



OVP Guide to Using Processor Models

Model specific information for RISC-V_RV32GCP

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



| | |
|----------|--|
| Author | Imperas Software Limited |
| Version | 20240902.0 |
| Filename | OVP_Model_Specific_Information_riscv_RV32GCP.pdf |
| Created | 2 September 2024 |
| Status | OVP Standard Release |

Copyright Notice

Copyright (c) 2024 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 | 2 |
| 1.3.1 | Extensions Enabled by Default | 2 |
| 1.3.2 | Enabling Other Extensions | 2 |
| 1.3.3 | Disabling Extensions | 3 |
| 1.4 | General Features | 3 |
| 1.4.1 | mtvec CSR | 3 |
| 1.4.2 | stvec CSR | 4 |
| 1.4.3 | Reset | 4 |
| 1.4.4 | NMI | 4 |
| 1.4.5 | WFI | 5 |
| 1.4.6 | cycle CSR | 5 |
| 1.4.7 | instret CSR | 5 |
| 1.4.8 | hpmcounter CSR | 5 |
| 1.4.9 | time CSR | 5 |
| 1.4.10 | mcycle CSR | 6 |
| 1.4.11 | minstret CSR | 6 |
| 1.4.12 | mhpmcounter CSR | 6 |
| 1.4.13 | Virtual Memory | 6 |
| 1.4.14 | Unaligned Accesses | 6 |
| 1.4.15 | PMP | 7 |
| 1.4.16 | Time and Timers | 7 |
| 1.5 | Compressed Extension | 7 |
| 1.5.1 | Compressed Extension Parameters | 8 |
| 1.5.2 | Legacy Version 1.10 | 8 |
| 1.5.3 | Version 0.70.1 | 8 |
| 1.5.4 | Version 0.70.5 | 8 |
| 1.5.5 | Version 1.0.0-RC5.7 | 8 |
| 1.5.6 | Version 1.0 | 9 |
| 1.6 | Floating Point Features | 9 |
| 1.7 | Privileged Architecture | 10 |
| 1.7.1 | Legacy Version 1.10 | 10 |
| 1.7.2 | Version 20190608 | 10 |
| 1.7.3 | Version 20211203 | 10 |
| 1.7.4 | Version 1.12 | 10 |

| | | |
|---------|---|----|
| 1.7.5 | Version master | 10 |
| 1.8 | Unprivileged Architecture | 11 |
| 1.8.1 | Legacy Version 2.2 | 11 |
| 1.8.2 | Version 20191213 | 11 |
| 1.9 | DSP Extension | 11 |
| 1.9.1 | DSP Extension Parameters | 11 |
| 1.9.2 | DSP Extension Versions | 11 |
| 1.9.3 | Version 0.5.2 | 11 |
| 1.9.4 | Version 0.9.6 | 12 |
| 1.10 | Other Extensions | 12 |
| 1.10.1 | Zmmul | 12 |
| 1.10.2 | Zicsr | 12 |
| 1.10.3 | Zifencei | 12 |
| 1.10.4 | Zicbom | 13 |
| 1.10.5 | Zicbop | 13 |
| 1.10.6 | Zicboz | 13 |
| 1.10.7 | Smstateen | 13 |
| 1.10.8 | Sscofpmf | 13 |
| 1.10.9 | Smcentrpfm | 14 |
| 1.10.10 | Smcsrind | 14 |
| 1.10.11 | Smcdeleg | 14 |
| 1.10.12 | Ssqosid | 14 |
| 1.10.13 | Zawrs | 14 |
| 1.10.14 | Zimop | 14 |
| 1.10.15 | Zcmop | 14 |
| 1.11 | CLIC | 15 |
| 1.12 | Advanced Interrupt Architecture | 15 |
| 1.13 | Load-Reserved/Store-Conditional Locking | 15 |
| 1.14 | Active Atomic Operation Indication | 16 |
| 1.15 | Interrupts | 16 |
| 1.16 | Debug Mode (Sdext) | 17 |
| 1.16.1 | Debug State Entry | 18 |
| 1.16.2 | Debug State Exit | 18 |
| 1.16.3 | Debug Registers | 18 |
| 1.16.4 | Debug Mode Execution | 19 |
| 1.16.5 | Debug Single Step | 19 |
| 1.16.6 | Debug Event Priorities | 19 |
| 1.16.7 | Debug Ports | 20 |
| 1.16.8 | Debug Mode Versions | 20 |
| 1.16.9 | Version 0.13.2-DRAFT | 20 |
| 1.16.10 | Version 0.14.0-DRAFT | 20 |
| 1.16.11 | Version 1.0.0-STABLE | 20 |
| 1.16.12 | Version 1.0-STABLE | 20 |
| 1.17 | Trigger Module (Sdtrig) | 20 |
| 1.17.1 | Trigger Module Restrictions | 20 |
| 1.17.2 | Trigger Module Parameters | 21 |
| 1.18 | Debug Mask | 22 |
| 1.19 | Integration Support | 22 |

| | | |
|----------|--|-----------|
| 1.19.1 | Command “setPMA -attributes <attrs>-lo <addr>-hi <addr>” | 22 |
| 1.19.2 | Command “getCSRIndex -name <name>” | 23 |
| 1.19.3 | Command “listCSRs” | 23 |
| 1.19.4 | CSR Register External Implementation | 23 |
| 1.19.5 | LR/SC Active Address | 24 |
| 1.19.6 | Page Table Walk Introspection | 24 |
| 1.19.7 | Page Table Walk Query | 24 |
| 1.19.8 | Artifact Page Table Walks | 25 |
| 1.19.9 | Page Table Walk Event | 25 |
| 1.19.10 | Artifact Register “fflags_i” | 25 |
| 1.19.11 | External Stimulation of Illegal Instruction Trap | 25 |
| 1.20 | Instruction Disassembly | 25 |
| 1.21 | Limitations | 26 |
| 1.22 | Verification | 26 |
| 1.23 | References | 26 |
| 2 | Configuration | 28 |
| 2.1 | Location | 28 |
| 2.2 | GDB Path | 28 |
| 2.3 | Semi-Host Library | 28 |
| 2.4 | Processor Endian-ness | 28 |
| 2.5 | QuantumLeap Support | 28 |
| 2.6 | Processor ELF code | 28 |
| 3 | All Variants in this model | 29 |
| 4 | Bus Master Ports | 31 |
| 5 | Bus Slave Ports | 32 |
| 6 | Net Ports | 33 |
| 7 | FIFO Ports | 35 |
| 8 | Formal Parameters | 36 |
| 8.1 | Parameters with enumerated types | 42 |
| 8.1.1 | Parameter user_version | 42 |
| 8.1.2 | Parameter priv_version | 42 |
| 8.1.3 | Parameter compress_version | 43 |
| 8.1.4 | Parameter dsp_version | 43 |
| 8.1.5 | Parameter debug_version | 43 |
| 8.1.6 | Parameter rnmi_version | 43 |
| 8.1.7 | Parameter Smepmp_version | 43 |
| 8.1.8 | Parameter fp16_version | 44 |
| 8.1.9 | Parameter mstatus_fs_mode | 44 |
| 8.1.10 | Parameter debug_mode | 44 |
| 8.1.11 | Parameter lr_sc_constraint | 44 |
| 8.1.12 | Parameter amo_constraint | 44 |
| 8.1.13 | Parameter push_pop_constraint | 45 |

| | | |
|-----------|---|-----------|
| 8.1.14 | Parameter chain_tval | 45 |
| 8.1.15 | Parameter PMP_R0W1 | 45 |
| 8.1.16 | Parameter Ssnpm | 45 |
| 8.1.17 | Parameter Smnpm | 45 |
| 8.1.18 | Parameter Smmpm | 46 |
| 8.1.19 | Parameter Zfinx_version | 46 |
| 8.2 | Parameter values and limits | 46 |
| 9 | Execution Modes | 51 |
| 10 | Exceptions | 52 |
| 11 | Hierarchy of the model | 53 |
| 11.1 | Level 1: Hart | 53 |
| 12 | Model Commands | 54 |
| 12.1 | Level 1: Hart | 54 |
| 12.1.1 | debugflags | 54 |
| 12.1.2 | dumpTLB | 54 |
| 12.1.2.1 | Argument description | 54 |
| 12.1.3 | getCSRIndex | 54 |
| 12.1.4 | isync | 54 |
| 12.1.5 | itrace | 55 |
| 12.1.6 | listCSRs | 55 |
| 12.1.6.1 | Argument description | 55 |
| 12.1.7 | setPMA | 55 |
| 13 | Registers | 56 |
| 13.1 | Level 1: Hart | 56 |
| 13.1.1 | Core | 56 |
| 13.1.2 | Floating_point | 57 |
| 13.1.3 | User_Control_and_Status | 57 |
| 13.1.4 | Supervisor_Control_and_Status | 59 |
| 13.1.5 | Machine_Control_and_Status | 59 |
| 13.1.6 | Integration_support | 63 |

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 RV32GCP 32-bit processor model

1.2 Licensing

This Model is released under the Open Source Apache 2.0

1.3 Extensions

1.3.1 Extensions Enabled by Default

The model has the following architectural extensions enabled, and the corresponding 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/RV64I/RV128I base integer instruction set

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

misa bit 15: extension P (DSP instructions)

misa bit 18: extension S (Supervisor mode)

misa bit 20: extension U (User mode)

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, if supported on this variant. Parameter “sub_Extensions_mask” can be used to disable dynamic update of features in the same way.

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

1.3.2 Enabling Other Extensions

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

misa bit 4: RV32E base integer instruction set (embedded)

misa bit 7: extension H (hypervisor)

misa bit 13: extension N (user-level interrupts)

misa bit 21: extension V (vector extension)

misa bit 23: extension X (non-standard extensions present)

To add features from this list to the visible set in the misa register, use parameter “add_Extensions”. This is a string containing identification letters of features to enable; for example, value “DV”

indicates that double-precision floating point and the Vector Extension should be enabled, if they are currently absent and are available on this variant.

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

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

implicit feature bit 1: extension B (bit manipulation extension)

implicit feature bit 10: extension K (cryptographic)

To add features from this list to the implicitly-enabled set (not visible in the `misa` register), use parameter “`add_implicit_Extensions`”. This is a string parameter in the same format as the “`add_Extensions`” parameter described above.

1.3.3 Disabling Extensions

The following `misa`-visible extensions are enabled by default in the model and can be disabled:

`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 12: extension M (integer multiply/divide instructions)

`misa` bit 15: extension P (DSP instructions)

`misa` bit 18: extension S (Supervisor mode)

`misa` bit 20: extension U (User mode)

To disable features that are enabled by default, use parameter “`sub_Extensions`”. This is a string containing identification letters of features to disable; for example, value “DF” indicates that double-precision and single-precision floating point extensions should be disabled, if they are enabled by default on this variant.

1.4 General Features

1.4.1 mtvec CSR

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 `0xfffffd`. 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.

If parameter “mtvec_sext” is True, values written to “mtvec” are sign-extended from the most-significant writable bit. In this variant, “mtvec_sext” is False, indicating that “mtvec” is not sign-extended.

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

1.4.2 stvec CSR

Values written to “stvec” are masked using the value 0xffffffd. A different mask of writable bits may be specified using parameter “stvec_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.

If parameter “stvec_sext” is True, values written to “stvec” are sign-extended from the most-significant writable bit. In this variant, “stvec_sext” is False, indicating that “stvec” is not sign-extended.

1.4.3 Reset

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

1.4.4 NMI

An NMI input is implemented. To instead specify that NMI is not implemented, set parameter “nmi_absent” to True.

On an NMI, the model will restart at address 0x0; a different NMI address may be specified using parameter “nmi_address” or applied using optional input port “nmi_addr” if required. The cause reported on an NMI is 0x0 by default; a different cause may be specified using parameter “ecode_nmi” or applied using optional input port “nmi_cause” if required.

If parameter “rnmi_version” is not “none”, resumable NMIs are supported, managed by additional CSRs “mnscratch”, “mnepc”, “mncause” and “mnstatus”, following the indicated version of the Resumable NMI extension proposal. In this variant, “rnmi_version” is “none”.

On taking an NMI exception, the mstatus CSR will not be updated. To instead specify that mstatus fields mie and mpie should be updated upon taking an NMI exception, set parameter “nmi_update_mstatus” to True.

On taking an NMI exception, the tcontrol CSR will not be updated. To instead specify that tcontrol fields mte and mpie should be updated upon taking an NMI exception, set parameter “nmi_update_tcontrol” to True.

On taking an NMI exception, the mtval CSR will not be updated. To instead specify that mtval should be written with zero upon taking an NMI exception, set parameter “nmi_zero_mtval” to

True.

The NMI input is level-sensitive. To instead specify that the NMI input is latched on the rising edge of the NMI signal, set parameter “nmi_is_latched” to True.

NMI interrupts are lower priority than Debug and Trigger Module events. To instead specify that NMI interrupts are higher priority, set parameter “nmi_high_priority” to True.

1.4.5 WFI

WFI will halt the processor until an interrupt occurs. It can instead be configured as a NOP by setting parameter “wfi_is_nop” to True.

The nominal time limit for WFI instructions can be set by parameter “TW_time_limit”. In this variant, the time limit is 0 cycles.

Output signal “core_wfi_mode” indicates whether the processor is currently in WFI state.

Input signal “restart_wfi” will cause the processor to restart from WFI state when high.

Parameter “wfi_resume_not_trap” is 0 on this variant, meaning that pending wakeup events when WFI is executed will not prevent a trap occurring. If “wfi_resume_not_trap” is set to 1 then pending wakeup events when WFI is executed will cause the WFI to be treated as a NOP.

1.4.6 cycle CSR

The “cycle” CSR is implemented in this variant. Set parameter “cycle_undefined” to True to instead specify that “cycle” is unimplemented and accesses should cause Illegal Instruction traps.

1.4.7 instret CSR

The “instret” CSR is implemented in this variant. Set parameter “instret_undefined” to True to instead specify that “instret” is unimplemented and accesses should cause Illegal Instruction traps.

1.4.8 hpmcounter CSR

The “hpmcounter” CSRs are implemented in this variant. Set parameter “hpmcounter_undefined” to True to instead specify that “hpmcounter” CSRs are unimplemented and accesses should cause Illegal Instruction traps.

1.4.9 time CSR

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 cause Illegal Instruction traps. Usually, the value of the “time” CSR should be provided by the platform - see section “Time and Timers” for more information.

1.4.10 mcycle CSR

The “mcycle” CSR is implemented in this variant. Set parameter “mcycle_undefined” to True to instead specify that “mcycle” is unimplemented and accesses should cause Illegal Instruction traps.

1.4.11 minstret CSR

The “minstret” CSR is implemented in this variant. Set parameter “minstret_undefined” to True to instead specify that “minstret” is unimplemented and accesses should cause Illegal Instruction traps.

1.4.12 mhpmcounter CSR

The “mhpmcounter” CSRs are implemented in this variant. Set parameter “mhpmcounter_undefined” to True to instead specify that “mhpmcounter” CSRs are unimplemented and accesses should cause Illegal Instruction traps.

1.4.13 Virtual Memory

This variant supports address translation modes 0 (bare) and 1 (Sv32). Use parameter “Sv_modes” to specify a bit mask of different implemented modes if required; for example, setting “Sv_modes” to $(1 < 0) + (1 < 8)$ indicates that mode 0 (bare) and mode 8 (Sv39) are implemented. These indices correspond to writable values in the satp.MODE CSR field.

A 9-bit ASID is implemented. Use parameter “ASID_bits” to specify a different implemented ASID size if required.

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. 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.

Boolean parameter “ignore_non_leaf_DAU” specifies how non-zero D, A and U fields in non-leaf PTE entries are handled. If “ignore_non_leaf_DAU” is False, then using such entries will trigger Page Fault traps. If “ignore_non_leaf_DAU” is True, then such entries will be handled as if D, A and U fields are all zero. In this variant, “ignore_non_leaf_DAU” is 0.

1.4.14 Unaligned Accesses

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 “Zam” to “T” to enable such accesses.

Address misaligned exceptions are higher priority than page fault or access fault exceptions on this variant. Set parameter “unaligned_low_pri” to “T” to specify that they are lower priority instead.

1.4.15 PMP

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. (The grain size in bytes is a 64 bit value calculated using the formula $4 \ll \text{PMP_grain}$.)

Unaligned PMP accesses are not decomposed into separate aligned accesses; use parameter “PMP_decompose” to modify this behavior if required.

Parameters to change the write masks for the PMP CSRs are not enabled; use parameter “PMP_maskparams” to modify this behavior if required.

Parameters to change the reset values for the PMP CSRs are not enabled; use parameter “PMP_initialparams” to modify this behavior if required

The total number of PMP registers present is 64, (this includes CSRs for unimplemented PMP regions), as determined by the requirements of the selected priv_version. Set parameter “PMP_csrs” to specify a different total number of PMP registers.

Accesses to unimplemented PMP registers are write-ignored and read as zero on this variant. Set parameter “PMP_undefined” to True to indicate that such accesses should cause Illegal Instruction exceptions instead.

1.4.16 Time and Timers

A RISC-V hart requires an incrementing time source to be available in any of the following cases:

1. The “time” CSR is implemented (“time_undefined” is False);
2. The “Sstc” extension is present (“Sstc” is True);
3. The internal CLINT model is enabled (“CLINT_address” is non-zero).

The time value in this model is implemented by an abstract counter object, which is shared with any other processor models or peripherals that use the same name to define the counter. The name of the counter is specified by the “mtime_counter” parameter (“mtime_counter” by default). The bit width of the counter is specified by the “mtime_bits” parameter (64 by default). The frequency of the counter is specified by the “mtime_Hz” parameter (1e+06Hz by default).

1.5 Compressed Extension

This variant implements the compressed extension with version specified in the References section of this document. Note that parameter “compress_version” can be used to select the required

architecture version. See the following sections for detailed information about differences between each supported version.

1.5.1 Compressed Extension Parameters

Parameter “Zca” is used to specify that basic C extension instructions are present. By default, “Zca” is set to 1 in this variant. Updates to this parameter require a commercial product license.

Parameter “Zcf” is used to specify that floating point load/store instructions are present. By default, “Zcf” is set to 1 in this variant. Updates to this parameter require a commercial product license.

Parameter “Zcb” is used to specify that additional simple operation instructions are present. By default, “Zcb” is set to 0 in this variant. Updates to this parameter require a commercial product license.

Parameter “Zcmp” is used to specify that push/pop and double move instructions are present. By default, “Zcmp” is set to 0 in this variant. Updates to this parameter require a commercial product license.

Parameter “Zcmt” is used to specify that table jump instructions are present. By default, “Zcmt” is set to 0 in this variant. Updates to this parameter require a commercial product license.

Parameter “jvt_mask” is used to specify writable bits in the jvt CSR. By default, “jvt_mask” is set to 0xffffffffffc0 in this variant.

1.5.2 Legacy Version 1.10

Legacy encodings with version specified using Zcea, Zceb and Zcee parameters.

1.5.3 Version 0.70.1

All instruction encodings changed from legacy version, with instructions divided into Zca, Zcf, Zcb, Zcmb, Zcmp, Zcmpe and Zcmt subsets.

1.5.4 Version 0.70.5

Version 0.70.5, with these changes compared to version 0.70.1:

- access to jt and jalt instructions is enabled by Smstateen.
- jvt.base is WARL and fewer bits than the maximum can be implemented

1.5.5 Version 1.0.0-RC5.7

Version 1.0.0-RC5.7, with these changes compared to version 0.70.5:

- encodings of jt and jalt instructions changed.

- Zcmb and Zcmpe subsets removed.

1.5.6 Version 1.0

Ratified version 1.0, identical to version 1.0.0-RC5.7.

1.6 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.

Additional floating point instructions are not implemented. Use parameter “Zfa” to enable these if required.

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.

If “mstatus_fs_mode” is set to “execute_not_store” then mstatus.FS will be set to 3 (Dirty) when any floating point instruction except a floating point store is executed. This includes all instructions selected by “write_any” mode, and also floating point moves targeting GPRs.

If “mstatus_fs_mode” is set to “execute_any” then mstatus.FS will be set to 3 (Dirty) when any floating point instruction is executed. This includes all instructions selected by “execute_not_store” mode, and also floating point stores.

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

Half precision floating point is not implemented. Use parameter “Zfh” to enable this if required.

1.7 Privileged Architecture

This variant implements the Privileged Architecture with version specified in the References section of this document. Note that parameter “priv_version” can be used to select the required architecture version; see the following sections for detailed information about differences between each supported version.

1.7.1 Legacy Version 1.10

1.10 version of May 7 2017.

1.7.2 Version 20190608

Stable 1.11 version of June 8 2019, with these changes compared to version 1.10:

- mcountinhibit CSR defined;
- pages are never executable in Supervisor mode if page table entry U bit is 1;
- mstatus.TW is writable if any lower-level privilege mode is implemented (previously, it was just if Supervisor mode was implemented);

1.7.3 Version 20211203

1.12 draft version of December 3 2021, with these changes compared to version 20190608:

- mstatush, mseccfg, mseccfgh, menvcfg, menvcfgh, senvcfg, henvcfg, henvcfgh and mconfigptr CSRs defined;
- xret instructions clear mstatus.MPRV when leaving Machine mode if new mode is less privileged than M-mode;
- maximum number of PMP registers increased to 64;
- data endian is now configurable.

1.7.4 Version 1.12

Official 1.12 version, identical to 20211203.

1.7.5 Version master

Unstable master version, currently identical to 1.12.

1.8 Unprivileged Architecture

This variant implements the Unprivileged Architecture with version specified in the References section of this document. Note that parameter “user_version” can be used to select the required architecture version; see the following sections for detailed information about differences between each supported version.

1.8.1 Legacy Version 2.2

2.2 version of May 7 2017.

1.8.2 Version 20191213

Stable 20191213-Base-Ratified version of December 13 2019, with these changes compared to version 2.2:

- floating point fmin/fmax instruction behavior modified to comply with IEEE 754-201x.
- numerous other optional behaviors can be separately enabled using Z-prefixed parameters.

1.9 DSP Extension

This variant implements the DSP extension with version specified in the References section of this document. Note that parameter “dsp_version” can be used to select the required version of this extension.

1.9.1 DSP Extension Parameters

Parameter “Zpsfooperand” is used to specify that register-pair instructions are present. By default, “Zpsfooperand” is set to 1 in this variant. Updates to this parameter require a commercial product license.

1.9.2 DSP Extension Versions

The DSP 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 versions of the specification. The differing features of these are listed below, in chronological order.

1.9.3 Version 0.5.2

Stable 0.5.2 version of July 8 2019.

1.9.4 Version 0.9.6

Stable 0.9.6 version of September 8 2021, with these changes compared to version 0.5.2:

- major opcode is changed from 1111111 to 1110111;
- use of misaligned (odd-numbered) registers for RV32 64-bit instructions is reserved;
- regardless of endianness, for RV32 64-bit instructions the lower-numbered register holds the low-order bits, and the higher-numbered register holds the high-order bits;
- when an RV32 64-bit result is written to x0, the entire write takes no effect;
- when x0 is used as an RV32 64-bit operand, the entire operand is zero;
- changed ucode (0x801) CSR to vxsat CSR (0x009);
- CLO and SWAP16 instructions have been removed;
- modified encoding and field layout of BPICK instruction;
- modified minor encodings of STAS16, RSTAS16, KSTAS16, URSTAS16, UKSTAS16, STSA16, RSTSA16, KSTSA16, URSTSA16, UKSTSA16, STAS32, RSTAS32, KSTAS32, URSTAS32, UKSTAS32, STSA32, RSTSA32, KSTSA32, URSTSA32 and UKSTSA32.

1.10 Other Extensions

Other extensions that can be configured are described in this section.

1.10.1 Zmmul

Parameter “Zmmul” is 0 on this variant, meaning that all multiply and divide instructions are implemented. if “Zmmul” is set to 1 then multiply instructions are implemented but divide and remainder instructions are not implemented.

1.10.2 Zicsr

Parameter “Zicsr” is 1 on this variant, meaning that standard CSRs and CSR access instructions are implemented. if “Zicsr” is set to 0 then standard CSRs and CSR access instructions are not implemented and an alternative scheme must be provided as a processor extension.

1.10.3 Zifencei

Parameter “Zifencei” is 1 on this variant, meaning that the fence.i instruction is implemented (but treated as a NOP by the model). if “Zifencei” is set to 0 then the fence.i instruction is not implemented.

1.10.4 Zicbom

Parameter “Zicbom” is 0 on this variant, meaning that code block management instructions are undefined. if “Zicbom” is set to 1 then code block management instructions `cbo.clean`, `cbo.flush` and `cbo.inval` are defined.

If Zicbom is present, the cache block size is given by parameter “`cmomp_bytes`”. The instructions may cause traps if used illegally but otherwise are NOPs in this model.

1.10.5 Zicbop

Parameter “Zicbop” is 0 on this variant, meaning that prefetch instructions are undefined. if “Zicbop” is set to 1 then prefetch instructions `prefetch.i`, `prefetch.r` and `prefetch.w` are defined (but behave as NOPs in this model).

1.10.6 Zicboz

Parameter “Zicboz” is 0 on this variant, meaning that the `cbo.zero` instruction is undefined. if “Zicboz” is set to 1 then the `cbo.zero` instruction is defined.

If Zicboz is present, the cache block size is given by parameter “`cmoz_bytes`”.

1.10.7 Smstateen

Parameter “Smstateen” is 0 on this variant, meaning that state enable CSRs are undefined. if “Smstateen” is set to 1 then state enable CSRs are defined.

These state enable bits are currently implemented:

bit 1 : Zfinx extension enable

bit 2 : Zcmt extension enable

bit 57 : xcontext CSR access

bit 58 : IMSIC CSR access

bit 59 : AIA CSR access

bit 60 : sireg CSR access

bit 62 : xenvcfg CSR access

bit 63 : lower-level state enable CSR access

1.10.8 Sscofpmf

Parameter “Sscofpmf” is 0 on this variant, meaning that count overflow and mode-based filtering extension CSRs are undefined. if “Sscofpmf” is set to 1 then count overflow and mode-based filtering extension CSRs are defined.

Note that this model implements CSR state only for this extension; hardware performance counters themselves are implementation-specific and not implemented.

1.10.9 Smcntrpmf

Parameter “Smcntrpmf” is 0 on this variant, meaning that cycle and instret mode-based filtering extension CSRs are undefined. if “Smcntrpmf” is set to 1 then cycle and instret mode-based filtering extension CSRs are defined.

1.10.10 Smcsrind

Parameter “Smcsrind” is 0 on this variant, meaning that indirect CSR access is not implemented. if “Smcsrind” is set to 1 then indirect CSR access is implemented. If Supervisor mode is implemented, this also implicitly enables Sscsrind.

1.10.11 Smcdeleg

Parameter “Smcdeleg” is 0 on this variant, meaning that supervisor counter delegation is not implemented. if “Smcdeleg” is set to 1 then supervisor counter delegation is implemented. This also implicitly enables Sscfgr.

1.10.12 Ssqosid

Parameter “Ssqosid” is 0 on this variant, meaning that CSR srmcgr is undefined. if “Ssqosid” is set to 1 then CSR srmcgr is defined.

1.10.13 Zawrs

Parameter “Zawrs” is 0 on this variant, meaning that wait-for-reservation-set instructions are not implemented. if “Zawrs” is set to 1 then wait-for-reservation-set instructions are implemented, in which case parameter “TW_time_limit” is used to specify the nominal cycle delay for wrs.nto, and parameter “STO_time_limit” is used to specify the nominal cycle delay for wrs.sto.

1.10.14 Zimop

Parameter “Zimop” is 0 on this variant, meaning that maybe operations are not implemented. if “Zimop” is set to 1 then maybe operations are implemented.

1.10.15 Zcmop

Parameter “Zcmop” is 0 on this variant, meaning that compressed maybe operations are not implemented. if “Zcmop” is set to 1 then compressed maybe operations are implemented.

1.11 CLIC

The model can be configured to implement a Core Local Interrupt Controller (CLIC) using parameter “CLICLEVELS”; when non-zero, the CLIC is present with the specified number of interrupt levels (2-256), as described in the RISC-V Core-Local Interrupt Controller specification, and further parameters are made available to configure other aspects of the CLIC. “CLICLEVELS” is zero in this variant, indicating that a CLIC is not implemented.

1.12 Advanced Interrupt Architecture

The model can be configured to implement the Advanced Interrupt Architecture (AIA) interface using Boolean parameter “Smaia”; when True, the AIA interface is present as described in the RISC-V Advanced Interrupt Architecture specification, and further parameters are made available to configure other aspects of the interface. “Smaia” is False in this variant, indicating that the AIA interface is not implemented.

1.13 Load-Reserved/Store-Conditional Locking

By default, LR/SC locking is implemented automatically by the model and simulator, with a reservation granule defined by the “lr_sc_grain” parameter; this variant implements a 1-byte reservation granule. It is also possible to implement locking externally to the model in a platform component, using the “LR_address”, “SC_address” and “SC_valid” net ports, as described below.

The “LR_address” output net port is written by the model with the address used by a load-reserved instruction as it executes. This port should be connected as an input to the external lock management component, which should record the address, and also that an LR/SC transaction is active.

The “SC_address” output net port is written by the model with the address used by a store-conditional instruction as it executes. This should be connected as an input to the external lock management component, which should compare the address with the previously-recorded load-reserved address, and determine from this (and other implementation-specific constraints) whether the store should succeed. It should then immediately write the Boolean success/fail code to the “SC_valid” input net port of the model. Finally, it should update state to indicate that an LR/SC transaction is no longer active.

It is also possible to write zero to the “SC_valid” input net port at any time outside the context of a store-conditional instruction, which will mark any active LR/SC transaction as invalid.

Irrespective of whether LR/SC locking is implemented internally or externally, taking any exception or interrupt or executing exception-return instructions (e.g. `mret`) will always mark any active LR/SC transaction as invalid.

Parameter “amo_aborts_lr_sc” is used to specify whether AMO operations abort any active LR/SC pair. In this variant, “amo_aborts_lr_sc” is 0.

Parameter “Zacas” is used to specify whether atomic compare-and-swap instructions are implemented. In this variant, “Zacas” is False.

1.14 Active Atomic Operation Indication

The “AMO_active” output net port is written by the model with a code indicating any current atomic memory operation while the instruction is active. The written codes are:

- 0: no atomic instruction active
- 1: AMOMIN active
- 2: AMOMAX active
- 3: AMOMINU active
- 4: AMOMAXU active
- 5: AMOADD active
- 6: AMOXOR active
- 7: AMOOR active
- 8: AMOAND active
- 9: AMOSWAP active
- 10: AMOCAS.W active (Zacas extension)
- 11: AMOCAS.D active (Zacas extension)
- 12: AMOCAS.Q active (Zacas extension)
- 13: LR active
- 14: SC active

1.15 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 or “reset_addr” port when the signal goes low. The “mcause” register is cleared to zero.

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

All other interrupt ports are active high. For each implemented privileged execution level, there are by default input ports for software interrupt, timer interrupt and external interrupt; for example, for Machine mode, these are called “MSWInterrupt”, “MTimerInterrupt” and “MExternalInterrupt”, respectively. When the N extension is implemented, ports are also present for User mode. Parameter “unimp_int_mask” allows the default behavior to be changed to exclude certain interrupt ports. The parameter value is a mask in the same format as the “mip” CSR; any interrupt corresponding to a non-zero bit in this mask will be removed from the processor and read as zero in “mip”, “mie” and “mideleg” CSRs (and Supervisor and User mode equivalents if implemented).

Parameter “external_int_id” can be used to enable extra interrupt ID input ports on each hart. If

the parameter is True then when an external interrupt is taken the value on the ID port is sampled and used to fill the Exception Code field in the relevant “xcause” CSR. For Machine External interrupts, the extra interrupt ID port is called “MExternalInterruptID”; for Supervisor External interrupts, the extra interrupt ID port is called “SEExternalInterruptID”.

The “deferint” port is an active-high artifact input that, when written to 1, prevents any pending-and-enabled interrupt being taken (normally, such an interrupt would be taken on the next instruction after it becomes pending-and-enabled). The purpose of this signal is to enable alignment with hardware models in step-and-compare usage.

1.16 Debug Mode (Sdext)

The model can be configured to implement Debug mode using parameter “debug_mode”. This implements features described in Chapter 4 of the RISC-V External Debug Support specification with version specified by parameter “debug_version” (see References). Some aspects of this mode are not defined in the specification because they are implementation-specific; the model provides infrastructure to allow implementation of a Debug Module using a custom harness. Features added are described below.

Parameter “debug_mode” can be used to specify four different behaviors, as follows:

1. If set to value “vector”, then operations that would cause entry to Debug mode result in the processor jumping to the address specified by the “debug_address” parameter. It will execute at this address, in Debug mode, until a “dret” instruction causes return to non-Debug mode. Any exception generated during this execution will cause a jump to the address specified by the “dexc_address” parameter.
2. If set to value “interrupt”, then operations that would cause entry to Debug mode result in the processor simulation call (e.g. `opProcessorSimulate`) returning, with a stop reason of `OP_SR_INTERRUPT`. In this usage scenario, the Debug Module is implemented in the simulation harness.
3. If set to value “halt”, then operations that would cause entry to Debug mode result in the processor halting. Depending on the simulation environment, this might cause a return from the simulation call with a stop reason of `OP_SR_HALT`, or debug mode might be implemented by another platform component which then restarts the debugged processor again.
4. If set to value “inject”, then operations that would cause entry to Debug mode result in the processor continuing to execute from the current address in Debug mode. The harness should detect that Debug mode has been entered by monitoring the “DM” integration support register, and inject Debug-mode instructions one at a time using function `opProcessorSimulateInstruction`. Debug mode is exited by either an explicit write of False to the “DM” register or by execution of an injected “dret” instruction, as described by “Debug State Exit” below.

Note that parameter “debug_mode” can be also be set to “none” to specify that Sdext is not implemented.

1.16.1 Debug State Entry

The specification does not define how Debug mode is implemented. In this model, Debug mode is enabled by a Boolean pseudo-register, “DM”. When “DM” is True, the processor is in Debug mode. When “DM” is False, mode is defined by “mstatus” in the usual way.

Entry to Debug mode can be performed in any of these ways:

1. By writing True to register “DM” (e.g. using opProcessorRegWrite) followed by simulation of at least one cycle (e.g. using opProcessorSimulate) - in this case, dcsr.cause will report a cause of trigger (2);
2. By writing a 1 then 0 to net “haltreq” (using opNetWrite) followed by simulation of at least one cycle (e.g. using opProcessorSimulate) - in this case, dcsr.cause will report a cause of haltreq (3);
3. By writing a 1 to net “resethaltreq” (using opNetWrite) while the “reset” signal undergoes a negedge transition, followed by simulation of at least one cycle (e.g. using opProcessorSimulate) - in this case, dcsr.cause will report a cause of resethaltreq (5) or haltreq (3), depending on the value of parameter “no_resethaltreq”;
4. By executing an “ebreak” instruction when Debug mode entry for the current processor mode is enabled by dcsr.ebreakm, dcsr.ebreaks or dcsr.ebreaku - in this case, dcsr.cause will report a cause of ebreak (1);
5. By executing a single instruction when Debug mode entry for the current processor mode is enabled by dcsr.step - in this case, dcsr.cause will report a cause of step (4);
6. By a Trigger Module trigger, when that trigger is configured to enter Debug mode - in this case, dcsr.cause will report a cause of trigger (2).

In all cases, the processor will save required state in “dpc” and “dcsr” and then perform actions described above, depending in the value of the “debug_mode” parameter.

1.16.2 Debug State Exit

Exit from Debug mode can be performed in either of these ways:

1. By writing False to register “DM” (e.g. using opProcessorRegWrite) followed by simulation of at least one cycle (e.g. using opProcessorSimulate);
2. By executing an “dret” instruction when Debug mode.

In both cases, the processor will perform the steps described in section 4.6 (Resume) of the Debug specification.

1.16.3 Debug Registers

When Debug mode is enabled, registers “dcsr” and “dpc” are implemented as described in the specification. Registers “dscratch0” and “dscratch1” may also be implemented (see parameters below). Implemented registers may be manipulated externally by a Debug Module using opProcessorRegRead or opProcessorRegWrite; for example, the Debug Module could write “dcsr” to enable

“ebreak” instruction behavior as described above, or read and write “dpc” to emulate stepping over an “ebreak” instruction prior to resumption from Debug mode.

Parameter “dscratch0_undefined” is used to specify whether the “dscratch0” CSR is undefined, in which case accesses to it trap to Machine mode. In this variant, “dscratch0_undefined” is 0.

Parameter “dscratch1_undefined” is used to specify whether the “dscratch1” CSR is undefined, in which case accesses to it trap to Machine mode. In this variant, “dscratch1_undefined” is 0.

1.16.4 Debug Mode Execution

The specification allows execution of code fragments in Debug mode. A Debug Module implementation can cause execution in Debug mode by the following steps:

1. Write the address of a Program Buffer to the program counter using `opProcessorPCSet`;
2. If “debug_mode” is set to “halt”, write 0 to pseudo-register “DMStall” (to leave halted state);
3. If entry to Debug mode was handled by exiting the simulation callback, call `opProcessorSimulate` or `opRootModuleSimulate` to resume simulation.

Debug mode will be re-entered in these cases:

1. By execution of an “ebreak” instruction; or:
2. By execution of an instruction that causes an exception.

In both cases, the processor will either jump to the debug exception address, or return control immediately to the harness, with `stopReason` of `OP_SR_INTERRUPT`, or perform a halt, depending on the value of the “debug_mode” parameter.

1.16.5 Debug Single Step

When in Debug mode, the processor or harness can cause a single instruction to be executed on return from that mode by setting `dcsr.step`. After one non-Debug-mode instruction has been executed, control will be returned to the harness. The processor will remain in single-step mode until `dcsr.step` is cleared.

1.16.6 Debug Event Priorities

The model supports three different models for determining which debug exception occurs when step, execute address, `resethaltreq` and `haltreq` events are all pending. These options are listed below, with highest-priority event first:

1. when parameter “debug_priority”=“sxh”: `step ->execute address ->resethaltreq ->haltreq`;
2. when parameter “debug_priority”=“shx”: `step ->resethaltreq ->haltreq ->execute address`;
3. when parameter “debug_priority”=“hsx”: `resethaltreq ->haltreq ->step ->execute address`.

1.16.7 Debug Ports

Port “DM” is an output signal that indicates whether the processor is in Debug mode

Port “haltreq” is a rising-edge-triggered signal that triggers entry to Debug mode (see above).

Port “resethaltreq” is a level-sensitive signal that triggers entry to Debug mode after reset (see above).

1.16.8 Debug Mode Versions

Debug mode 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 versions of the specification. The differing features of these are listed below, in chronological order.

1.16.9 Version 0.13.2-DRAFT

0.13.2-DRAFT version of March 22 2019.

1.16.10 Version 0.14.0-DRAFT

0.14.0-DRAFT version of November 6 2020.

1.16.11 Version 1.0.0-STABLE

1.0.0-STABLE version of February 9 2022.

1.16.12 Version 1.0-STABLE

1.0-STABLE version of December 28 2022, with these changes compared to version 1.0.0-STABLE:

- nmi is moved from etrigger to itrigger and is now subject to the mode bits in that trigger.

1.17 Trigger Module (Sdtrig)

This model is configured with a trigger module, implementing a subset of the behavior described in Chapter 5 of the RISC-V External Debug Support specification with version specified by parameter “debug_version” (see References).

1.17.1 Trigger Module Restrictions

The model currently supports tdata1 of type 0, type 2 (mcontrol), type 3 (icount), type 4 (itrigger), type 5 (etrigger) and type 6 (mcontrol6). icount triggers are implemented for a single instruction

only, with count hard-wired to 1 and automatic zeroing of mode bits when the trigger fires.

1.17.2 Trigger Module Parameters

Parameter “trigger_num” is used to specify the number of implemented triggers. In this variant, “trigger_num” is 4. Set this parameter to 0 to specify that Sdtrig is not implemented.

Parameter “tinfo” is used to specify the value of the read-only “tinfo” register, which indicates the trigger types supported and also version information which controls the behavior of “mcontrol6”. In this variant, “tinfo” is 0x100807c.

Parameter “trigger_match” is used to specify the legal “match” values for triggers of types 2 and 6. This parameter is a bitmask with 1 bits corresponding to legal values; for example, a “trigger_match” of 0xd, means that triggers of types 0, 2 and 3 are supported. In this variant, “trigger_match” is 0x333f.

Parameter “tdata2_undefined” is used to specify whether the “tdata2” register is undefined, in which case reads of it trap to Machine mode. In this variant, “tdata2_undefined” is 0.

Parameter “tdata3_undefined” is used to specify whether the “tdata3” register is undefined, in which case reads of it trap to Machine mode. In this variant, “tdata3_undefined” is 0.

Parameter “tinfo_undefined” is used to specify whether the “tinfo” register is undefined, in which case reads of it trap to Machine mode. In this variant, “tinfo_undefined” is 0.

Parameter “tcontrol_undefined” is used to specify whether the “tcontrol” register is undefined, in which case accesses to it trap to Machine mode. In this variant, “tcontrol_undefined” is 0.

Parameter “mcontext_undefined” is used to specify whether the “mcontext” register is undefined, in which case accesses to it trap to Machine mode. In this variant, “mcontext_undefined” is 0.

Parameter “scontext_undefined” is used to specify whether the “scontext” register is undefined, in which case accesses to it trap to Machine mode. In this variant, “scontext_undefined” is 0.

Parameter “mscontext_undefined” is used to specify whether the “mscontext” register is undefined, in which case accesses to it trap to Machine mode. In this variant, “mscontext_undefined” is 0.

Parameter “amo_trigger” is used to specify whether load/store triggers are activated for AMO instructions. In this variant, “amo_trigger” is 0.

Parameter “no_hit” is used to specify whether the “hit” bits in tdata1 are unimplemented. In this variant, “no_hit” is 0.

Parameter “no_sselect_2” is used to specify whether the “sselect” field in “textra32”/“textra64” registers is unable to hold value 2 (indicating match by ASID is not allowed). In this variant, “no_sselect_2” is 0.

Parameter “mcontext_bits” is used to specify the number of writable bits in the “mcontext” register. In this variant, “mcontext_bits” is 6.

Parameter “scontext_bits” is used to specify the number of writable bits in the “scontext” register. In this variant, “scontext_bits” is 16.

Parameter “mvalue_bits” is used to specify the number of writable bits in the “mvalue” field in “tex-

tra32”/“textra64” registers; if zero, the “mselect” field is tied to zero. In this variant, “mvalue_bits” is 6.

Parameter “svalue_bits” is used to specify the number of writable bits in the “svalue” field in “textra32”/“textra64” registers; if zero, the “sselect” is tied to zero. In this variant, “svalue_bits” is 16.

Parameter “mcontrol_maskmax” is used to specify the value of field “maskmax” in the “mcontrol” register. In this variant, “mcontrol_maskmax” is 63.

1.18 Debug Mask

It is possible to enable model debug messages in various categories. This can be done statically using the “debugflags” 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;

Value 0x008: enable TLB consistency checking (automatically detect inconsistencies between cached TLB entries and the page table in memory, if these would affect model behavior).

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

1.19 Integration Support

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

1.19.1 Command “setPMA -attributes <attrs>-lo <addr>-hi <addr>”

This command allows PMA attributes to be set for the address range lo:hi. The required attributes are described by the “attrs” string, which can contain any combination of these characters:

“r”: allow read access

“w”: allow write access

“x”: allow execute access

“a”: disallow unaligned accesses

“A”: disallow RVMC_USER1 accesses (often AMO and LR/SC)

“P”: disallow RVMC_USER2 accesses (often push/pop)

“1”: allow 1-byte accesses (if none of 1248 are specified then all are allowed)

“2”: allow 2-byte accesses

“4”: allow 4-byte accesses

“8”: allow 8-byte accesses

<space>, “-”: ignored, use for formatting

The command may be used multiple times, in which case PMA attributes for later commands override those specified for earlier ones where ranges overlap. A common idiom is to deny all access to the entire memory range in the first command before adding back permissions for subregions with subsequent commands.

1.19.2 Command “getCSRIndex -name <name>”

This command returns the index number of a named CSR, or -1 if that CSR does not exist.

1.19.3 Command “listCSRs”

This command lists all implemented CSRs in index order.

1.19.4 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 a read callback at address 0xC010 on the CSR bus.

If both read and write callbacks are installed, or if a read callback is installed and the CSR is in the read-only address space, then the read callback will be used to provide the value for both true accesses and for trace and API register read (using opRegRead, etc). However, if only a read callback is installed and the CSR is in the CSR read/write address space then the callback will be used for true register reads *only*; in this case, the *model* CSR implementation will be used for trace and API register read. This idiom allows values to be injected for volatile CSRs without changing fundamental model behavior.

An artifact net, “readcsr”, can also be used to override the value apparently read from a CSR without resorting to the CSR bus. When a CSR is read into a GPR that is not “x0”, this net is written with a value encoding the CSR number (in bits 11:0) and destination GPR number (in bits 20:16). To use this net:

1. Install a net monitor callback on “readcsr” using opNetWriteMonitorAdd;
2. When the callback is activated, extract the encoded CSR and GPR numbers;
3. If the CSR number corresponds to a CSR of interest, find the OP register corresponding to the GPR using opProcessorRegByIndex;
4. Use opProcessorRegWrite to modify the GPR value.

1.19.5 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 or if LR/SC locking is implemented externally as described above. When parameter “lr_sc_match_size” is True and this is a 64-bit access, the least-significant bit of “LRSCAddress” is one (to indicate a 64-bit access). If parameter “lr_sc_match_size” is False or this is a 32-bit access, the least-significant bit of “LRSCAddress” is zero.

1.19.6 Page Table Walk Introspection

Artifact register “PTWStage” shows the active page table translation stage (0 if no stage active, 1 if HS-stage active, 2 if VS-stage active and 3 if G-stage active). This register is visibly non-zero only in a memory access callback triggered by a page table walk event.

Artifact register “PTWInputAddr” shows the input address of active page table translation. This register is visibly non-zero only in a memory access callback triggered by a page table walk event.

Artifact register “PTWLevel” shows the active level of page table translation (corresponding to index variable “i” in the algorithm described by Virtual Address Translation Process in the RISC-V Privileged Architecture specification). This register is visibly non-zero only in a memory access callback triggered by a page table walk event.

1.19.7 Page Table Walk Query

A banked set of registers provides information about the most recently completed page table walk. There are up to seven banks of registers: bank 0 is for stage 1 walks, bank 1 is for stage 2 walks, and banks 2-6 are for stage 2 walks initiated by stage 1 level 0-4 entry lookups, respectively. Banks 1-6 are present only for processors with the Virtualization Extension. 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”-“PTWAddressL4” record the addresses of level 0 to level 4 entries read, respectively. Register “PTWAddressValid” is a bitmask indicating which address registers contain valid data: bits 0-4 indicate “PTWAddressL0”-“PTWAddressL4”, respectively, bit 5 indicates “PTWBase”, bit 6 indicates “PTWInput”, bit 7 indicates both “PTWOutput” and “PTWPgSize”. For example, the value 0xe3 indicates that “PTWBase”, “PTWInput”, “PTWOutput”, “PTWPgSize” and “PTWAddressL0” and “PTWAddressL1” are valid but “PTWAddressL2”-“PTWAddressL4” are not. Registers “PTWValueL0”-“PTWValueL4” contain page table entry values read at level 0 to level 4. Register “PTWValueValid” is a bitmask indicating which value registers contain valid data: bits 0-4 indicate “PTWValueL0”-“PTWValueL4”, respectively.

1.19.8 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 PTWUD initiates a U/HU-mode walk for a load/store address. Register PTWUX initiates a U/HU-mode walk for a fetch address. Registers PTWSD and PTWSX perform corresponding walks for S/HS mode accesses. If Hypervisor mode is implemented, registers PTWVUD and PTWVUX perform corresponding walks for VU mode accesses, and registers PTWVSD and PTWVSX perform corresponding walks for VS mode accesses. Each walk fills the query registers described above.

1.19.9 Page Table Walk Event

Event “pageTableWalk” triggers on completion of any page table walk (including artifact walks). An event callback registered on this event can access the Page Table Walk Query artifact registers described in the previous section.

1.19.10 Artifact Register “fflags.i”

If parameter “enable_fflags.i” is True, an 8-bit artifact register “fflags.i” is added to the model. This register shows the floating point flags set by the current instruction (unlike the standard “fflags” CSR, in which the flag bits are sticky).

1.19.11 External Stimulation of Illegal Instruction Trap

Artifact input net port “illegalinstr” allows Illegal Instruction traps to be raised externally. On a rising edge of the signal connected to this port, the hart will immediately take an Illegal Instruction trap with “xepc” set to the current program counter.

As a special case, if the hart is currently stalled by a WFI instruction (“wfi_is_nop” is False), it will be restarted and take either an Illegal Instruction or Virtual Instruction trap, based on the current processor mode and the governing TW bit.

1.20 Instruction Disassembly

This model implements a number of parameters to control instruction disassembly, as shown in trace output.

If parameter “use_hw_reg_names” is True, instruction disassembly shows hardware names x0-x31. If “use_hw_reg_names” is False, ABI names are shown instead.

If parameter “no_pseudo_inst” is True, instruction disassembly always shows true instructions. If “no_pseudo_inst” is False, pseudo-instructions are shown instead where applicable.

If parameter “show_c_prefix” is True, instruction disassembly of 16-bit instructions will include a compressed prefix (e.g. “c.” or “cm.”). If “show_c_prefix” is False, the compressed prefix will be omitted.

1.21 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.

Hardware Performance Monitor 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.

1.22 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 Foundation 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.

1.23 References

The Model details are based upon the following specifications:

RISC-V Instruction Set Manual, Volume I: User-Level ISA (User Architecture Version 20191213)

RISC-V Instruction Set Manual, Volume II: Privileged Architecture (Privileged Architecture Version 1.12, equivalent to 20211203)

RISC-V “C” Compressed Extension (Compressed Architecture Version 1.0)

RISC-V “P” DSP Extension (DSP Architecture Version 0.5.2)

RISC-V External Debug Support (RISC-V External Debug Support Version 1.0-STABLE as of commit 83483b1 of 21 August 2023 (this is subject to change))

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 | |
| RV32IMCZ _{ce} | |
| RV32IMAC | |
| RV32G | |
| RV32GC | |
| RV32GCZ _{finx} | |
| RV32GCB | |
| RV32GCH | |
| RV32GCK | |
| RV32GCN | |
| RV32GCP | (described in this document) |
| RV32GCV | |
| RV32E | |
| RV32EC | |
| RV32EM | |
| RV64I | |
| RV64IM | |
| RV64IMC | |
| RV64IMCZ _{ce} | |
| RV64IMAC | |
| RV64G | |
| RV64GC | |
| RV64GCZ _{finx} | |
| RV64GCB | |
| RV64GCH | |
| RV64GCK | |
| RV64GCN | |
| RV64GCP | |
| RV64GCV | |

| | |
|----------|--|
| RVB32I | |
| RVB32E | |
| RVB64I | |
| RVI20U32 | |
| RVI20U64 | |
| RVA20U64 | |
| RVA20S64 | |
| RVA22U64 | |
| RVA22S64 | |

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 | 34 | mandatory | Instruction bus |
| DATA | 32 | 34 | 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 |
| reset_addr | input | optional | Externally-applied reset address |
| nmi | input | optional | NMI |
| nmi_cause | input | optional | Externally-applied NMI cause |
| nmi_addr | input | optional | Externally-applied NMI address |
| 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 |
| irq_ack_o | output | optional | Interrupt acknowledge (pulse) |
| irq_id_o | output | optional | Acknowledged interrupt id (valid during irq_ack_o pulse) |
| sec_lvl_o | output | optional | Current privilege level |
| LR_address | output | optional | Port written with effective address for LR instruction |
| SC_address | output | optional | Port written with effective address for SC instruction |
| SC_valid | input | optional | SC_address valid input signal |
| AMO_active | output | optional | Port written with code indicating active AMO |
| illegalinstr | input | optional | Artifact signal raising Illegal Instruction on rising edge |
| deferint | input | optional | Artifact signal causing interrupts to be held off when high |
| coverpoint | output | optional | Artifact port written with coverage point identifier |
| readcsr | output | optional | Artifact port written with CSR/GPR information when CSR is read |
| core_wfi_mode | output | optional | WFI is active |

| | | | |
|-------------|-------|----------|--|
| restart_wfi | input | optional | Artifact signal causing restart from WFI state when high |
|-------------|-------|----------|--|

Table 6.1: Net Ports

Chapter 7

FIFO Ports

This model has no FIFO ports.

Chapter 8

Formal Parameters

| Name | Type | Description |
|-----------------------------|-------------|---|
| Fundamental | | |
| user_version | Enumeration | Specify required User Architecture version |
| | 2.2 | User Architecture Version 2.2 |
| | 2.3 | Deprecated and equivalent to 20191213 |
| | 20190305 | Deprecated and equivalent to 20191213 |
| | 20191213 | User Architecture Version 20191213 |
| priv_version | Enumeration | Specify required Privileged Architecture version |
| | 1.10 | Privileged Architecture Version 1.10 |
| | 1.11 | Privileged Architecture Version 1.11, equivalent to 20190608 |
| | 20190405 | Deprecated and equivalent to 20190608 |
| | 20190608 | Privileged Architecture Version Ratified-IMFDQC-and-Priv-v1.11 |
| | 20211203 | Privileged Architecture Version 20211203 |
| | 1.12 | Privileged Architecture Version 1.12, equivalent to 20211203 |
| | master | Privileged Architecture Master Branch as of commit 6bdeb58 (this is subject to change) |
| Smepmp_version | Enumeration | Specify required Smepmp Architecture version |
| | none | Smepmp not implemented |
| | 0.9.5 | Smepmp version 0.9.5 (deprecated and identical to 1.0) |
| | 1.0 | Smepmp version 1.0 |
| numHarts | Uns32 | Specify the number of hart contexts in a multiprocessor |
| enable_expanded | Boolean | Specify that 48-bit and 64-bit expanded instructions are supported |
| endianFixed | Boolean | Specify that data endianness is fixed (mstatus.{MBE,SBE,UBE} fields are read-only) |
| misa_MXL | Uns32 | Override default value of 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) |
| sub_Extensions | String | Remove extensions specified by letters from misa.Extensions (for example, specify “VD” to remove 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) |
| sub_Extensions_mask | String | Remove extensions specified by letters from mask of writable bits in misa.Extensions (for example, specify “VD” to remove V and D features) |
| add_implicit_Extensions | String | Add extensions specified by letters to implicitly-present extensions not visible in misa.Extensions |
| sub_implicit_Extensions | String | Remove extensions specified by letters from implicitly-present extensions not visible in misa.Extensions |
| Compressed Extension | | |
| compress_version | Enumeration | Specify required Compressed Architecture version |

| | | |
|------------------------------|--------------|---|
| | legacy | Compressed Architecture absent or legacy version |
| | 0.70.1 | Compressed Architecture Version 0.70.1 |
| | 0.70.5 | Compressed Architecture Version 0.70.5 |
| | 1.0.0-RC5.7 | Compressed Architecture Version 1.0.0-RC5.7 |
| | 1.0 | Compressed Architecture Version 1.0 |
| Zca | Boolean | Specify that Zca is implemented |
| Zcb | Boolean | Specify that Zcb is implemented |
| Zcf | Boolean | Specify that Zcf is implemented |
| Zcmp | Boolean | Specify that Zcmp is implemented |
| Zcmt | Boolean | Specify that Zcmt is implemented |
| Zcmop | Boolean | Specify that Zcmop is implemented |
| DSP Extension | | |
| dsp_version | Enumeration | Specify required DSP Architecture version |
| | 0.5.2 | DSP Architecture Version 0.5.2 |
| | 0.9.6 | DSP Architecture Version 0.9.6 |
| Zpsfoperand | Boolean | Specify that Zpsfoperand is implemented |
| Debug Extension | | |
| debug_version | Enumeration | Specify required Debug Architecture version |
| | 0.13.2-DRAFT | RISC-V External Debug Support Version 0.13.2-DRAFT |
| | 0.14.0-DRAFT | RISC-V External Debug Support Version 0.14.0-DRAFT |
| | 1.0.0-STABLE | RISC-V External Debug Support Version 1.0.0-STABLE |
| | 1.0-STABLE | RISC-V External Debug Support Version 1.0-STABLE as of commit 83483b1 of 21 August 2023 (this is subject to change) |
| debug_mode | Enumeration | Specify how Debug mode is implemented (value 'none' implies Sdext is not implemented) |
| | none | Debug mode not implemented |
| | vector | Debug mode implemented by execution at vector |
| | interrupt | Debug mode implemented by interrupt |
| | halt | Debug mode implemented by halt |
| | inject | Debug mode implemented using injected instructions |
| Interrupts Exceptions | | |
| rnmi_version | Enumeration | Specify required RNMI Architecture version |
| | none | RNMI not implemented |
| | 0.2.1 | RNMI version 0.2.1 |
| | 0.4_nmie1 | RNMI version 0.4, except mnstatus.nmie=1 at reset |
| | 0.4 | RNMI version 0.4 |
| 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 |
| ecode_mask | Uns64 | Specify hardware-enforced mask of writable bits in xcause.ExceptionCode |
| ecode_nmi | Uns64 | Specify xcause.ExceptionCode for NMI |
| nmi_absent | Boolean | Specify whether NMI input is absent |
| nmi_is_latched | Boolean | Specify whether NMI input is latched on rising edge (if False, it is level-sensitive) |
| nmi_high_priority | Boolean | Specify whether NMI input is higher priority than Debug and Trigger Module events |
| nmi_update_mstatus | Boolean | Specify whether an NMI exception updates the mstatus CSR mie and mpie fields |
| nmi_update_tcontrol | Boolean | Specify whether an NMI exception updates the tcontrol CSR mte and mpte fields |
| nmi_zero_mtval | Boolean | Specify whether an NMI exception writes 0 to the mtval CSR |
| mtval.is_ro | Boolean | Specify whether mtval CSR is read-only |
| tval_zero | Boolean | Specify whether mtval/stval/utval are hard wired to zero |
| tval_zero_ebreak | Boolean | Specify whether mtval/stval/utval are set to zero by an ebreak |
| tval_ii_code | Boolean | Specify whether mtval/stval contain faulting instruction bits on illegal instruction exception |

| | | |
|-----------------------|-------------------|--|
| trap_preserves_lr | Boolean | Whether a trap preserves active LR/SC state |
| xret_preserves_lr | Boolean | Whether an xret instruction preserves active LR/SC state |
| reset_address | Uns64 | Override reset vector address |
| nmi_address | Uns64 | Override NMI vector address |
| CLINT_address | Uns64 | Specify base address of internal CLINT model (or 0 for no CLINT) |
| local_int_num | Uns32 | Specify number of local interrupts (excludes standard set 0 to 15 if basic interrupt mode is present) |
| unimp_int_mask | Uns64 | Specify mask of unimplemented interrupts (e.g. 1<<9 indicates Supervisor external interrupt unimplemented) |
| force_mideleg | Uns64 | Specify mask of interrupts always delegated to lower-priority execution level from Machine execution level |
| force_sideleg | Uns64 | Specify mask of interrupts always delegated to User execution level from Supervisor execution level |
| no_ideleg | Uns64 | Specify mask of interrupts that cannot be delegated to lower-priority execution levels |
| no_e deleg | Uns64 | Specify mask of exceptions that cannot be delegated to lower-priority execution levels |
| external_int_id | Boolean | Whether to add nets allowing External Interrupt ID codes to be forced |
| Floating Point | | |
| fp16_version | Enumeration | Specify required 16-bit floating point format |
| | none | No 16-bit floating point implemented |
| | IEEE754 | IEEE 754 half precision implemented |
| | BFLOAT16 | BFLOAT16 implemented |
| | dynamic | Dynamic 16-bit floating point implemented |
| mstatus_fs_mode | Enumeration | Specify conditions causing update of mstatus.FS to dirty |
| | write_1 | Any non-zero flag result sets mstatus.fs dirty |
| | write_any | Any write of flags sets mstatus.fs dirty |
| | execute_not_store | Any floating point non-store instruction sets mstatus.fs dirty |
| | execute_any | Any floating point instruction sets mstatus.fs dirty |
| | always_dirty | mstatus.fs is either off or dirty |
| | force_dirty | mstatus.fs is forced to dirty |
| 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 |
| enable_fflags_i | Boolean | Whether fflags.i artifact register present (shows per-instruction floating point flags) |
| enable_DAZ | Boolean | Whether subnormal floating point operands are flushed to zero (non-compliant option) |
| enable_FZ | Boolean | Whether subnormal floating point results are flushed to zero (non-compliant option) |
| mstatus_FS | Uns32 | Override default value of mstatus.FS (initial state of floating point unit) |
| Zfa | Boolean | Specify that Zfa is implemented (additional floating point instructions) |
| Zfh | Boolean | Specify that Zfh is implemented (IEEE half-precision floating point is supported) |
| Zfhmin | Boolean | Specify that Zfhmin is implemented (restricted IEEE half-precision floating point is supported) |
| Zfbfmin | Boolean | Specify that Zfbfmin is implemented (restricted BFLOAT16 floating point is supported) |
| Zfmx_version | Enumeration | Specify version of Zfmx implemented (use integer register file for floating point instructions) |
| | none | Zfmx not implemented |
| | 0.4 | Zfmx version 0.4 |
| | 0.41 | Zfmx version 0.41 |
| | 1.0 | Zfmx version 1.0 |
| Memory | | |
| lr_sc_constraint | Enumeration | Specify memory constraint for LR/SC instructions |
| | none | Memory access not constrained |

| | | |
|----------------------------|-------------|---|
| | user1 | Memory access constrained by MEM_CONSTRAINT_USER1 |
| | user2 | Memory access constrained by MEM_CONSTRAINT_USER2 |
| | user3 | Memory access constrained by MEM_CONSTRAINT_USER3 |
| | user4 | Memory access constrained by MEM_CONSTRAINT_USER4 |
| amo_constraint | Enumeration | Specify memory constraint for AMO instructions |
| | none | Memory access not constrained |
| | user1 | Memory access constrained by MEM_CONSTRAINT_USER1 |
| | user2 | Memory access constrained by MEM_CONSTRAINT_USER2 |
| | user3 | Memory access constrained by MEM_CONSTRAINT_USER3 |
| | user4 | Memory access constrained by MEM_CONSTRAINT_USER4 |
| push_pop_constraint | Enumeration | Specify memory constraint for PUSH/POP instructions |
| | none | Memory access not constrained |
| | user1 | Memory access constrained by MEM_CONSTRAINT_USER1 |
| | user2 | Memory access constrained by MEM_CONSTRAINT_USER2 |
| | user3 | Memory access constrained by MEM_CONSTRAINT_USER3 |
| | user4 | Memory access constrained by MEM_CONSTRAINT_USER4 |
| updatePTEA | Boolean | Specify whether hardware update of PTE A bit is supported (ignored when parameter Svadu is True) |
| updatePTED | Boolean | Specify whether hardware update of PTE D bit is supported (ignored when parameter Svadu is True) |
| unaligned_low_pri | Boolean | Specify whether address misaligned exceptions are lower priority than page or access fault exceptions |
| unaligned | Boolean | Specify whether the processor supports unaligned memory accesses |
| Zam | Boolean | Specify whether the processor supports unaligned memory accesses for AMO instructions |
| amo_aborts_lr_sc | Boolean | Specify whether AMO operations abort any active LR/SC pair |
| ASID_bits | Uns32 | Specify the number of implemented ASID bits |
| lr_sc_grain | Uns32 | Specify byte granularity of LR/SC lock region (constrained to a power of two) |
| ignore_non_leaf_DAU | Boolean | Whether non-zero D, A and U bits in non-leaf PTEs are ignored (if False, a trap is taken) |
| Sv_modes | Uns32 | Specify bit mask of implemented address translation modes (e.g. (1<<0)+(1<<8) indicates “bare” and “Sv39” modes may be selected in satp.MODE) |
| Simulation Artifact | | |
| leaf_hart_prefix | String | Specify string name prefix for harts in a cluster |
| mtime_counter | String | Specify name of shared mtime counter |
| mtime_Hz | Double | Specify mtime counter frequency |
| mtime_bits | Uns32 | Specify mtime counter bit width |
| use_hw_reg_names | Boolean | Specify whether to use hardware register names x0-x31 and f0-f31 instead of ABI register names |
| no_pseudo_inst | Boolean | Specify whether pseudo-instructions should not be reported in trace and disassembly |
| show_c_prefix | Boolean | Specify whether compressed instruction prefix should be reported in trace and disassembly |
| ABId | Boolean | Specify whether D registers are used for parameters (ABI SemiHosting) |
| verbose | Boolean | Specify verbose output messages |
| traceVolatile | Boolean | Specify whether volatile registers (e.g. minstret) should be shown in change trace |
| wfi_restart | Boolean | Specify whether to automatically restart from WFI state when model is simulated |
| enable_CSR_bus | Boolean | Add artifact CSR bus port, allowing CSR registers to be externally implemented |
| CSR_remap | String | Comma-separated list of CSR number mappings, each of the form <csr-Name>=<number> |

| | | |
|---------------------------------|---------|--|
| ASID.cache_size | Uns32 | Specify the number of different ASIDs for which TLB entries are cached; a value of 0 implies no limit |
| Instruction_CSR_Behavior | | |
| wfi_is_nop | Boolean | Specify whether WFI should be treated as a NOP (if not, halt while waiting for interrupts) |
| wfi_resume_not_trap | Boolean | Specify whether pending wakeup events should cause WFI to be treated as a NOP instead of taking a trap |
| TW_time_limit | Uns32 | Specify nominal cycle timeout for instructions controlled by mstatus.TW |
| counteren_mask | Uns32 | Specify hardware-enforced mask of writable bits in mcounteren/scounteren registers |
| scounteren_zero_mask | Uns32 | Specify hardware-enforced mask of always-zero bits in scounteren register |
| noinhibit_mask | Uns32 | Specify hardware-enforced mask of always-zero bits in mcountinhibit register |
| cycle_undefined | Boolean | Specify that the cycle CSR is undefined |
| mcycle_undefined | Boolean | Specify that the mcycle CSR is undefined |
| time_undefined | Boolean | Specify that the time CSR is undefined |
| instret_undefined | Boolean | Specify that the instret CSR is undefined |
| minstret_undefined | Boolean | Specify that the minstret CSR is undefined |
| hpmcounter_undefined | Boolean | Specify that the hpmcounter CSRs are undefined |
| mhpmcounter_undefined | Boolean | Specify that the mhpmcounter CSRs are undefined |
| CSR_Masks | | |
| 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 |
| jvt_mask | Uns64 | Specify hardware-enforced mask of writable bits in Zcmt jvt register |
| tdata1_mask | Uns64 | Specify hardware-enforced mask of writable bits in Trigger Module tdata1 register |
| mip_mask | Uns64 | Specify hardware-enforced mask of writable bits in mip register |
| sip_mask | Uns64 | Specify hardware-enforced mask of writable bits in sip register |
| envcfg_mask | Uns64 | Specify hardware-enforced mask of writable bits in envcfg registers |
| mtvec_sext | Boolean | Specify whether mtvec is sign-extended from most-significant bit |
| stvec_sext | Boolean | Specify whether stvec is sign-extended from most-significant bit |
| Trigger | | |
| tdata2_undefined | Boolean | Specify that the tdata2 CSR is undefined |
| tdata3_undefined | Boolean | Specify that the tdata3 CSR is undefined |
| tinfo_undefined | Boolean | Specify that the tinfo CSR is undefined |
| tcontrol_undefined | Boolean | Specify that the tcontrol CSR is undefined |
| mcontext_undefined | Boolean | Specify that the mcontext CSR is undefined |
| scontext_undefined | Boolean | Specify that the scontext CSR is undefined |
| mscontext_undefined | Boolean | Specify that the mscontext CSR is undefined (Debug Version 0.14.0 and later) |
| amo_trigger | Boolean | Specify whether AMO load/store operations activate triggers |
| no_hit | Boolean | Specify that tdata1.hit* bits are unimplemented |
| no_sselect_2 | Boolean | Specify that textra.sselect=2 is not supported (no trigger match by ASID) |
| trigger_num | Uns32 | Specify the number of implemented hardware triggers (0 implies Sdtrig is not implemented) |
| mask_tselect | Boolean | Whether values written to tselect are masked according to the trigger_num setting |
| tinfo | Uns32 | Override tinfo register (for all triggers) |
| trigger_match | Uns32 | Specify legal “match” values for triggers of type 2 and 6 (bitmask) |
| mcontext_bits | Uns32 | Specify the number of implemented bits in mcontext |
| scontext_bits | Uns32 | Specify the number of implemented bits in scontext |
| mvalue_bits | Uns32 | Specify the number of implemented bits in textra.mvalue (if zero, textra.mselect is tied to zero) |
| svalue_bits | Uns32 | Specify the number of implemented bits in textra.svalue (if zero, textra.sselect is tied to zero) |
| mcontrol_maskmax | Uns32 | Specify mcontrol.maskmax value |

| | | |
|--------------------------|---------------|---|
| chain_tval | Enumeration | Specify which trigger provides xtval when triggers are chained |
| | first | first matching trigger provides xtval |
| | last | last matching trigger provides xtval |
| | first_non_epc | first matching trigger provides xtval (prefer non-epc) |
| | last_non_epc | last matching trigger provides xtval (prefer non-epc) |
| PMP Configuration | | |
| PMP_grain | Uns32 | Specify PMP region granularity, G (0 =>4 bytes, 1 =>8 bytes, etc) |
| PMP_registers | Uns32 | Specify the number of supported PMP regions |
| PMP_csrs | Uns32 | Specify the number of PMP address CSRs (are RAZ/WI if corresponding region is not implemented) |
| PMP_max_page | Uns32 | Specify the maximum size of PMP region to map if non-zero (may improve performance; constrained to a power of two) |
| PMP_decompose | Boolean | Whether unaligned PMP accesses are decomposed into separate aligned accesses |
| PMP_undefined | Boolean | Whether accesses to unimplemented PMP registers are undefined (if True) or write ignored and zero (if False) |
| PMP_R0W1 | Enumeration | Specify behavior of PMP RWX field on illegal write with R=0 and W=1 |
| | RWX_00X | set R=0 and W=0, modify X |
| | RWX_00P | set R=0 and W=0, preserve X |
| | RWX_11X | set R=1 and W=1, modify X |
| | RWX_PPX | preserve previous R and W, modify X |
| | RWX_PPP | preserve previous RWX |
| | RWX_000 | set RWX=000 |
| PMP_maskparams | Boolean | Enable parameters to change the read-only masks for PMP CSRs |
| PMP_initialparams | Boolean | Enable parameters to change the reset values for PMP CSRs |
| Other Extensions | | |
| Svnapot_page_mask | Uns64 | Specify mask of implemented Svnapot intermediate page sizes (e.g. 1<<16 means 64KiB contiguous regions are supported) |
| Smstateen | Boolean | Specify that Smstateen is implemented |
| Smcsrind | Boolean | Specify that Smcsrind is implemented |
| Sstc | Boolean | Specify that Sstc is implemented |
| Sscfpmf | Boolean | Specify that Sscfpmf is implemented |
| Ssdbltrp | Boolean | Specify that Ssdbltrp is implemented |
| Smcntrpmf | Boolean | Specify that Smcntrpmf is implemented |
| Smcdeleg | Boolean | Specify that Smcdeleg is implemented |
| Svpbmt | Boolean | Specify that Svpbmt is implemented |
| Ssnpm | Enumeration | Specify Ssnpm implemented modes |
| | none | pointer masking not implemented |
| | 48 | PM=XLEN-48 is implemented |
| | 57 | PM=XLEN-57 is implemented |
| | 48.57 | PM=XLEN-48 and PM=XLEN-57 are implemented |
| Smnpm | Enumeration | Specify Smnpm implemented modes |
| | none | pointer masking not implemented |
| | 48 | PM=XLEN-48 is implemented |
| | 57 | PM=XLEN-57 is implemented |
| | 48.57 | PM=XLEN-48 and PM=XLEN-57 are implemented |
| Smmpm | Enumeration | Specify Smmpm implemented modes |
| | none | pointer masking not implemented |
| | 48 | PM=XLEN-48 is implemented |
| | 57 | PM=XLEN-57 is implemented |
| | 48.57 | PM=XLEN-48 and PM=XLEN-57 are implemented |
| Svinval | Boolean | Specify that Svinval is implemented |
| Svadu | Boolean | Specify that Svadu is implemented |
| Ssqosid | Boolean | Specify that Ssqosid is implemented |
| Zihintntl | Boolean | Specify that Zihintntl is implemented (instruction decode only, implemented as NOP) |

| | | |
|-----------------------|---------|--|
| Zicond | Boolean | Specify that Zicond is implemented |
| Zicshr | Boolean | Specify that Zicshr is implemented |
| Zifencei | Boolean | Specify that Zifencei is implemented |
| Zicbom | Boolean | Specify that Zicbom is implemented |
| Zicbop | Boolean | Specify that Zicbop is implemented |
| Zicboz | Boolean | Specify that Zicboz is implemented |
| Zimop | Boolean | Specify that Zimop is implemented |
| Zicfiss | Boolean | Specify that Zicfiss is implemented |
| Zicfilp | Boolean | Specify that Zicfilp is implemented |
| Zaamo | Boolean | Specify that Zaamo is implemented |
| Zalrsc | Boolean | Specify that Zalrsc is implemented |
| Zacas | Boolean | Specify that Zacas is implemented |
| Zabha | Boolean | Specify that Zabha is implemented |
| Zawrs | Boolean | Specify that Zawrs is implemented |
| Zmmul | Boolean | Specify that Zmmul is implemented |
| CSR.Defaults | | |
| mvendorid | Uns64 | Override mvendorid register |
| marchid | Uns64 | Override marchid register |
| mimpid | Uns64 | Override mimpid register |
| mhartid | Uns64 | Override mhartid register (or first mhartid of an incrementing sequence if this is an SMP variant) |
| mconfigptr | Uns64 | Override mconfigptr register |
| mtvec | Uns64 | Override mtvec register |
| mseccfg | Uns64 | Override mseccfg register |
| Fast Interrupt | | |
| CLICLEVELS | Uns32 | Specify number of interrupt levels implemented by CLIC, or 0 if CLIC absent |
| AIA Interrupts | | |
| Smaia | Boolean | Specify that Smaia CSRs are present |

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 20191213 |
| 20190305 | Deprecated and equivalent to 20191213 |
| 20191213 | User Architecture Version 20191213 |

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 | Privileged Architecture Version 1.11, equivalent to 20190608 |
| 20190405 | Deprecated and equivalent to 20190608 |
| 20190608 | Privileged Architecture Version Ratified-IMFDQC-and-Priv-v1.11 |
| 20211203 | Privileged Architecture Version 20211203 |
| 1.12 | Privileged Architecture Version 1.12, equivalent to 20211203 |
| master | Privileged Architecture Master Branch as of commit 6bdeb58 (this is subject to change) |

Table 8.3: Values for Parameter `priv_version`

8.1.3 Parameter `compress_version`

| Set to this value | Description |
|-------------------|--|
| legacy | Compressed Architecture absent or legacy version |
| 0.70.1 | Compressed Architecture Version 0.70.1 |
| 0.70.5 | Compressed Architecture Version 0.70.5 |
| 1.0.0-RC5.7 | Compressed Architecture Version 1.0.0-RC5.7 |
| 1.0 | Compressed Architecture Version 1.0 |

Table 8.4: Values for Parameter `compress_version`

8.1.4 Parameter `dsp_version`

| Set to this value | Description |
|-------------------|--------------------------------|
| 0.5.2 | DSP Architecture Version 0.5.2 |
| 0.9.6 | DSP Architecture Version 0.9.6 |

Table 8.5: Values for Parameter `dsp_version`

8.1.5 Parameter `debug_version`

| Set to this value | Description |
|-------------------|---|
| 0.13.2-DRAFT | RISC-V External Debug Support Version 0.13.2-DRAFT |
| 0.14.0-DRAFT | RISC-V External Debug Support Version 0.14.0-DRAFT |
| 1.0.0-STABLE | RISC-V External Debug Support Version 1.0.0-STABLE |
| 1.0-STABLE | RISC-V External Debug Support Version 1.0-STABLE as of commit 83483b1 of 21 August 2023 (this is subject to change) |

Table 8.6: Values for Parameter `debug_version`

8.1.6 Parameter `rnmi_version`

| Set to this value | Description |
|-------------------|--|
| none | RNMI not implemented |
| 0.2.1 | RNMI version 0.2.1 |
| 0.4_nmie1 | RNMI version 0.4, except <code>mnstatus.nmie=1</code> at reset |
| 0.4 | RNMI version 0.4 |

Table 8.7: Values for Parameter `rnmi_version`

8.1.7 Parameter `Smepmp_version`

| Set to this value | Description |
|-------------------|--|
| none | Smepmp not implemented |
| 0.9.5 | Smepmp version 0.9.5 (deprecated and identical to 1.0) |
| 1.0 | Smepmp version 1.0 |

Table 8.8: Values for Parameter `Smepmp_version`

8.1.8 Parameter fp16_version

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

Table 8.9: Values for Parameter fp16_version

8.1.9 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 |
| execute_not_store | Any floating point non-store instruction sets mstatus.fs dirty |
| execute_any | Any floating point instruction sets mstatus.fs dirty |
| always_dirty | mstatus.fs is either off or dirty |
| force_dirty | mstatus.fs is forced to dirty |

Table 8.10: Values for Parameter mstatus_fs_mode

8.1.10 Parameter debug_mode

| Set to this value | Description |
|-------------------|--|
| none | Debug mode not implemented |
| vector | Debug mode implemented by execution at vector |
| interrupt | Debug mode implemented by interrupt |
| halt | Debug mode implemented by halt |
| inject | Debug mode implemented using injected instructions |

Table 8.11: Values for Parameter debug_mode

8.1.11 Parameter lr_sc_constraint

| Set to this value | Description |
|-------------------|---|
| none | Memory access not constrained |
| user1 | Memory access constrained by MEM_CONSTRAINT_USER1 |
| user2 | Memory access constrained by MEM_CONSTRAINT_USER2 |
| user3 | Memory access constrained by MEM_CONSTRAINT_USER3 |
| user4 | Memory access constrained by MEM_CONSTRAINT_USER4 |

Table 8.12: Values for Parameter lr_sc_constraint

8.1.12 Parameter amo_constraint

| Set to this value | Description |
|-------------------|---|
| none | Memory access not constrained |
| user1 | Memory access constrained by MEM_CONSTRAINT_USER1 |
| user2 | Memory access constrained by MEM_CONSTRAINT_USER2 |
| user3 | Memory access constrained by MEM_CONSTRAINT_USER3 |
| user4 | Memory access constrained by MEM_CONSTRAINT_USER4 |

Table 8.13: Values for Parameter amo_constraint

8.1.13 Parameter push_pop_constraint

| Set to this value | Description |
|-------------------|---|
| none | Memory access not constrained |
| user1 | Memory access constrained by MEM.CONSTRAINT.USER1 |
| user2 | Memory access constrained by MEM.CONSTRAINT.USER2 |
| user3 | Memory access constrained by MEM.CONSTRAINT.USER3 |
| user4 | Memory access constrained by MEM.CONSTRAINT.USER4 |

Table 8.14: Values for Parameter push_pop_constraint

8.1.14 Parameter chain_tval

| Set to this value | Description |
|-------------------|--|
| first | first matching trigger provides xtval |
| last | last matching trigger provides xtval |
| first_non_epc | first matching trigger provides xtval (prefer non-epc) |
| last_non_epc | last matching trigger provides xtval (prefer non-epc) |

Table 8.15: Values for Parameter chain_tval

8.1.15 Parameter PMP_R0W1

| Set to this value | Description |
|-------------------|-------------------------------------|
| RWX_00X | set R=0 and W=0, modify X |
| RWX_00P | set R=0 and W=0, preserve X |
| RWX_11X | set R=1 and W=1, modify X |
| RWX_PPX | preserve previous R and W, modify X |
| RWX_PPP | preserve previous RWX |
| RWX_000 | set RWX=000 |

Table 8.16: Values for Parameter PMP_R0W1

8.1.16 Parameter Ssnpm

| Set to this value | Description |
|-------------------|---|
| none | pointer masking not implemented |
| 48 | PM=XLEN-48 is implemented |
| 57 | PM=XLEN-57 is implemented |
| 48_57 | PM=XLEN-48 and PM=XLEN-57 are implemented |

Table 8.17: Values for Parameter Ssnpm

8.1.17 Parameter Smnpm

| Set to this value | Description |
|-------------------|---|
| none | pointer masking not implemented |
| 48 | PM=XLEN-48 is implemented |
| 57 | PM=XLEN-57 is implemented |
| 48_57 | PM=XLEN-48 and PM=XLEN-57 are implemented |

Table 8.18: Values for Parameter Smnpm

8.1.18 Parameter Smmpm

| Set to this value | Description |
|-------------------|---|
| none | pointer masking not implemented |
| 48 | PM=XLEN-48 is implemented |
| 57 | PM=XLEN-57 is implemented |
| 48.57 | PM=XLEN-48 and PM=XLEN-57 are implemented |

Table 8.19: Values for Parameter Smmpm

8.1.19 Parameter Zfinx_version

| Set to this value | Description |
|-------------------|-----------------------|
| none | Zfinx not implemented |
| 0.4 | Zfinx version 0.4 |
| 0.41 | Zfinx version 0.41 |
| 1.0 | Zfinx version 1.0 |

Table 8.20: Values for Parameter Zfinx_version

8.2 Parameter values and limits

These are the formal parameter limits and actual parameter values

| Name | Min | Max | Default | Actual |
|-----------------------------|-----|-----------|----------|----------|
| Fundamental | | | | |
| variant | | | | RV32GCP |
| user_version | | | 20191213 | 20191213 |
| priv_version | | | 1.12 | 1.12 |
| Smempmp_version | | | none | none |
| numHarts | 0 | 32 | 0 | 0 |
| endian | | | none | none |
| enable_expanded | | | f | f |
| endianFixed | | | f | f |
| misa_MXL | 1 | 2 | 1 | 1 |
| misa_Extensions | 0 | 0x3ffffff | 0x14912d | 0x14912d |
| add_Extensions | | | | |
| sub_Extensions | | | | |
| misa_Extensions_mask | 0 | 0x3ffffff | 0x912d | 0x912d |
| add_Extensions_mask | | | | |
| sub_Extensions_mask | | | | |
| add_implicit_Extensions | | | | |
| sub_implicit_Extensions | | | | |
| Compressed_Extension | | | | |
| compress_version | | | 1.0 | 1.0 |
| Zca | | | t | t |
| Zcb | | | f | f |
| Zcf | | | t | t |
| Zcmp | | | f | f |

| | | | | |
|------------------------------|---|----------------|------------|------------|
| Zcmt | | | f | f |
| Zcmop | | | f | f |
| DSP_Extension | | | | |
| dsp_version | | | 0.5.2 | 0.5.2 |
| Zpsfoperand | | | t | t |
| Debug_Extension | | | | |
| debug_version | | | 1.0-STABLE | 1.0-STABLE |
| debug_mode | | | none | none |
| Interrupts_Exceptions | | | | |
| rnmi_version | | | none | none |
| mtvec_is_ro | | | f | f |
| tvec_align | 0 | 0x10000 | 0 | 0 |
| ecode_mask | 0 | 0xffffffffffff | 0x7ffffff | 0x7ffffff |
| ecode_nmi | 0 | 0xffffffffffff | 0 | 0 |
| nmi_absent | | | f | f |
| nmi_is_latched | | | f | f |
| nmi_high_priority | | | f | f |
| nmi_update_mstatus | | | f | f |
| nmi_update_tcontrol | | | f | f |
| nmi_zero_mtval | | | f | f |
| mtval_is_ro | | | f | f |
| tval_zero | | | f | f |
| tval_zero_ebreak | | | f | f |
| tval_ii_code | | | t | t |
| trap_preserves_lr | | | f | f |
| xret_preserves_lr | | | f | f |
| reset_address | 0 | 0xffffffffffff | 0 | 0 |
| nmi_address | 0 | 0xffffffffffff | 0 | 0 |
| CLINT_address | 0 | 0xffffffffffff | 0 | 0 |
| local_int_num | 0 | 16 | 0 | 0 |
| unimp_int_mask | 0 | 0xffffffffffff | 0 | 0 |
| force_mideleg | 0 | 0xffffffffffff | 0 | 0 |
| force_sideleg | 0 | 0xffffffffffff | 0 | 0 |
| no_ideleg | 0 | 0xffffffffffff | 0 | 0 |
| no_e deleg | 0 | 0xffffffffffff | 0 | 0 |
| external_int_id | | | f | f |
| Floating_Point | | | | |
| fp16_version | | | none | none |
| mstatus_fs_mode | | | write_1 | write_1 |
| d_requires_f | | | f | f |
| enable_fflags_i | | | f | f |
| enable_DAZ | | | f | f |
| enable_FZ | | | f | f |
| mstatus_FS | 0 | 3 | 0 | 0 |
| Zfa | | | f | f |

| | | | | |
|---------------------------------|--------------|--------------|---------------|---------------|
| Zfh | | | f | f |
| Zfhmin | | | f | f |
| Zfbfmin | | | f | f |
| Zfinx_version | | | none | none |
| Memory | | | | |
| lr_sc_constraint | | | user1 | user1 |
| amo_constraint | | | user1 | user1 |
| push_pop_constraint | | | user2 | user2 |
| updatePTEA | | | f | f |
| updatePTED | | | f | f |
| unaligned_low_pri | | | f | f |
| unaligned | | | f | f |
| Zam | | | f | f |
| amo_aborts_lr_sc | | | f | f |
| ASID_bits | 0 | 9 | 9 | 9 |
| lr_sc_grain | 1 | 0x10000 | 1 | 1 |
| ignore_non_leaf_DAU | | | f | f |
| Sv_modes | 0 | 0xffff | 3 | 3 |
| Simulation Artifact | | | | |
| leaf_hart_prefix | | | hart | hart |
| mtime_counter | | | mtime_counter | mtime_counter |
| mtime_Hz | 0.000000e+00 | 1.000000e+09 | 1.000000e+06 | 1.000000e+06 |
| mtime_bits | 0 | 64 | 64 | 64 |
| use_hw_reg_names | | | f | f |
| no_pseudo_inst | | | f | f |
| show_c_prefix | | | f | f |
| ABI_d | | | t | t |
| verbose | | | f | f |
| traceVolatile | | | f | f |
| wfi_restart | | | f | f |
| enable_CSR_bus | | | f | f |
| CSR_remap | | | | |
| ASID_cache_size | 0 | 0x100 | 8 | 8 |
| Instruction_CSR_Behavior | | | | |
| wfi_is_nop | | | f | f |
| wfi_resume_not_trap | | | f | f |
| TW_time_limit | 0 | 0xffffffff | 0 | 0 |
| counteren_mask | 0 | 0xffffffff | 0xffffffff | 0xffffffff |
| scounteren_zero_mask | 0 | 0xffffffff | 0 | 0 |
| noinhibit_mask | 0 | 0xffffffff | 0 | 0 |
| cycle_undefined | | | f | f |
| mcycle_undefined | | | f | f |
| time_undefined | | | f | f |
| instret_undefined | | | f | f |
| minstret_undefined | | | f | f |

| | | | | |
|--------------------------|---|----------------|------------------|------------------|
| hpmcounter_undefined | | | f | f |
| mhpmcounter_undefined | | | f | f |
| CSR Masks | | | | |
| mtvec_mask | 0 | 0xffffffffffff | 0 | 0 |
| stvec_mask | 0 | 0xffffffffffff | 0 | 0 |
| jvt_mask | 0 | 0xffffffffffff | 0xffffffffffffc0 | 0xffffffffffffc0 |
| tdata1_mask | 0 | 0xffffffffffff | 0xffffffffffff | 0xffffffffffff |
| mip_mask | 0 | 0xffffffffffff | 0x337 | 0x337 |
| sip_mask | 0 | 0xffffffffffff | 0x103 | 0x103 |
| envcfg_mask | 0 | 0xffffffffffff | 0 | 0 |
| mtvec_sext | | | f | f |
| stvec_sext | | | f | f |
| Trigger | | | | |
| tdata2_undefined | | | f | f |
| tdata3_undefined | | | f | f |
| tinfo_undefined | | | f | f |
| tcontrol_undefined | | | f | f |
| mcontext_undefined | | | f | f |
| scontext_undefined | | | f | f |
| mscontext_undefined | | | f | f |
| amo_trigger | | | f | f |
| no_hit | | | f | f |
| no_sselect_2 | | | f | f |
| trigger_num | 0 | 255 | 4 | 4 |
| mask_tselect | | | f | f |
| tinfo | 0 | 0x100fff | 0x100807c | 0x100807c |
| trigger_match | 1 | 0xffff | 0x333f | 0x333f |
| mcontext_bits | 0 | 32 | 6 | 6 |
| scontext_bits | 0 | 32 | 16 | 16 |
| mvalue_bits | 0 | 6 | 6 | 6 |
| svalue_bits | 0 | 16 | 16 | 16 |
| mcontrol_maskmax | 0 | 63 | 63 | 63 |
| chain_tval | | | first | first |
| PMP Configuration | | | | |
| PMP_grain | 0 | 29 | 0 | 0 |
| PMP_registers | 0 | 64 | 16 | 16 |
| PMP_csrs | 0 | 64 | 0 | 0 |
| PMP_max_page | 0 | 0xffffffff | 0 | 0 |
| PMP_decompose | | | f | f |
| PMP_undefined | | | f | f |
| PMP_R0W1 | | | RWX_00X | RWX_00X |
| PMP_maskparams | | | f | f |
| PMP_initialparams | | | f | f |
| Other Extensions | | | | |
| Svnapot_page_mask | 0 | 0xffffffffffff | 0 | 0 |

| | | | | |
|-----------------------|---|----------------|------|------|
| Smstateen | | | f | f |
| Smcsrind | | | f | f |
| Sstc | | | f | f |
| Sscofpmf | | | f | f |
| Ssdbltrp | | | f | f |
| Smcntrpmf | | | f | f |
| Smcdeleg | | | f | f |
| Svpbmt | | | f | f |
| Ssnpm | | | none | none |
| Smnpm | | | none | none |
| Smmppm | | | none | none |
| Svinval | | | f | f |
| Svadu | | | f | f |
| Ssqosid | | | f | f |
| Zihintntl | | | f | f |
| Zicond | | | f | f |
| Zicsr | | | t | t |
| Zifencei | | | t | t |
| Zicbom | | | f | f |
| Zicbop | | | f | f |
| Zicboz | | | f | f |
| Zimop | | | f | f |
| Zicfiss | | | f | f |
| Zicfilp | | | f | f |
| Zaamo | | | t | t |
| Zalrsc | | | t | t |
| Zacas | | | f | f |
| Zabha | | | f | f |
| Zawrs | | | f | f |
| Zmmul | | | f | f |
| CSR Defaults | | | | |
| mvendorid | 0 | 0xffffffffffff | 0 | 0 |
| marchid | 0 | 0xffffffffffff | 0 | 0 |
| mimpid | 0 | 0xffffffffffff | 0 | 0 |
| mhartid | 0 | 0xffffffffffff | 0 | 0 |
| mconfigptr | 0 | 0xffffffffffff | 0 | 0 |
| mtvec | 0 | 0xffffffffffff | 0 | 0 |
| mseccfg | 0 | 0xffffffffffff | 0 | 0 |
| Fast Interrupt | | | | |
| CLICLEVELS | 0 | 0x100 | 0 | 0 |
| AIA Interrupts | | | | |
| Smaia | | | f | f |

Table 8.21: Parameter values and limits

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 |
| GenericNMI | 4294967295 | Generic NMI |

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 7 commands.

This level in the model hierarchy has 6 register groups:

| Group name | Registers |
|-------------------------------|-----------|
| Core | 33 |
| Floating_point | 32 |
| User_Control_and_Status | 68 |
| Supervisor_Control_and_Status | 13 |
| Machine_Control_and_Status | 199 |
| Integration_support | 64 |

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 debugflags

show or modify the processor debug flags

| Argument | Type | Description |
|----------|---------|---|
| -get | Boolean | print current processor flags value |
| -mask | Boolean | print valid debug flag bits |
| -set | Int32 | new processor flags (only flags 0x00000006 can be modified) |

Table 12.1: debugflags command arguments

12.1.2 dumpTLB

12.1.2.1 Argument description

Show TLB contents

12.1.3 getCSRIndex

Return index for a named CSR (or -1 if no matching CSR)

| Argument | Type | Description |
|----------|--------|-------------|
| -name | String | CSR name |

Table 12.2: getCSRIndex command arguments

12.1.4 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 12.3: isync command arguments

12.1.5 itrace

enable or disable instruction tracing

| Argument | Type | Description |
|-------------------|---------|---|
| -access | String | show memory accesses by this instruction. Argument can be any combination of X (execute), A (load or store access) and S (system) |
| -after | Uns64 | apply after this many instructions |
| -enable | Boolean | enable instruction tracing |
| -full | Boolean | turn on all trace features |
| -instructioncount | Boolean | include the instruction number in each trace |
| -memory | String | (Alias for access). show memory accesses by this instruction. Argument can be any combination of X (execute), A (load or store access) and S (system) |
| -mode | Boolean | show processor mode changes |
| -off | Boolean | disable instruction tracing |
| -on | Boolean | enable instruction tracing |
| -processorname | Boolean | Include processor name in all trace lines |
| -registerchange | Boolean | show registers changed by this instruction |
| -registers | Boolean | show registers after each trace |

Table 12.4: itrace command arguments

12.1.6 listCSRs

12.1.6.1 Argument description

List all CSRs in index order

12.1.7 setPMA

Set PMA region permissions and legal access sizes

| Argument | Type | Description |
|-------------|--------|--|
| -attributes | String | region attributes (string containing r, w, x, a, A, P, 1, 2, 4 or 8) |
| -hi | Uns64 | high address |
| -lo | Uns64 | low address |

Table 12.5: setPMA 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 | 32 | 0 | r- | |
| ra | 32 | 0 | rw | |
| sp | 32 | 0 | rw | stack pointer |
| gp | 32 | 0 | rw | |
| tp | 32 | 0 | rw | |
| t0 | 32 | 0 | rw | |
| t1 | 32 | 0 | rw | |
| t2 | 32 | 0 | rw | |
| s0 | 32 | 0 | rw | |
| s1 | 32 | 0 | rw | |
| a0 | 32 | 0 | rw | |
| a1 | 32 | 0 | rw | |
| a2 | 32 | 0 | rw | |
| a3 | 32 | 0 | rw | |
| a4 | 32 | 0 | rw | |
| a5 | 32 | 0 | rw | |
| a6 | 32 | 0 | rw | |
| a7 | 32 | 0 | rw | |
| s2 | 32 | 0 | rw | |
| s3 | 32 | 0 | rw | |
| s4 | 32 | 0 | rw | |
| s5 | 32 | 0 | rw | |
| s6 | 32 | 0 | rw | |
| s7 | 32 | 0 | rw | |
| s8 | 32 | 0 | rw | |
| s9 | 32 | 0 | rw | |
| s10 | 32 | 0 | rw | |
| s11 | 32 | 0 | rw | |
| t3 | 32 | 0 | rw | |
| t4 | 32 | 0 | rw | |
| t5 | 32 | 0 | rw | |
| t6 | 32 | 0 | rw | |
| pc | 32 | 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 User_Control_and_Status

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

| Name | Bits | Initial-Hex | RW | Description |
|---------|------|-------------|----|-----------------------------------|
| fflags | 32 | 0 | rw | Floating-Point Flags |
| frm | 32 | 0 | rw | Floating-Point Rounding Mode |
| fcsr | 32 | 0 | rw | Floating-Point Control and Status |
| ucode | 32 | 0 | rw | Code |
| cycle | 32 | 0 | r- | Cycle Counter |
| time | 32 | 0 | r- | Timer |
| instret | 32 | 0 | r- | Instructions Retired |

| | | | | |
|---------------|----|---|----|--------------------------------|
| hpmcounter3 | 32 | 0 | r- | Performance Monitor Counter 3 |
| hpmcounter4 | 32 | 0 | r- | Performance Monitor Counter 4 |
| hpmcounter5 | 32 | 0 | r- | Performance Monitor Counter 5 |
| hpmcounter6 | 32 | 0 | r- | Performance Monitor Counter 6 |
| hpmcounter7 | 32 | 0 | r- | Performance Monitor Counter 7 |
| hpmcounter8 | 32 | 0 | r- | Performance Monitor Counter 8 |
| hpmcounter9 | 32 | 0 | r- | Performance Monitor Counter 9 |
| hpmcounter10 | 32 | 0 | r- | Performance Monitor Counter 10 |
| hpmcounter11 | 32 | 0 | r- | Performance Monitor Counter 11 |
| hpmcounter12 | 32 | 0 | r- | Performance Monitor Counter 12 |
| hpmcounter13 | 32 | 0 | r- | Performance Monitor Counter 13 |
| hpmcounter14 | 32 | 0 | r- | Performance Monitor Counter 14 |
| hpmcounter15 | 32 | 0 | r- | Performance Monitor Counter 15 |
| hpmcounter16 | 32 | 0 | r- | Performance Monitor Counter 16 |
| hpmcounter17 | 32 | 0 | r- | Performance Monitor Counter 17 |
| hpmcounter18 | 32 | 0 | r- | Performance Monitor Counter 18 |
| hpmcounter19 | 32 | 0 | r- | Performance Monitor Counter 19 |
| hpmcounter20 | 32 | 0 | r- | Performance Monitor Counter 20 |
| hpmcounter21 | 32 | 0 | r- | Performance Monitor Counter 21 |
| hpmcounter22 | 32 | 0 | r- | Performance Monitor Counter 22 |
| hpmcounter23 | 32 | 0 | r- | Performance Monitor Counter 23 |
| hpmcounter24 | 32 | 0 | r- | Performance Monitor Counter 24 |
| hpmcounter25 | 32 | 0 | r- | Performance Monitor Counter 25 |
| hpmcounter26 | 32 | 0 | r- | Performance Monitor Counter 26 |
| hpmcounter27 | 32 | 0 | r- | Performance Monitor Counter 27 |
| hpmcounter28 | 32 | 0 | r- | Performance Monitor Counter 28 |
| hpmcounter29 | 32 | 0 | r- | Performance Monitor Counter 29 |
| hpmcounter30 | 32 | 0 | r- | Performance Monitor Counter 30 |
| hpmcounter31 | 32 | 0 | r- | Performance Monitor Counter 31 |
| cycleh | 32 | 0 | r- | Cycle Counter High |
| timeh | 32 | 0 | r- | Timer High |
| instreth | 32 | 0 | r- | Instructions Retired High |
| hpmcounter3h | 32 | 0 | r- | Performance Monitor High 3 |
| hpmcounter4h | 32 | 0 | r- | Performance Monitor High 4 |
| hpmcounter5h | 32 | 0 | r- | Performance Monitor High 5 |
| hpmcounter6h | 32 | 0 | r- | Performance Monitor High 6 |
| hpmcounter7h | 32 | 0 | r- | Performance Monitor High 7 |
| hpmcounter8h | 32 | 0 | r- | Performance Monitor High 8 |
| hpmcounter9h | 32 | 0 | r- | Performance Monitor High 9 |
| hpmcounter10h | 32 | 0 | r- | Performance Monitor High 10 |
| hpmcounter11h | 32 | 0 | r- | Performance Monitor High 11 |
| hpmcounter12h | 32 | 0 | r- | Performance Monitor High 12 |
| hpmcounter13h | 32 | 0 | r- | Performance Monitor High 13 |
| hpmcounter14h | 32 | 0 | r- | Performance Monitor High 14 |
| hpmcounter15h | 32 | 0 | r- | Performance Monitor High 15 |
| hpmcounter16h | 32 | 0 | r- | Performance Monitor High 16 |
| hpmcounter17h | 32 | 0 | r- | Performance Monitor High 17 |
| hpmcounter18h | 32 | 0 | r- | Performance Monitor High 18 |
| hpmcounter19h | 32 | 0 | r- | Performance Monitor High 19 |
| hpmcounter20h | 32 | 0 | r- | Performance Monitor High 20 |
| hpmcounter21h | 32 | 0 | r- | Performance Monitor High 21 |
| hpmcounter22h | 32 | 0 | r- | Performance Monitor High 22 |
| hpmcounter23h | 32 | 0 | r- | Performance Monitor High 23 |
| hpmcounter24h | 32 | 0 | r- | Performance Monitor High 24 |
| hpmcounter25h | 32 | 0 | r- | Performance Monitor High 25 |
| hpmcounter26h | 32 | 0 | r- | Performance Monitor High 26 |

| | | | | |
|---------------|----|---|----|-----------------------------|
| hpmcounter27h | 32 | 0 | r- | Performance Monitor High 27 |
| hpmcounter28h | 32 | 0 | r- | Performance Monitor High 28 |
| hpmcounter29h | 32 | 0 | r- | Performance Monitor High 29 |
| hpmcounter30h | 32 | 0 | r- | Performance Monitor High 30 |
| hpmcounter31h | 32 | 0 | r- | Performance Monitor High 31 |

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

13.1.4 Supervisor_Control_and_Status

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

| Name | Bits | Initial-Hex | RW | Description |
|------------|------|-------------|----|---|
| sstatus | 32 | 0 | rw | Supervisor Status |
| sie | 32 | 0 | rw | Supervisor Interrupt Enable |
| stvec | 32 | 0 | rw | Supervisor Trap-Vector Base-Address |
| scounteren | 32 | 0 | rw | Supervisor Counter Enable |
| senvcfg | 32 | 0 | rw | Supervisor Environment Configuration |
| sscratch | 32 | 0 | rw | Supervisor Scratch |
| sepc | 32 | 0 | rw | Supervisor Exception Program Counter |
| scause | 32 | 0 | rw | Supervisor Cause |
| stval | 32 | 0 | rw | Supervisor Trap Value |
| sip | 32 | 0 | rw | Supervisor Interrupt Pending |
| satp | 32 | 0 | rw | Supervisor Address Translation and Protection |
| scontext | 32 | 0 | rw | Trigger Supervisor Context |
| mscontext | 32 | 0 | rw | Trigger Machine Context Alias |

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

13.1.5 Machine_Control_and_Status

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

| Name | Bits | Initial-Hex | RW | Description |
|----------------|------|-------------|----|---|
| mstatus | 32 | 0 | rw | Machine Status |
| misa | 32 | 4014912d | rw | ISA and Extensions |
| medeleg | 32 | 0 | rw | Machine Exception Delegation |
| mideleg | 32 | 0 | rw | Machine Interrupt Delegation |
| mie | 32 | 0 | rw | Machine Interrupt Enable |
| mtvec | 32 | 0 | rw | Machine Trap-Vector Base-Address |
| mcounteren | 32 | 0 | rw | Machine Counter Enable |
| menvcfg | 32 | 0 | rw | Machine Environment Configuration |
| mstatush | 32 | 0 | rw | Machine Status High |
| menvcfgh | 32 | 0 | rw | Machine Environment Configuration High |
| mcounthinhibit | 32 | 0 | rw | Machine Counter Inhibit |
| mhpmevent3 | 32 | 0 | rw | Machine Performance Monitor Event Select 3 |
| mhpmevent4 | 32 | 0 | rw | Machine Performance Monitor Event Select 4 |
| mhpmevent5 | 32 | 0 | rw | Machine Performance Monitor Event Select 5 |
| mhpmevent6 | 32 | 0 | rw | Machine Performance Monitor Event Select 6 |
| mhpmevent7 | 32 | 0 | rw | Machine Performance Monitor Event Select 7 |
| mhpmevent8 | 32 | 0 | rw | Machine Performance Monitor Event Select 8 |
| mhpmevent9 | 32 | 0 | rw | Machine Performance Monitor Event Select 9 |
| mhpmevent10 | 32 | 0 | rw | Machine Performance Monitor Event Select 10 |
| mhpmevent11 | 32 | 0 | rw | Machine Performance Monitor Event Select 11 |
| mhpmevent12 | 32 | 0 | rw | Machine Performance Monitor Event Select 12 |

| | | | | |
|-------------|----|---|----|---|
| mhpmevent13 | 32 | 0 | rw | Machine Performance Monitor Event Select 13 |
| mhpmevent14 | 32 | 0 | rw | Machine Performance Monitor Event Select 14 |
| mhpmevent15 | 32 | 0 | rw | Machine Performance Monitor Event Select 15 |
| mhpmevent16 | 32 | 0 | rw | Machine Performance Monitor Event Select 16 |
| mhpmevent17 | 32 | 0 | rw | Machine Performance Monitor Event Select 17 |
| mhpmevent18 | 32 | 0 | rw | Machine Performance Monitor Event Select 18 |
| mhpmevent19 | 32 | 0 | rw | Machine Performance Monitor Event Select 19 |
| mhpmevent20 | 32 | 0 | rw | Machine Performance Monitor Event Select 20 |
| mhpmevent21 | 32 | 0 | rw | Machine Performance Monitor Event Select 21 |
| mhpmevent22 | 32 | 0 | rw | Machine Performance Monitor Event Select 22 |
| mhpmevent23 | 32 | 0 | rw | Machine Performance Monitor Event Select 23 |
| mhpmevent24 | 32 | 0 | rw | Machine Performance Monitor Event Select 24 |
| mhpmevent25 | 32 | 0 | rw | Machine Performance Monitor Event Select 25 |
| mhpmevent26 | 32 | 0 | rw | Machine Performance Monitor Event Select 26 |
| mhpmevent27 | 32 | 0 | rw | Machine Performance Monitor Event Select 27 |
| mhpmevent28 | 32 | 0 | rw | Machine Performance Monitor Event Select 28 |
| mhpmevent29 | 32 | 0 | rw | Machine Performance Monitor Event Select 29 |
| mhpmevent30 | 32 | 0 | rw | Machine Performance Monitor Event Select 30 |
| mhpmevent31 | 32 | 0 | rw | Machine Performance Monitor Event Select 31 |
| mscratch | 32 | 0 | rw | Machine Scratch |
| mepc | 32 | 0 | rw | Machine Exception Program Counter |
| mcause | 32 | 0 | rw | Machine Cause |
| mtval | 32 | 0 | rw | Machine Trap Value |
| mip | 32 | 0 | rw | Machine Interrupt Pending |
| pmpcfg0 | 32 | 0 | rw | Physical Memory Protection Configuration 0 |
| pmpcfg1 | 32 | 0 | rw | Physical Memory Protection Configuration 1 |
| pmpcfg2 | 32 | 0 | rw | Physical Memory Protection Configuration 2 |
| pmpcfg3 | 32 | 0 | rw | Physical Memory Protection Configuration 3 |
| pmpcfg4 | 32 | 0 | rw | Physical Memory Protection Configuration 4 |
| pmpcfg5 | 32 | 0 | rw | Physical Memory Protection Configuration 5 |
| pmpcfg6 | 32 | 0 | rw | Physical Memory Protection Configuration 6 |
| pmpcfg7 | 32 | 0 | rw | Physical Memory Protection Configuration 7 |
| pmpcfg8 | 32 | 0 | rw | Physical Memory Protection Configuration 8 |
| pmpcfg9 | 32 | 0 | rw | Physical Memory Protection Configuration 9 |
| pmpcfg10 | 32 | 0 | rw | Physical Memory Protection Configuration 10 |
| pmpcfg11 | 32 | 0 | rw | Physical Memory Protection Configuration 11 |
| pmpcfg12 | 32 | 0 | rw | Physical Memory Protection Configuration 12 |
| pmpcfg13 | 32 | 0 | rw | Physical Memory Protection Configuration 13 |
| pmpcfg14 | 32 | 0 | rw | Physical Memory Protection Configuration 14 |
| pmpcfg15 | 32 | 0 | rw | Physical Memory Protection Configuration 15 |
| pmpaddr0 | 32 | 0 | rw | Physical Memory Protection Address 0 |
| pmpaddr1 | 32 | 0 | rw | Physical Memory Protection Address 1 |
| pmpaddr2 | 32 | 0 | rw | Physical Memory Protection Address 2 |
| pmpaddr3 | 32 | 0 | rw | Physical Memory Protection Address 3 |
| pmpaddr4 | 32 | 0 | rw | Physical Memory Protection Address 4 |
| pmpaddr5 | 32 | 0 | rw | Physical Memory Protection Address 5 |
| pmpaddr6 | 32 | 0 | rw | Physical Memory Protection Address 6 |
| pmpaddr7 | 32 | 0 | rw | Physical Memory Protection Address 7 |
| pmpaddr8 | 32 | 0 | rw | Physical Memory Protection Address 8 |
| pmpaddr9 | 32 | 0 | rw | Physical Memory Protection Address 9 |
| pmpaddr10 | 32 | 0 | rw | Physical Memory Protection Address 10 |
| pmpaddr11 | 32 | 0 | rw | Physical Memory Protection Address 11 |
| pmpaddr12 | 32 | 0 | rw | Physical Memory Protection Address 12 |
| pmpaddr13 | 32 | 0 | rw | Physical Memory Protection Address 13 |
| pmpaddr14 | 32 | 0 | rw | Physical Memory Protection Address 14 |
| pmpaddr15 | 32 | 0 | rw | Physical Memory Protection Address 15 |

| | | | | |
|-----------|----|----------|----|---------------------------------------|
| pmpaddr16 | 32 | 0 | rw | Physical Memory Protection Address 16 |
| pmpaddr17 | 32 | 0 | rw | Physical Memory Protection Address 17 |
| pmpaddr18 | 32 | 0 | rw | Physical Memory Protection Address 18 |
| pmpaddr19 | 32 | 0 | rw | Physical Memory Protection Address 19 |
| pmpaddr20 | 32 | 0 | rw | Physical Memory Protection Address 20 |
| pmpaddr21 | 32 | 0 | rw | Physical Memory Protection Address 21 |
| pmpaddr22 | 32 | 0 | rw | Physical Memory Protection Address 22 |
| pmpaddr23 | 32 | 0 | rw | Physical Memory Protection Address 23 |
| pmpaddr24 | 32 | 0 | rw | Physical Memory Protection Address 24 |
| pmpaddr25 | 32 | 0 | rw | Physical Memory Protection Address 25 |
| pmpaddr26 | 32 | 0 | rw | Physical Memory Protection Address 26 |
| pmpaddr27 | 32 | 0 | rw | Physical Memory Protection Address 27 |
| pmpaddr28 | 32 | 0 | rw | Physical Memory Protection Address 28 |
| pmpaddr29 | 32 | 0 | rw | Physical Memory Protection Address 29 |
| pmpaddr30 | 32 | 0 | rw | Physical Memory Protection Address 30 |
| pmpaddr31 | 32 | 0 | rw | Physical Memory Protection Address 31 |
| pmpaddr32 | 32 | 0 | rw | Physical Memory Protection Address 32 |
| pmpaddr33 | 32 | 0 | rw | Physical Memory Protection Address 33 |
| pmpaddr34 | 32 | 0 | rw | Physical Memory Protection Address 34 |
| pmpaddr35 | 32 | 0 | rw | Physical Memory Protection Address 35 |
| pmpaddr36 | 32 | 0 | rw | Physical Memory Protection Address 36 |
| pmpaddr37 | 32 | 0 | rw | Physical Memory Protection Address 37 |
| pmpaddr38 | 32 | 0 | rw | Physical Memory Protection Address 38 |
| pmpaddr39 | 32 | 0 | rw | Physical Memory Protection Address 39 |
| pmpaddr40 | 32 | 0 | rw | Physical Memory Protection Address 40 |
| pmpaddr41 | 32 | 0 | rw | Physical Memory Protection Address 41 |
| pmpaddr42 | 32 | 0 | rw | Physical Memory Protection Address 42 |
| pmpaddr43 | 32 | 0 | rw | Physical Memory Protection Address 43 |
| pmpaddr44 | 32 | 0 | rw | Physical Memory Protection Address 44 |
| pmpaddr45 | 32 | 0 | rw | Physical Memory Protection Address 45 |
| pmpaddr46 | 32 | 0 | rw | Physical Memory Protection Address 46 |
| pmpaddr47 | 32 | 0 | rw | Physical Memory Protection Address 47 |
| pmpaddr48 | 32 | 0 | rw | Physical Memory Protection Address 48 |
| pmpaddr49 | 32 | 0 | rw | Physical Memory Protection Address 49 |
| pmpaddr50 | 32 | 0 | rw | Physical Memory Protection Address 50 |
| pmpaddr51 | 32 | 0 | rw | Physical Memory Protection Address 51 |
| pmpaddr52 | 32 | 0 | rw | Physical Memory Protection Address 52 |
| pmpaddr53 | 32 | 0 | rw | Physical Memory Protection Address 53 |
| pmpaddr54 | 32 | 0 | rw | Physical Memory Protection Address 54 |
| pmpaddr55 | 32 | 0 | rw | Physical Memory Protection Address 55 |
| pmpaddr56 | 32 | 0 | rw | Physical Memory Protection Address 56 |
| pmpaddr57 | 32 | 0 | rw | Physical Memory Protection Address 57 |
| pmpaddr58 | 32 | 0 | rw | Physical Memory Protection Address 58 |
| pmpaddr59 | 32 | 0 | rw | Physical Memory Protection Address 59 |
| pmpaddr60 | 32 | 0 | rw | Physical Memory Protection Address 60 |
| pmpaddr61 | 32 | 0 | rw | Physical Memory Protection Address 61 |
| pmpaddr62 | 32 | 0 | rw | Physical Memory Protection Address 62 |
| pmpaddr63 | 32 | 0 | rw | Physical Memory Protection Address 63 |
| tselect | 32 | 0 | rw | Trigger Register Select |
| tdata1 | 32 | f0000000 | rw | Trigger Data 1 |
| tdata2 | 32 | 0 | rw | Trigger Data 2 |
| tdata3 | 32 | 0 | rw | Trigger Data 3 |
| tinfo | 32 | 100807c | rw | Trigger Info |
| tcontrol | 32 | 0 | rw | Trigger Control |
| mcontext | 32 | 0 | rw | Trigger Machine Context |
| mcycle | 32 | 0 | rw | Machine Cycle Counter |

| | | | | |
|-----------------|----|---|----|---|
| minstret | 32 | 0 | rw | Machine Instructions Retired |
| mhpmpcounter3 | 32 | 0 | rw | Machine Performance Monitor Counter 3 |
| mhpmpcounter4 | 32 | 0 | rw | Machine Performance Monitor Counter 4 |
| mhpmpcounter5 | 32 | 0 | rw | Machine Performance Monitor Counter 5 |
| mhpmpcounter6 | 32 | 0 | rw | Machine Performance Monitor Counter 6 |
| mhpmpcounter7 | 32 | 0 | rw | Machine Performance Monitor Counter 7 |
| mhpmpcounter8 | 32 | 0 | rw | Machine Performance Monitor Counter 8 |
| mhpmpcounter9 | 32 | 0 | rw | Machine Performance Monitor Counter 9 |
| mhpmpcounter10 | 32 | 0 | rw | Machine Performance Monitor Counter 10 |
| mhpmpcounter11 | 32 | 0 | rw | Machine Performance Monitor Counter 11 |
| mhpmpcounter12 | 32 | 0 | rw | Machine Performance Monitor Counter 12 |
| mhpmpcounter13 | 32 | 0 | rw | Machine Performance Monitor Counter 13 |
| mhpmpcounter14 | 32 | 0 | rw | Machine Performance Monitor Counter 14 |
| mhpmpcounter15 | 32 | 0 | rw | Machine Performance Monitor Counter 15 |
| mhpmpcounter16 | 32 | 0 | rw | Machine Performance Monitor Counter 16 |
| mhpmpcounter17 | 32 | 0 | rw | Machine Performance Monitor Counter 17 |
| mhpmpcounter18 | 32 | 0 | rw | Machine Performance Monitor Counter 18 |
| mhpmpcounter19 | 32 | 0 | rw | Machine Performance Monitor Counter 19 |
| mhpmpcounter20 | 32 | 0 | rw | Machine Performance Monitor Counter 20 |
| mhpmpcounter21 | 32 | 0 | rw | Machine Performance Monitor Counter 21 |
| mhpmpcounter22 | 32 | 0 | rw | Machine Performance Monitor Counter 22 |
| mhpmpcounter23 | 32 | 0 | rw | Machine Performance Monitor Counter 23 |
| mhpmpcounter24 | 32 | 0 | rw | Machine Performance Monitor Counter 24 |
| mhpmpcounter25 | 32 | 0 | rw | Machine Performance Monitor Counter 25 |
| mhpmpcounter26 | 32 | 0 | rw | Machine Performance Monitor Counter 26 |
| mhpmpcounter27 | 32 | 0 | rw | Machine Performance Monitor Counter 27 |
| mhpmpcounter28 | 32 | 0 | rw | Machine Performance Monitor Counter 28 |
| mhpmpcounter29 | 32 | 0 | rw | Machine Performance Monitor Counter 29 |
| mhpmpcounter30 | 32 | 0 | rw | Machine Performance Monitor Counter 30 |
| mhpmpcounter31 | 32 | 0 | rw | Machine Performance Monitor Counter 31 |
| mcycleh | 32 | 0 | rw | Machine Cycle Counter High |
| minstreth | 32 | 0 | rw | Machine Instructions Retired High |
| mhpmpcounter3h | 32 | 0 | rw | Machine Performance Monitor Counter High 3 |
| mhpmpcounter4h | 32 | 0 | rw | Machine Performance Monitor Counter High 4 |
| mhpmpcounter5h | 32 | 0 | rw | Machine Performance Monitor Counter High 5 |
| mhpmpcounter6h | 32 | 0 | rw | Machine Performance Monitor Counter High 6 |
| mhpmpcounter7h | 32 | 0 | rw | Machine Performance Monitor Counter High 7 |
| mhpmpcounter8h | 32 | 0 | rw | Machine Performance Monitor Counter High 8 |
| mhpmpcounter9h | 32 | 0 | rw | Machine Performance Monitor Counter High 9 |
| mhpmpcounter10h | 32 | 0 | rw | Machine Performance Monitor Counter High 10 |
| mhpmpcounter11h | 32 | 0 | rw | Machine Performance Monitor Counter High 11 |
| mhpmpcounter12h | 32 | 0 | rw | Machine Performance Monitor Counter High 12 |
| mhpmpcounter13h | 32 | 0 | rw | Machine Performance Monitor Counter High 13 |
| mhpmpcounter14h | 32 | 0 | rw | Machine Performance Monitor Counter High 14 |
| mhpmpcounter15h | 32 | 0 | rw | Machine Performance Monitor Counter High 15 |
| mhpmpcounter16h | 32 | 0 | rw | Machine Performance Monitor Counter High 16 |
| mhpmpcounter17h | 32 | 0 | rw | Machine Performance Monitor Counter High 17 |
| mhpmpcounter18h | 32 | 0 | rw | Machine Performance Monitor Counter High 18 |
| mhpmpcounter19h | 32 | 0 | rw | Machine Performance Monitor Counter High 19 |
| mhpmpcounter20h | 32 | 0 | rw | Machine Performance Monitor Counter High 20 |
| mhpmpcounter21h | 32 | 0 | rw | Machine Performance Monitor Counter High 21 |
| mhpmpcounter22h | 32 | 0 | rw | Machine Performance Monitor Counter High 22 |
| mhpmpcounter23h | 32 | 0 | rw | Machine Performance Monitor Counter High 23 |
| mhpmpcounter24h | 32 | 0 | rw | Machine Performance Monitor Counter High 24 |
| mhpmpcounter25h | 32 | 0 | rw | Machine Performance Monitor Counter High 25 |
| mhpmpcounter26h | 32 | 0 | rw | Machine Performance Monitor Counter High 26 |

| | | | | |
|----------------|----|---|----|---|
| mhpmcounter27h | 32 | 0 | rw | Machine Performance Monitor Counter High 27 |
| mhpmcounter28h | 32 | 0 | rw | Machine Performance Monitor Counter High 28 |
| mhpmcounter29h | 32 | 0 | rw | Machine Performance Monitor Counter High 29 |
| mhpmcounter30h | 32 | 0 | rw | Machine Performance Monitor Counter High 30 |
| mhpmcounter31h | 32 | 0 | rw | Machine Performance Monitor Counter High 31 |
| mvendorid | 32 | 0 | r- | Vendor ID |
| marchid | 32 | 0 | r- | Architecture ID |
| mimpid | 32 | 0 | r- | Implementation ID |
| mhartid | 32 | 0 | r- | Hardware Thread ID |
| mconfigptr | 32 | 0 | r- | Configuration Data Structure |

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

13.1.6 Integration_support

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

| Name | Bits | Initial-Hex | RW | Description |
|-----------------|------|----------------|----|---|
| LRSCAddress | 32 | ffffff | rw | LR/SC active lock address |
| commercial | 8 | 0 | r- | Commercial feature in use |
| PTWStage | 8 | 0 | r- | PTW active stage from memory callback context (0:none 1:HS 2:VS 3:G) |
| PTWInputAddr | 64 | 0 | r- | PTW input address from memory callback context |
| PTWLevel | 8 | 0 | r- | PTW active level from memory callback context |
| ASYNCPE | 8 | 0 | r- | Asynchronous Event Pending & Enabled |
| mask_pmpcfg0 | 64 | ffffff fffffff | r- | Write mask for pmpcfg0 |
| mask_pmpcfg1 | 32 | ffffff | r- | Write mask for pmpcfg1 |
| mask_pmpcfg2 | 64 | ffffff fffffff | r- | Write mask for pmpcfg2 |
| mask_pmpcfg3 | 32 | ffffff | r- | Write mask for pmpcfg3 |
| mask_pmpaddr0 | 64 | ffffff fffffff | r- | Write mask for pmpaddr0 |
| mask_pmpaddr1 | 64 | ffffff fffffff | r- | Write mask for pmpaddr1 |
| mask_pmpaddr2 | 64 | ffffff fffffff | r- | Write mask for pmpaddr2 |
| mask_pmpaddr3 | 64 | ffffff fffffff | r- | Write mask for pmpaddr3 |
| mask_pmpaddr4 | 64 | ffffff fffffff | r- | Write mask for pmpaddr4 |
| mask_pmpaddr5 | 64 | ffffff fffffff | r- | Write mask for pmpaddr5 |
| mask_pmpaddr6 | 64 | ffffff fffffff | r- | Write mask for pmpaddr6 |
| mask_pmpaddr7 | 64 | ffffff fffffff | r- | Write mask for pmpaddr7 |
| mask_pmpaddr8 | 64 | ffffff fffffff | r- | Write mask for pmpaddr8 |
| mask_pmpaddr9 | 64 | ffffff fffffff | r- | Write mask for pmpaddr9 |
| mask_pmpaddr10 | 64 | ffffff fffffff | r- | Write mask for pmpaddr10 |
| mask_pmpaddr11 | 64 | ffffff fffffff | r- | Write mask for pmpaddr11 |
| mask_pmpaddr12 | 64 | ffffff fffffff | r- | Write mask for pmpaddr12 |
| mask_pmpaddr13 | 64 | ffffff fffffff | r- | Write mask for pmpaddr13 |
| mask_pmpaddr14 | 64 | ffffff fffffff | r- | Write mask for pmpaddr14 |
| mask_pmpaddr15 | 64 | ffffff fffffff | r- | Write mask for pmpaddr15 |
| PTWBankSelect | 8 | 0 | rw | select PTW bank (0 is stage 1, 1 is stage 2, 2-6 are stage 2 walks initiated by stage 1 level 0-4 entry lookups, respectively) |
| PTWBankValid | 8 | 0 | r- | bitmask of valid banks (0x01 is stage 1, 0x02 is stage 2, 0x04-0x40 are stage 2 walks initiated by stage 1 level 0-4 entry lookups, respectively) |
| PTWAddressValid | 8 | 0 | r- | bitmask of valid bits for each of PTWAddressL0...PTWAddressL4, PTWBase, PTWInput and PTWOutput in current bank |
| PTWValueValid | 8 | 0 | r- | bitmask of valid bits for each of PTWValueL0...PTWValueL4 in current bank |
| PTWAddressL0 | 64 | 0 | r- | current bank PTW address, level 0 |

| | | | | |
|--------------|----|---|----|---|
| PTWAddressL1 | 64 | 0 | r- | current bank PTW address, level 1 |
| PTWAddressL2 | 64 | 0 | r- | current bank PTW address, level 2 |
| PTWAddressL3 | 64 | 0 | r- | current bank PTW address, level 3 |
| PTWAddressL4 | 64 | 0 | r- | current bank PTW address, level 4 |
| PTWValueL0 | 64 | 0 | r- | current bank PTW value, level 0 |
| PTWValueL1 | 64 | 0 | r- | current bank PTW value, level 1 |
| PTWValueL2 | 64 | 0 | r- | current bank PTW value, level 2 |
| PTWValueL3 | 64 | 0 | r- | current bank PTW value, level 3 |
| PTWValueL4 | 64 | 0 | r- | current bank PTW value, level 4 |
| PTWBase | 64 | 0 | r- | current bank PTW table base address |
| PTWInput | 64 | 0 | r- | current bank PTW input address |
| PTWOutput | 64 | 0 | r- | current bank PTW output address |
| PTWPgSize | 64 | 0 | r- | current bank PTW page size (Valid only when PTWOutput is valid) |
| PTWUD | 64 | 0 | -w | perform [H]U-mode page table walk for the given data address |
| PTWUX | 64 | 0 | -w | perform [H]U-mode page table walk for the given execute address |
| PTWSD | 64 | 0 | -w | perform [H]S-mode page table walk for the given data address |
| PTWSX | 64 | 0 | -w | perform [H]S-mode page table walk for the given execute address |
| pmp0cfg0 | 8 | 0 | r- | |
| pmp1cfg0 | 8 | 0 | r- | |
| pmp2cfg0 | 8 | 0 | r- | |
| pmp3cfg0 | 8 | 0 | r- | |
| pmp4cfg1 | 8 | 0 | r- | |
| pmp5cfg1 | 8 | 0 | r- | |
| pmp6cfg1 | 8 | 0 | r- | |
| pmp7cfg1 | 8 | 0 | r- | |
| pmp8cfg2 | 8 | 0 | r- | |
| pmp9cfg2 | 8 | 0 | r- | |
| pmp10cfg2 | 8 | 0 | r- | |
| pmp11cfg2 | 8 | 0 | r- | |
| pmp12cfg3 | 8 | 0 | r- | |
| pmp13cfg3 | 8 | 0 | r- | |
| pmp14cfg3 | 8 | 0 | r- | |
| pmp15cfg3 | 8 | 0 | r- | |

Table 13.6: Registers at level 1, type:Hart group:Integration support