 | 64 | 0 | rw | |
| fs9 | 64 | 0 | rw | |
| fs10 | 64 | 0 | rw | |
| fs11 | 64 | 0 | rw | |
| ft8 | 64 | 0 | rw | |
| ft9 | 64 | 0 | rw | |
| ft10 | 64 | 0 | rw | |
| ft11 | 64 | 0 | rw | |

Table 13.2: Registers at level 1, type:Hart group:Floating_point

13.1.3 Vector

Registers at level:1, type:Hart group:Vector

| Name | Bits | Initial-Hex | RW | Description |
|------|------|-------------|----|-------------|
| v0 | 512 | - | rw | |
| v1 | 512 | - | rw | |
| v2 | 512 | - | rw | |
| v3 | 512 | - | rw | |
| v4 | 512 | - | rw | |
| v5 | 512 | - | rw | |
| v6 | 512 | - | rw | |

| | | | | |
|-----|-----|---|----|--|
| v7 | 512 | - | rw | |
| v8 | 512 | - | rw | |
| v9 | 512 | - | rw | |
| v10 | 512 | - | rw | |
| v11 | 512 | - | rw | |
| v12 | 512 | - | rw | |
| v13 | 512 | - | rw | |
| v14 | 512 | - | rw | |
| v15 | 512 | - | rw | |
| v16 | 512 | - | rw | |
| v17 | 512 | - | rw | |
| v18 | 512 | - | rw | |
| v19 | 512 | - | rw | |
| v20 | 512 | - | rw | |
| v21 | 512 | - | rw | |
| v22 | 512 | - | rw | |
| v23 | 512 | - | rw | |
| v24 | 512 | - | rw | |
| v25 | 512 | - | rw | |
| v26 | 512 | - | rw | |
| v27 | 512 | - | rw | |
| v28 | 512 | - | rw | |
| v29 | 512 | - | rw | |
| v30 | 512 | - | rw | |
| v31 | 512 | - | rw | |

Table 13.3: Registers at level 1, type:Hart group:Vector

13.1.4 User_Control_and_Status

Registers at level:1, type:Hart group:User_Control_and_Status

| Name | Bits | Initial-Hex | RW | Description |
|--------------|------|-------------|----|-----------------------------------|
| fflags | 64 | 0 | rw | Floating-Point Flags |
| frm | 64 | 0 | rw | Floating-Point Rounding Mode |
| fcsr | 64 | 0 | rw | Floating-Point Control and Status |
| vstart | 64 | 0 | rw | Vector Start Index |
| vxsat | 64 | 0 | rw | Fixed-Point Saturate Flag |
| vxrm | 64 | 0 | rw | Fixed-Point Rounding Mode |
| cycle | 64 | 0 | r- | Cycle Counter |
| time | 64 | 0 | r- | Timer |
| instret | 64 | 0 | r- | Instructions Retired |
| hpmcounter3 | 64 | 0 | r- | Performance Monitor Counter 3 |
| hpmcounter4 | 64 | 0 | r- | Performance Monitor Counter 4 |
| hpmcounter5 | 64 | 0 | r- | Performance Monitor Counter 5 |
| hpmcounter6 | 64 | 0 | r- | Performance Monitor Counter 6 |
| hpmcounter7 | 64 | 0 | r- | Performance Monitor Counter 7 |
| hpmcounter8 | 64 | 0 | r- | Performance Monitor Counter 8 |
| hpmcounter9 | 64 | 0 | r- | Performance Monitor Counter 9 |
| hpmcounter10 | 64 | 0 | r- | Performance Monitor Counter 10 |
| hpmcounter11 | 64 | 0 | r- | Performance Monitor Counter 11 |
| hpmcounter12 | 64 | 0 | r- | Performance Monitor Counter 12 |
| hpmcounter13 | 64 | 0 | r- | Performance Monitor Counter 13 |
| hpmcounter14 | 64 | 0 | r- | Performance Monitor Counter 14 |
| hpmcounter15 | 64 | 0 | r- | Performance Monitor Counter 15 |
| hpmcounter16 | 64 | 0 | r- | Performance Monitor Counter 16 |

| | | | | |
|--------------|----|----|----|--------------------------------|
| hpmcounter17 | 64 | 0 | r- | Performance Monitor Counter 17 |
| hpmcounter18 | 64 | 0 | r- | Performance Monitor Counter 18 |
| hpmcounter19 | 64 | 0 | r- | Performance Monitor Counter 19 |
| hpmcounter20 | 64 | 0 | r- | Performance Monitor Counter 20 |
| hpmcounter21 | 64 | 0 | r- | Performance Monitor Counter 21 |
| hpmcounter22 | 64 | 0 | r- | Performance Monitor Counter 22 |
| hpmcounter23 | 64 | 0 | r- | Performance Monitor Counter 23 |
| hpmcounter24 | 64 | 0 | r- | Performance Monitor Counter 24 |
| hpmcounter25 | 64 | 0 | r- | Performance Monitor Counter 25 |
| hpmcounter26 | 64 | 0 | r- | Performance Monitor Counter 26 |
| hpmcounter27 | 64 | 0 | r- | Performance Monitor Counter 27 |
| hpmcounter28 | 64 | 0 | r- | Performance Monitor Counter 28 |
| hpmcounter29 | 64 | 0 | r- | Performance Monitor Counter 29 |
| hpmcounter30 | 64 | 0 | r- | Performance Monitor Counter 30 |
| hpmcounter31 | 64 | 0 | r- | Performance Monitor Counter 31 |
| vl | 64 | 0 | r- | Vector Length |
| vtype | 64 | 0 | r- | Vector Type |
| vlenb | 64 | 40 | r- | Vector Length in Bytes |

Table 13.4: Registers at level 1, type:Hart group:User_Control_and_Status

13.1.5 Supervisor_Control_and_Status

Registers at level:1, type:Hart group:Supervisor_Control_and_Status

| Name | Bits | Initial-Hex | RW | Description |
|------------|------|-------------|----|---|
| sstatus | 64 | 2 00000000 | rw | Supervisor Status |
| sie | 64 | 0 | rw | Supervisor Interrupt Enable |
| stvec | 64 | 0 | rw | Supervisor Trap-Vector Base-Address |
| scounteren | 64 | 0 | rw | Supervisor Counter Enable |
| sscratch | 64 | 0 | rw | Supervisor Scratch |
| sepc | 64 | 0 | rw | Supervisor Exception Program Counter |
| scause | 64 | 0 | rw | Supervisor Cause |
| stval | 64 | 0 | rw | Supervisor Trap Value |
| sip | 64 | 0 | rw | Supervisor Interrupt Pending |
| satp | 64 | 0 | rw | Supervisor Address Translation and Protection |

Table 13.5: Registers at level 1, type:Hart group:Supervisor_Control_and_Status

13.1.6 Machine_Control_and_Status

Registers at level:1, type:Hart group:Machine_Control_and_Status

| Name | Bits | Initial-Hex | RW | Description |
|---------------|------|----------------------|----|--|
| mstatus | 64 | a 00000000 | rw | Machine Status |
| misa | 64 | 80000000 0034112d | rw | ISA and Extensions |
| medeleg | 64 | 0 | rw | Machine Exception Delegation |
| mideleg | 64 | 0 | rw | Machine Interrupt Delegation |
| mie | 64 | 0 | rw | Machine Interrupt Enable |
| mtvec | 64 | 0 | rw | Machine Trap-Vector Base-Address |
| mcounteren | 64 | 0 | rw | Machine Counter Enable |
| mcountinhibit | 64 | 0 | rw | Machine Counter Inhibit |
| mhpmevent3 | 64 | 0 | rw | Machine Performance Monitor Event Select 3 |
| mhpmevent4 | 64 | 0 | rw | Machine Performance Monitor Event Select 4 |

| | | | | |
|-------------|----|---|----|---|
| mhpmevent5 | 64 | 0 | rw | Machine Performance Monitor Event Select 5 |
| mhpmevent6 | 64 | 0 | rw | Machine Performance Monitor Event Select 6 |
| mhpmevent7 | 64 | 0 | rw | Machine Performance Monitor Event Select 7 |
| mhpmevent8 | 64 | 0 | rw | Machine Performance Monitor Event Select 8 |
| mhpmevent9 | 64 | 0 | rw | Machine Performance Monitor Event Select 9 |
| mhpmevent10 | 64 | 0 | rw | Machine Performance Monitor Event Select 10 |
| mhpmevent11 | 64 | 0 | rw | Machine Performance Monitor Event Select 11 |
| mhpmevent12 | 64 | 0 | rw | Machine Performance Monitor Event Select 12 |
| mhpmevent13 | 64 | 0 | rw | Machine Performance Monitor Event Select 13 |
| mhpmevent14 | 64 | 0 | rw | Machine Performance Monitor Event Select 14 |
| mhpmevent15 | 64 | 0 | rw | Machine Performance Monitor Event Select 15 |
| mhpmevent16 | 64 | 0 | rw | Machine Performance Monitor Event Select 16 |
| mhpmevent17 | 64 | 0 | rw | Machine Performance Monitor Event Select 17 |
| mhpmevent18 | 64 | 0 | rw | Machine Performance Monitor Event Select 18 |
| mhpmevent19 | 64 | 0 | rw | Machine Performance Monitor Event Select 19 |
| mhpmevent20 | 64 | 0 | rw | Machine Performance Monitor Event Select 20 |
| mhpmevent21 | 64 | 0 | rw | Machine Performance Monitor Event Select 21 |
| mhpmevent22 | 64 | 0 | rw | Machine Performance Monitor Event Select 22 |
| mhpmevent23 | 64 | 0 | rw | Machine Performance Monitor Event Select 23 |
| mhpmevent24 | 64 | 0 | rw | Machine Performance Monitor Event Select 24 |
| mhpmevent25 | 64 | 0 | rw | Machine Performance Monitor Event Select 25 |
| mhpmevent26 | 64 | 0 | rw | Machine Performance Monitor Event Select 26 |
| mhpmevent27 | 64 | 0 | rw | Machine Performance Monitor Event Select 27 |
| mhpmevent28 | 64 | 0 | rw | Machine Performance Monitor Event Select 28 |
| mhpmevent29 | 64 | 0 | rw | Machine Performance Monitor Event Select 29 |
| mhpmevent30 | 64 | 0 | rw | Machine Performance Monitor Event Select 30 |
| mhpmevent31 | 64 | 0 | rw | Machine Performance Monitor Event Select 31 |
| mscratch | 64 | 0 | rw | Machine Scratch |
| mepc | 64 | 0 | rw | Machine Exception Program Counter |
| mcause | 64 | 0 | rw | Machine Cause |
| mtval | 64 | 0 | rw | Machine Trap Value |
| mip | 64 | 0 | rw | Machine Interrupt Pending |
| pmpcfg0 | 64 | 0 | rw | Physical Memory Protection Configuration 0 |
| pmpcfg2 | 64 | 0 | rw | Physical Memory Protection Configuration 2 |
| pmpaddr0 | 64 | 0 | rw | Physical Memory Protection Address 0 |
| pmpaddr1 | 64 | 0 | rw | Physical Memory Protection Address 1 |
| pmpaddr2 | 64 | 0 | rw | Physical Memory Protection Address 2 |
| pmpaddr3 | 64 | 0 | rw | Physical Memory Protection Address 3 |
| pmpaddr4 | 64 | 0 | rw | Physical Memory Protection Address 4 |
| pmpaddr5 | 64 | 0 | rw | Physical Memory Protection Address 5 |
| pmpaddr6 | 64 | 0 | rw | Physical Memory Protection Address 6 |
| pmpaddr7 | 64 | 0 | rw | Physical Memory Protection Address 7 |
| pmpaddr8 | 64 | 0 | rw | Physical Memory Protection Address 8 |
| pmpaddr9 | 64 | 0 | rw | Physical Memory Protection Address 9 |
| pmpaddr10 | 64 | 0 | rw | Physical Memory Protection Address 10 |
| pmpaddr11 | 64 | 0 | rw | Physical Memory Protection Address 11 |
| pmpaddr12 | 64 | 0 | rw | Physical Memory Protection Address 12 |
| pmpaddr13 | 64 | 0 | rw | Physical Memory Protection Address 13 |
| pmpaddr14 | 64 | 0 | rw | Physical Memory Protection Address 14 |
| pmpaddr15 | 64 | 0 | rw | Physical Memory Protection Address 15 |
| tselect | 64 | - | rw | Debug/Trace Trigger Register Select (not implemented) |
| tdata1 | 64 | - | rw | Debug/Trace Trigger Data 1 (not implemented) |
| tdata2 | 64 | - | rw | Debug/Trace Trigger Data 2 (not implemented) |
| tdata3 | 64 | - | rw | Debug/Trace Trigger Data 3 (not implemented) |
| dcsr | 64 | - | rw | Debug Control and Status (not implemented) |
| dpc | 64 | - | rw | Debug PC (not implemented) |

| | | | | |
|---------------|----|---|----|--|
| dscratch | 64 | - | rw | Debug Scratch (not implemented) |
| mcycle | 64 | 0 | rw | Machine Cycle Counter |
| minstret | 64 | 0 | rw | Machine Instructions Retired |
| mhpmcounter3 | 64 | 0 | rw | Machine Performance Monitor Counter 3 |
| mhpmcounter4 | 64 | 0 | rw | Machine Performance Monitor Counter 4 |
| mhpmcounter5 | 64 | 0 | rw | Machine Performance Monitor Counter 5 |
| mhpmcounter6 | 64 | 0 | rw | Machine Performance Monitor Counter 6 |
| mhpmcounter7 | 64 | 0 | rw | Machine Performance Monitor Counter 7 |
| mhpmcounter8 | 64 | 0 | rw | Machine Performance Monitor Counter 8 |
| mhpmcounter9 | 64 | 0 | rw | Machine Performance Monitor Counter 9 |
| mhpmcounter10 | 64 | 0 | rw | Machine Performance Monitor Counter 10 |
| mhpmcounter11 | 64 | 0 | rw | Machine Performance Monitor Counter 11 |
| mhpmcounter12 | 64 | 0 | rw | Machine Performance Monitor Counter 12 |
| mhpmcounter13 | 64 | 0 | rw | Machine Performance Monitor Counter 13 |
| mhpmcounter14 | 64 | 0 | rw | Machine Performance Monitor Counter 14 |
| mhpmcounter15 | 64 | 0 | rw | Machine Performance Monitor Counter 15 |
| mhpmcounter16 | 64 | 0 | rw | Machine Performance Monitor Counter 16 |
| mhpmcounter17 | 64 | 0 | rw | Machine Performance Monitor Counter 17 |
| mhpmcounter18 | 64 | 0 | rw | Machine Performance Monitor Counter 18 |
| mhpmcounter19 | 64 | 0 | rw | Machine Performance Monitor Counter 19 |
| mhpmcounter20 | 64 | 0 | rw | Machine Performance Monitor Counter 20 |
| mhpmcounter21 | 64 | 0 | rw | Machine Performance Monitor Counter 21 |
| mhpmcounter22 | 64 | 0 | rw | Machine Performance Monitor Counter 22 |
| mhpmcounter23 | 64 | 0 | rw | Machine Performance Monitor Counter 23 |
| mhpmcounter24 | 64 | 0 | rw | Machine Performance Monitor Counter 24 |
| mhpmcounter25 | 64 | 0 | rw | Machine Performance Monitor Counter 25 |
| mhpmcounter26 | 64 | 0 | rw | Machine Performance Monitor Counter 26 |
| mhpmcounter27 | 64 | 0 | rw | Machine Performance Monitor Counter 27 |
| mhpmcounter28 | 64 | 0 | rw | Machine Performance Monitor Counter 28 |
| mhpmcounter29 | 64 | 0 | rw | Machine Performance Monitor Counter 29 |
| mhpmcounter30 | 64 | 0 | rw | Machine Performance Monitor Counter 30 |
| mhpmcounter31 | 64 | 0 | rw | Machine Performance Monitor Counter 31 |
| mvendorid | 64 | 0 | r- | Vendor ID |
| marchid | 64 | 0 | r- | Architecture ID |
| mimpid | 64 | 0 | r- | Implementation ID |
| mhartid | 64 | 0 | r- | Hardware Thread ID |

Table 13.6: Registers at level 1, type:Hart group:Machine.Control.and.Status

13.1.7 Integration_support

Registers at level:1, type:Hart group:Integration_support

| Name | Bits | Initial-Hex | RW | Description |
|-------------|------|---------------|----|---------------------------|
| LRSCAddress | 64 | ffffff ffffff | rw | LR/SC active lock address |

Table 13.7: Registers at level 1, type:Hart group:Integration_support