
# OVP Guide to Using Processor Models

## Model specific information for SiFive_E20

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


| | |
|---|---|
| Author | Imperas Software Limited |
| Version | 20211118.0 |
| Filename | OVP_Model_Specific_Information_sifive_riscv_E20.pdf |
| Created | 31 December 2021 |
| Status | OVP Standard Release |

## Copyright Notice



## Right to Copy Documentation



## Destination Control Statement



## Disclaimer



## Model Release Status

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

# Contents

# 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 E20 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 1: extension B (bit manipulation extension)

misa bit 2: extension C (compressed instructions)

misa bit 8: RV32I/RV64I/RV128I base integer instruction set

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

To specify features that can be dynamically enabled or disabled by writes to the misa register in addition to those listed above, use parameter "add_Extensions_mask". This is a string parameter containing the feature letters to add; for example, value "DV" indicates that double-precision floating point and the Vector Extension can be enabled or disabled by writes to the misa register, 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 extensions are supported by the model, but not enabled by default in this variant:

misa bit 0: extension A (atomic instructions)

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

misa bit 20: extension U (User mode)

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.

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 extensions are enabled by default in the model and can be disabled:

misa bit 1: extension B (bit manipulation extension)

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

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.

To remove features from this list from the implicitly-enabled set (not visible in the misa register), use parameter "sub_implicit_Extensions". This is a string parameter in the same format as the "sub_Extensions" parameter described above.

## 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 0xffffffff. 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 64.

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 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.

### 1.4.3 NMI

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 0x2 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".

### 1.4.4 WFI

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

### 1.4.5 cycle CSR

The "cycle" CSR is implemented in this variant. Set parameter "cycle_undefined" to True to instead specify that "cycle" is unimplemented and reads of it should cause Illegal Instruction traps.

### 1.4.6 time CSR

The "time" CSR is not implemented in this variant and reads of it will cause Illegal Instruction traps. Set parameter "time_undefined" to False to instead specify that "time" is implemented.

### 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 reads of it should cause Illegal Instruction traps.

### 1.4.8 hpmcounter CSRs

"hpmcounter" CSRs are implemented in this variant. Set parameter "hpmcounter_undefined" to True to instead specify that "hpmcounter" CSRs are unimplemented and reads of them should cause Illegal Instruction traps.

### 1.4.9 Unaligned Accesses

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

### 1.4.10 PMP

A PMP unit is not implemented by this variant. Set parameter "PMP_registers" to indicate that the unit should be implemented with that number of PMP entries.

## 1.5 Compressed Extension

Standard compressed instructions are present in this variant.

Parameter Zcea_version is used to specify the version of Zcea instructions present. By default, Zcea_version is set to "none" in this variant. Updates to this parameter require a commercial product license.

Parameter Zceb_version is used to specify the version of Zceb instructions present. By default, Zceb_version is set to "none" in this variant. Updates to this parameter require a commercial product license.

Parameter Zcee_version is used to specify the version of Zcee instructions present. By default, Zcee_version is set to "none" in this variant. Updates to this parameter require a commercial product license.

## 1.6 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.

### 1.6.1 Legacy Version 1.10

1.10 version of May 7 2017.

### 1.6.2 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);

### 1.6.3 Version master

Unstable master version corresponding to evolving 1.12 specification, 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 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.

### 1.7.1 Legacy Version 2.2

2.2 version of May 7 2017.

### 1.7.2 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.

## 1.8 Bit-Manipulation Extension

This variant implements the Bit-Manipulation extension with version specified in the References section of this document. Note that parameter "bitmanip_version" can be used to select the required version of this extension. See section "Bit-Manipulation Extension Versions" for detailed information about differences between each supported version.

### 1.8.1 Bit-Manipulation Extension Parameters

Parameter Zbb is used to specify that the base instructions are present. By default, Zbb is set to 1 in this variant. Updates to this parameter require a commercial product license.

Parameter Zba is used to specify that address calculation instructions are present. By default, Zba is set to 1 in this variant. Updates to this parameter require a commercial product license.

Parameter Zbc is used to specify that carryless operation instructions are present. By default, Zbc is set to 0 in this variant. Updates to this parameter require a commercial product license.

Parameter Zbe is used to specify that bit deposit/extract instructions are present. By default, Zbe is set to 0 in this variant. Updates to this parameter require a commercial product license. This parameter is ignored for version 1.0.0, which does not implement that subset.

Parameter Zbf is used to specify that bit field place instructions are present. By default, Zbf is set to 0 in this variant. Updates to this parameter require a commercial product license. This parameter is ignored for version 1.0.0, which does not implement that subset.

Parameter Zbm is used to specify that bit matrix operation instructions are present. By default, Zbm is set to 0 in this variant. Updates to this parameter require a commercial product license. This parameter is ignored for version 1.0.0, which does not implement that subset.

Parameter Zbp is used to specify that permutation instructions are present. By default, Zbp is set to 0 in this variant. Updates to this parameter require a commercial product license. This parameter is ignored for version 1.0.0, which does not implement that subset.

Parameter Zbr is used to specify that CRC32 instructions are present. By default, Zbr is set to 0 in this variant. Updates to this parameter require a commercial product license. This parameter is ignored for version 1.0.0, which does not implement that subset.

Parameter Zbs is used to specify that single bit instructions are present. By default, Zbs is set to 0 in this variant. Updates to this parameter require a commercial product license.

Parameter Zbt is used to specify that ternary instructions are present. By default, Zbt is set to 0 in this variant. Updates to this parameter require a commercial product license. This parameter is ignored for version 1.0.0, which does not implement that subset.

### 1.8.2 Bit-Manipulation Extension Versions

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

### 1.8.3 Version 0.90

Stable 0.90 version of June 10 2019.

### 1.8.4 Version 0.91

Stable 0.91 version of August 29 2019, with these changes compared to version 0.90:

- change encodings of bmatxor, grev, grevw, grevi and greviw;

- add gorc, gorcw, gorci, gorciw, bfp and bfpw instructions.

### 1.8.5 Version 0.92

Stable 0.92 version of November 8 2019, with these changes compared to version 0.91:

- add packh, packu and packuw instructions;

- add sext.b and sext.h instructions;

- change encoding and behavior of bfp and bfpw instructions;

- change encoding of bdep and bdepw instructions.

### 1.8.6 Version 0.93-draft

Draft 0.93 version of January 29 2020, with these changes compared to version 0.92:

- add sh1add, sh2add, sh3add, sh1addu, sh2addu and sh3addu instructions;

- move slo, sloi, sro and sroi to Zbp subset;

- add orc16 to Zbb subset.

### 1.8.7 Version 0.93

Stable 0.93 version of January 10 2021, with these changes compared to version 0.93-draft:

- assignments of instructions to Z extension groups changed;

- exchange encodings of max and minu instructions;

- add xperm.[nbhw] instructions;

- instructions named *u.w renamed to *.uw;

- instructions named sb* renamed to b*;

- instructions named pcnt* renamed to cpop*;

- instructions subu.w, addiwu, addwu, subwu, clmulw, clmulrw and clmulhw removed;

- instructions slo, sro, sloi, sroi, slow, srow, sloiw and sroiw removed from all Z extension groups and are therefore never implemented;

- instructions bext/bdep renamed to bcompress/bdecompress (this change is documented under the draft 0.94 version but is required to resolve an instruction name conflict introduced by instruction renames above);

### 1.8.8 Version 0.94

Stable 0.94 version of January 20 2021, with these changes compared to version 0.93:

- instructions bset[i]w, bclr[i]w, binv[i]w and bextw removed.

### 1.8.9 Version 1.0.0

Stable 1.0.0 version of June 6 2021, with these changes compared to version 0.94:

- instructions with immediate shift operands now follow base architecture semantics to determine operand legality instead of masking to XLEN-1;

- only subsets Zba, Zbb, Zbc and Zbs may be enabled;

- if the B extension is present, it is implicitly always enabled and not subject to control by misa.B, which is zero.

### 1.8.10 Version master

Unstable master version, with these changes compared to version 1.0.0:

- any subset may be enabled;

- xperm.n, xperm.b, xperm.h and xperm.w instructions renamed xperm4, xperm8, xperm16 and xperm32.

## 1.9 Other Extensions

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

### 1.9.1 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.

### 1.9.2 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.

### 1.9.3 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.

### 1.9.4 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.

### 1.9.5 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).

### 1.9.6 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 CLIC

This model implements a Core Local Interrupt Controller (CLIC) with 16 levels. Parameter "CLICLEVELS" can be used to specify a different number of interrupt levels (2-256), as described in the RISC-V Core-Local Interrupt Controller specification (see references). Set "CLICLEVELS" to zero to indicate the CLIC is not implemented.

The model can configured either to use an internal CLIC model (if parameter "externalCLIC" is False) or to present a net interface to allow the CLIC to be implemented externally in a platform component (if parameter "externalCLIC" is True). When the CLIC is implemented internally, net ports for standard interrupts and additional local interrupts are available. When the CLIC is implemented externally, a net port interface allowing the highest-priority pending interrupt to be delivered is instead present. This is described below.

### 1.10.1 CLIC Common Parameters

This section describes parameters applicable whether the CLIC is implemented internally or externally. These are:

"CLICANDBASIC": this Boolean parameter indicates whether both CLIC and basic interrupt controller are present (if True) or whether only the CLIC is present (if False). The default value in this variant is True.

"CLICXNXTI": this Boolean parameter indicates whether xnxti CSRs are implemented (if True) or unimplemented (if False). The default value in this variant is True.

"CLICXCSW": this Boolean parameter indicates whether xscratchcsw and xscratchcswl CSRs registers are implemented (if True) or unimplemented (if False). The default value in this variant is False.

"mclicbase": this parameter specifies the CLIC base address in physical memory. The default value in this variant is 0x2800000.

"tvt_undefined": this Boolean parameter indicates whether xtvt CSRs registers are implemented (if True) or unimplemented (if False). If the registers are unimplemented then the model will use

basic mode vectored interrupt semantics based on the xtvec CSRs instead of Selective Hardware Vectoring semantics described in the specification. The default value in this variant is False.

"intthresh_undefined": this Boolean parameter indicates whether xintthresh CSRs registers are implemented (if True) or unimplemented (if False). The default value in this variant is True.

"mclicbase_undefined": this Boolean parameter indicates whether the mclicbase CSR register is implemented (if True) or unimplemented (if False). The default value in this variant is True.

### 1.10.2 CLIC Internal-Implementation Parameters

This section describes parameters applicable only when the CLIC is implemented internally. These are:

"CLIC_version": this defines the version of the CLIC specification that is implemented. See the References section of this document for more information.

"CLICCFGMBITS": this Uns32 parameter indicates the number of bits implemented in cliccfg.nmbits, and also indirectly defines CLICPRIVMODES. For cores which implement only Machine mode, or which implement Machine and User modes but not the N extension, the parameter is absent ("CLICCFGMBITS" must be zero in these cases). The default value in this variant is 0.

"CLICCFGLBITS": this Uns32 parameter indicates the number of bits implemented in cliccfg.nlbits. The default value in this variant is 2.

"CLICSELHVEC": this Boolean parameter indicates whether Selective Hardware Vectoring is supported (if True) or unsupported (if False). The default value in this variant is True.

"posedge_0_63": this Uns64 parameter is a mask for interrupts 0 to 63 indicating whether an interrupt is fixed positive edge triggered. If the corresponding mask bit for interrupt N is 1 then the trig value for that interrupt is hard wired to 1. If the mask bit is zero, then the interrupt is either fixed level triggered (see below) or may be configured as either edge or level triggered. The default value in this variant is 0x100f.

"poslevel_0_63": this Uns64 parameter is a mask for interrupts 0 to 63 indicating whether an interrupt is fixed positive level triggered. If the corresponding mask bit for interrupt N is 1 then the trig value for that interrupt is hard wired to 0. If the mask bit is zero, then the interrupt may be configured as either edge or level triggered. The default value in this variant is 0x0.

"posedge_other": this Boolean parameter indicates whether interrupts 64 and above are fixed positive edge triggered. If True then the trig value for those interrupts is hard wired to 1. If False, then those interrupts are either fixed level triggered (see below) or may be configured as either edge or level triggered. The default value in this variant is False.

"poslevel_other": this Boolean parameter indicates whether interrupts 64 and above are fixed positive level triggered. If True then the trig value for those interrupts is hard wired to 0. If False, then those interrupts may be configured as either edge or level triggered. The default value in this variant is False.

### 1.10.3 CLIC External-Implementation Net Port Interface

When the CLIC is externally implemented, net ports are present allowing the external CLIC model to supply the highest-priority pending interrupt and to be notified when interrupts are handled. These are:

"irq_id_i": this input should be written with the id of the highest-priority pending interrupt.

"irq_lev_i": this input should be written with the highest-priority interrupt level.

"irq_sec_i": this 2-bit input should be written with the highest-priority interrupt security state (00:User, 01:Supervisor, 11:Machine).

"irq_shv_i": this input port should be written to indicate whether the highest-priority interrupt should be direct (0) or vectored (1). If the "tvt_undefined parameter" is False, vectored interrupts will use selective hardware vectoring, as described in the CLIC specification. If "tvt_undefined" is True, vectored interrupts will behave like basic mode vectored interrupts.

"irq_i": this input should be written with 1 to indicate that the external CLIC is presenting an interrupt, or 0 if no interrupt is being presented.

"irq_ack_o": this output is written by the model on entry to the interrupt handler (i.e. when the interrupt is taken). It will be written as an instantaneous pulse (i.e. written to 1, then immediately 0).

"irq_id_o": this output is written by the model with the id of the interrupt currently being handled. It is valid during the instantaneous irq_ack_o pulse.

"sec_lvl_o": this output signal indicates the current secure status of the processor, as a 2-bit value (00=User, 01:Supervisor, 11=Machine).

### 1.10.4 CLIC Versions

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

### 1.10.5 Version 20180831

Legacy version of August 31 2018, required for SiFive cores.

### 1.10.6 Version 0.9-draft-20191208

Stable 0.9 version of December 8 2019.

### 1.10.7 Version master

Unstable master version as of 11 May 2021 (commit dd15cd3), with these changes compared to version 0.9-draft-20191208:

- level-sensitive interrupts may no longer be set pending by software;

- if there is an access exception on table load when Selective Hardware Vectoring is enabled, both xtval and xepc hold the faulting address;

- if the xinhv bit is set, the target address for an xret instruction is obtained by a load from address xepc.

## 1.11 CLINT

This model implements a SiFive-compatible Core Local Interruptor (CLINT) block at address 0x2000000. Use parameter "CLINT_address" to specify a different address for this block, or set it to zero to disable the CLINT

The tick timer in the CLINT (mtime) will increment at 32768Hz. Use parameter "mtime_Hz" to specify a different timer frequency.

## 1.12 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 applied the value on the ID port is sampled and used to fill the Exception Code field in the "mcause" CSR (or the equivalent CSR for other execution levels). For Machine mode, the extra interrupt ID port is called "MExternalInterruptID".

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.

## 1.13 Debug Mode

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 three 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.

### 1.13.1 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), dcsr cause will be reported as trigger;

2. By writing a 1 then 0 to net "haltreq" (using opNetWrite) followed by simulation of at least one cycle (e.g. using opProcessorSimulate);

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);

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 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.

### 1.13.2 Debug State Exit

Exit from Debug mode can be performed in any 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.

### 1.13.3 Debug Registers

When Debug mode is enabled, registers "dcsr", "dpc", "dscratch0" and "dscratch1" are implemented as described in the specification. These 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.

### 1.13.4 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.

### 1.13.5 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.

### 1.13.6 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).

## 1.14 Debug Mask

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

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

Value 0x004: enable debugging of interrupt state.

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

## 1.15 Integration Support

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

### 1.15.1 CSR Register External Implementation

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

## 1.16 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.

## 1.17 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.

## 1.18 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 Ratified-IMFDQC-and-Priv-v1.11)

RISC-V “B” Bit Manipulation Extension (Bit Manipulation Architecture Version v0.94-20210120)

RISC-V Core-Local Interrupt Controller (CLIC Version 20180831)

SiFive E20 Manual v19.05

# Chapter 2

# Configuration

## 2.1 Location

This model's VLNV is sifive.ovpworld.org/processor/riscv/1.0.
The model source is usually at:
$IMPERAS_HOME/ImperasLib/source/sifive.ovpworld.org/processor/riscv/1.0
The model binary is usually at:
$IMPERAS_HOME/lib/$IMPERAS_ARCH/ImperasLib/sifive.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 |
|---|---|
| E20 | (described in this document) |
| E21 | |
| E24 | |
| E31 | |
| E34 | |
| E51 | |
| E76 | |
| S21 | |
| S51 | |
| S54 | |
| S76 | |
| U54 | |
| U74 | |
| X280 | |
| P550 | |

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 |
| MSWInterrupt | input | optional | Machine software interrupt |
| MTimerInterrupt | input | optional | Machine timer interrupt |
| MExternalInterrupt | input | optional | Machine external interrupt |
| CSIP | input | optional | CLIC software interrupt |
| LocalInterrupt0 | input | optional | Local interrupt 0 |
| LocalInterrupt1 | input | optional | Local interrupt 1 |
| LocalInterrupt2 | input | optional | Local interrupt 2 |
| LocalInterrupt3 | input | optional | Local interrupt 3 |
| LocalInterrupt4 | input | optional | Local interrupt 4 |
| LocalInterrupt5 | input | optional | Local interrupt 5 |
| LocalInterrupt6 | input | optional | Local interrupt 6 |
| LocalInterrupt7 | input | optional | Local interrupt 7 |
| LocalInterrupt8 | input | optional | Local interrupt 8 |
| LocalInterrupt9 | input | optional | Local interrupt 9 |
| LocalInterrupt10 | input | optional | Local interrupt 10 |
| LocalInterrupt11 | input | optional | Local interrupt 11 |
| LocalInterrupt12 | input | optional | Local interrupt 12 |
| LocalInterrupt13 | input | optional | Local interrupt 13 |
| LocalInterrupt14 | input | optional | Local interrupt 14 |
| LocalInterrupt15 | input | optional | Local interrupt 15 |
| LocalInterrupt16 | input | optional | Local interrupt 16 |
| LocalInterrupt17 | input | optional | Local interrupt 17 |
| LocalInterrupt18 | input | optional | Local interrupt 18 |
| LocalInterrupt19 | input | optional | Local interrupt 19 |
| LocalInterrupt20 | input | optional | Local interrupt 20 |
| LocalInterrupt21 | input | optional | Local interrupt 21 |

| | | | |
|---|---|---|---|
| LocalInterrupt22 | input | optional | Local interrupt 22 |
| LocalInterrupt23 | input | optional | Local interrupt 23 |
| LocalInterrupt24 | input | optional | Local interrupt 24 |
| LocalInterrupt25 | input | optional | Local interrupt 25 |
| LocalInterrupt26 | input | optional | Local interrupt 26 |
| LocalInterrupt27 | input | optional | Local interrupt 27 |
| LocalInterrupt28 | input | optional | Local interrupt 28 |
| LocalInterrupt29 | input | optional | Local interrupt 29 |
| LocalInterrupt30 | input | optional | Local interrupt 30 |
| LocalInterrupt31 | input | optional | Local interrupt 31 |
| 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 |
| deferint | input | optional | Artifact signal causing interrupts to be held off 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** | | |
| variant | Enumeration | Selects variant (either a generic UISA or a specific model) |
| user_version | Enumeration | Specify required User Architecture version (2.2, 2.3, 20190305 or 20191213) |
| priv_version | Enumeration | Specify required Privileged Architecture version (1.10, 1.11, 20190405, 20190608 or master) |
| endian | Endian | Model endian |
| 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 |
| Zicsr | Boolean | Specify that Zicsr 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 |
| Zmmul | Boolean | Specify that Zmmul is implemented |
| **Bit_Manipulation** | | |
| bitmanip_version | Enumeration | Specify required Bit Manipulation Architecture version (0.90, 0.91, 0.92, 0.93-draft, 0.93, 0.94, 1.0.0 or master) |
| Zba | Boolean | Specify that Zba is implemented (bit manipulation extension) |
| Zbb | Boolean | Specify that Zbb is implemented (bit manipulation extension) |
| Zbc | Boolean | Specify that Zbc is implemented (bit manipulation extension) |
| Zbe | Boolean | Specify that Zbe is implemented (bit manipulation extension; ignored if version 1.0.0) |

| | | |
|---|---|---|
| Zbf | Boolean | Specify that Zbf is implemented (bit manipulation extension; ignored if version 1.0.0) |
| Zbm | Boolean | Specify that Zbm is implemented (bit manipulation extension; ignored if version 1.0.0) |
| Zbp | Boolean | Specify that Zbp is implemented (bit manipulation extension; ignored if version 1.0.0) |
| Zbr | Boolean | Specify that Zbr is implemented (bit manipulation extension; ignored if version 1.0.0) |
| Zbs | Boolean | Specify that Zbs is implemented (bit manipulation extension) |
| Zbt | Boolean | Specify that Zbt is implemented (bit manipulation extension; ignored if version 1.0.0) |
| **Interrupts_Exceptions** | | |
| rnmi_version | Enumeration | Specify required RNMI Architecture version (none or 0.2.1) |
| 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 |
| 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 |
| 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) |
| mtime_Hz | Double | Specify clock frequency of CLINT mtime counter |
| local_int_num | Uns32 | Specify number of supplemental local interrupts |
| 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 |
| no_ideleg | Uns64 | Specify mask of interrupts that cannot be delegated to lower-priority execution levels |
| no_edeleg | 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 |
| **Fast_Interrupt** | | |
| CLIC_version | Enumeration | Specify required CLIC version (20180831, 0.9-draft-20191208 or master) |
| mclicbase | Uns64 | Override mclicbase register |
| CLICLEVELS | Uns32 | Specify number of interrupt levels implemented by CLIC, or 0 if CLIC absent |
| CLICANDBASIC | Boolean | Whether original basic mode is also implemented |
| CLICVERSION | Uns32 | Specify CLIC version |
| CLICINTCTLBITS | Uns32 | Specify number of bits implemented in clicintctl[i] |
| CLICCFGLBITS | Uns32 | Specify number of bits implemented for cliccfg.nlbits |
| CLICSELHVEC | Boolean | Whether selective hardware vectoring supported |
| CLICXNXTI | Boolean | Whether xnxti CSRs implemented |
| CLICXCSW | Boolean | Whether xscratchcsw/xscratchcswl CSRs implemented |
| externalCLIC | Boolean | Whether CLIC is implemented externally (if False, then use implementation in this model) |
| tvt_undefined | Boolean | Specify that mtvt, stvt and utvt CSRs are undefined |
| intthresh_undefined | Boolean | Specify that mintthreash, sintthresh and uintthresh CSRs are undefined |
| mclicbase_undefined | Boolean | Specify that mclicbase CSR is undefined |
| posedge_0_63 | Uns64 | Mask of interrupts 0 to 63 that are fixed positive edge triggered |
| poslevel_0_63 | Uns64 | Mask of interrupts 0 to 63 that are fixed positive level triggered |
| posedge_other | Boolean | Whether interrupts 64 and above are fixed positive edge triggered |

| | | |
|---|---|---|
| poslevel_other | Boolean | Whether interrupts 64 and above are fixed positive level triggered |
| **Debug** | | |
| debug_mode | Enumeration | Specify how Debug mode is implemented (none, vector, interrupt or halt) |
| **Simulation_Artifact** | | |
| use_hw_reg_names | Boolean | Specify whether to use hardware register names x0-x31 and f0-f31 instead of ABI register names |
| verbose | Boolean | Specify verbose output messages |
| traceVolatile | Boolean | Specify whether volatile registers (e.g. minstret) should be shown in change trace |
| 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 <csrName>=<number> |
| **Memory** | | |
| unaligned | Boolean | Specify whether the processor supports unaligned memory accesses |
| PMP_grain | Uns32 | Specify PMP region granularity, G (0 =>4 bytes, 1 =>8 bytes, etc) |
| PMP_registers | Uns32 | Specify the number of implemented PMP address registers |
| 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 |
| **Instruction_CSR_Behavior** | | |
| wfi_is_nop | Boolean | Specify whether WFI should be treated as a NOP (if not, halt while waiting for interrupts) |
| counteren_mask | Uns32 | Specify hardware-enforced mask of writable bits in mcounteren/scounteren registers |
| noinhibit_mask | Uns32 | Specify hardware-enforced mask of always-zero bits in mcountinhibit register |
| cycle_undefined | Boolean | Specify that the cycle CSR is undefined |
| time_undefined | Boolean | Specify that the time CSR is undefined |
| instret_undefined | Boolean | Specify that the instret CSR is undefined |
| hpmcounter_undefined | Boolean | Specify that the hpmcounter CSRs are undefined |
| **CSR_Masks** | | |
| mtvec_mask | Uns64 | Specify hardware-enforced mask of writable bits in mtvec register |
| mtvt_mask | Uns64 | Specify hardware-enforced mask of writable bits in CLIC mtvt register |
| mip_mask | Uns64 | Specify hardware-enforced mask of writable bits in mip register |
| mtvec_sext | Boolean | Specify whether mtvec is sign-extended from most-significant bit |
| mtvt_sext | Boolean | Specify whether mtvt is sign-extended from most-significant bit |
| **Trigger** | | |
| trigger_num | Uns32 | Specify the number of implemented hardware triggers |
| **CSR_Defauts** | | |
| 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) |
| mtvec | Uns64 | Override mtvec register |
| **Compressed** | | |
| Zcea_version | Enumeration | Specify version of Zcea implemented (code-size reduction extension) (none or 0.50.1) |
| Zceb_version | Enumeration | Specify version of Zceb implemented (code-size reduction extension) (none or 0.50.1) |
| Zcee_version | Enumeration | Specify version of Zcee implemented (code-size reduction extension) (none or 1.0.0-rc) |

Table 8.1: Parameters that can be set in: Hart

## 8.1 Extension Parameters

| Name | Type | Description |
|---|---|---|
| FeatureDisable_Present | Boolean | Specify whether Feature Disable CSR present |
| BranchPredictionMode_Present | Boolean | Specify whether Branch Prediction Mode CSR present |
| PowerDial_Present | Boolean | Specify whether PowerDial CSR present |
| CFLUSH_Present | Boolean | Specify whether CFLUSH.D.L1 and CDISCARD.D.L1 instructions present |
| CEASE_Present | Boolean | Specify whether CEASE instruction present |

Table 8.2: Parameters for sifiveExtensions

## 8.2 Parameters with enumerated types

### 8.2.1 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 8.3: Values for Parameter user_version

### 8.2.2 Parameter priv_version

| Set to this value | Description |
|---|---|
| 1.10 | Privileged Architecture Version 1.10 |
| 1.11 | Deprecated and equivalent to 20190608 |
| 20190405 | Deprecated and equivalent to 20190608 |
| 20190608 | Privileged Architecture Version Ratified-IMFDQC-and-Priv-v1.11 |
| master | Privileged Architecture Master Branch (1.12 draft) |

Table 8.4: Values for Parameter priv_version

### 8.2.3 Parameter bitmanip_version

| Set to this value | Description |
|---|---|
| 0.90 | Bit Manipulation Architecture Version v0.90-20190610 |
| 0.91 | Bit Manipulation Architecture Version v0.91-20190829 |
| 0.92 | Bit Manipulation Architecture Version v0.92-20191108 |
| 0.93-draft | Bit Manipulation Architecture Version 0.93-draft-20200129 |
| 0.93 | Bit Manipulation Architecture Version v0.93-20210110 |
| 0.94 | Bit Manipulation Architecture Version v0.94-20210120 |
| 1.0.0 | Bit Manipulation Architecture Version 1.0.0 |
| master | Bit Manipulation Master Branch as of commit 1f56afe (this is subject to change) |

Table 8.5: Values for Parameter bitmanip_version

### 8.2.4 Parameter rnmi_version

| Set to this value | Description |
|---|---|

| | |
|---|---|
| none | RNMI not implemented |
| 0.2.1 | RNMI version 0.2.1 |

Table 8.6: Values for Parameter rnmi_version

### 8.2.5 Parameter CLIC_version

| **Set to this value** | Description |
|---|---|
| 20180831 | CLIC Version 20180831 |
| 0.9-draft-20191208 | CLIC Version 0.9-draft-20191208 |
| master | CLIC Master Branch as of commit dd15cd3 (this is subject to change) |

Table 8.7: Values for Parameter CLIC_version

### 8.2.6 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 |

Table 8.8: Values for Parameter debug_mode

### 8.2.7 Parameter Zcea_version

| **Set to this value** | Description |
|---|---|
| none | Zcea not implemented |
| 0.50.1 | Zcea version 0.50.1 |

Table 8.9: Values for Parameter Zcea_version

### 8.2.8 Parameter Zceb_version

| **Set to this value** | Description |
|---|---|
| none | Zceb not implemented |
| 0.50.1 | Zceb version 0.50.1 |

Table 8.10: Values for Parameter Zceb_version

### 8.2.9 Parameter Zcee_version

| **Set to this value** | Description |
|---|---|
| none | Zcee not implemented |
| 1.0.0-rc | Zcee version 1.0.0-rc |

Table 8.11: Values for Parameter Zcee_version

## 8.3 Parameter values

These are the current parameter values.

| **Name** | Value |
|---|---|
| **Fundamental** | |
| variant | E20 |
| user_version | 2.3 |
| priv_version | 1.11 |
| endian | none |
| enable_expanded | F |
| endianFixed | F |
| misa_MXL | 1 |
| misa_Extensions | 0x1106 |
| add_Extensions | |
| sub_Extensions | |
| misa_Extensions_mask | 0 |
| add_Extensions_mask | |
| sub_Extensions_mask | |
| add_implicit_Extensions | |
| sub_implicit_Extensions | |
| Zicsr | T |
| Zifencei | T |
| Zicbom | F |
| Zicbop | F |
| Zicboz | F |
| Zmmul | F |
| **Bit_Manipulation** | |
| bitmanip_version | 0.94 |
| Zba | T |
| Zbb | T |
| Zbc | F |
| Zbe | F |
| Zbf | F |
| Zbm | F |
| Zbp | F |
| Zbr | F |
| Zbs | F |
| Zbt | F |
| **Interrupts_Exceptions** | |
| rnmi_version | none |
| mtvec_is_ro | F |
| tvec_align | 64 |
| ecode_mask | 31 |
| ecode_nmi | 2 |
| tval_zero | F |
| tval_zero_ebreak | F |
| tval_ii_code | T |
| reset_address | 0 |

| | |
|---|---|
| nmi_address | 0 |
| CLINT_address | 0x2000000 |
| mtime_Hz | 3.276800e+04 |
| local_int_num | 32 |
| unimp_int_mask | 0 |
| force_mideleg | 0 |
| no_ideleg | 0 |
| no_edeleg | 0 |
| external_int_id | F |
| **Fast_Interrupt** | |
| CLIC_version | 20180831 |
| mclicbase | 0x2800000 |
| CLICLEVELS | 16 |
| CLICANDBASIC | T |
| CLICVERSION | 17 |
| CLICINTCTLBITS | 5 |
| CLICCFGLBITS | 2 |
| CLICSELHVEC | T |
| CLICXNXTI | T |
| CLICXCSW | F |
| externalCLIC | F |
| tvt_undefined | F |
| intthresh_undefined | T |
| mclicbase_undefined | T |
| posedge_0_63 | 0x100f |
| poslevel_0_63 | 0 |
| posedge_other | F |
| poslevel_other | F |
| **Debug** | |
| debug_mode | none |
| **Simulation_Artifact** | |
| use_hw_reg_names | F |
| verbose | F |
| traceVolatile | F |
| enable_CSR_bus | F |
| CSR_remap | |
| **Memory** | |
| unaligned | F |
| PMP_grain | 0 |
| PMP_registers | 0 |
| PMP_max_page | 0 |
| PMP_decompose | F |
| **Instruction_CSR_Behavior** | |
| wfi_is_nop | F |
| counteren_mask | 0xffffffff |

| | |
|---|---|
| noinhibit_mask | 0 |
| cycle_undefined | F |
| time_undefined | T |
| instret_undefined | F |
| hpmcounter_undefined | F |
| **CSR_Masks** | |
| mtvec_mask | 0 |
| mtvt_mask | 0 |
| mip_mask | 0x337 |
| mtvec_sext | F |
| mtvt_sext | F |
| **Trigger** | |
| trigger_num | 0 |
| **CSR_Defauts** | |
| mvendorid | 0x489 |
| marchid | 0x80000002 |
| mimpid | 0x4210427 |
| mhartid | 0 |
| mtvec | 0 |
| **Compressed** | |
| Zcea_version | none |
| Zceb_version | none |
| Zcee_version | none |
| **sifiveExtensions** | |
| FeatureDisable_Present* | F |
| BranchPredictionMode_Present* | F |
| PowerDial_Present* | F |
| CFLUSH_Present* | F |
| CEASE_Present* | F |

Table 8.12: Parameter values

* Parameters marked with an asterisk are part of the processor extension library.

# Chapter 9

# Execution Modes

| **Mode** | Code | Description |
|---|---|---|
| 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 |
| 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 |
| MSWInterrupt | 67 | Machine software interrupt |
| MTimerInterrupt | 71 | Machine timer interrupt |
| MExternalInterrupt | 75 | Machine external interrupt |
| CSIP | 76 | CLIC software interrupt |
| LocalInterrupt0 | 80 | Local interrupt 0 |
| LocalInterrupt1 | 81 | Local interrupt 1 |
| LocalInterrupt2 | 82 | Local interrupt 2 |
| LocalInterrupt3 | 83 | Local interrupt 3 |
| LocalInterrupt4 | 84 | Local interrupt 4 |
| LocalInterrupt5 | 85 | Local interrupt 5 |
| LocalInterrupt6 | 86 | Local interrupt 6 |
| LocalInterrupt7 | 87 | Local interrupt 7 |
| LocalInterrupt8 | 88 | Local interrupt 8 |
| LocalInterrupt9 | 89 | Local interrupt 9 |
| LocalInterrupt10 | 90 | Local interrupt 10 |
| LocalInterrupt11 | 91 | Local interrupt 11 |

| | | |
|---|---|---|
| LocalInterrupt12 | 92 | Local interrupt 12 |
| LocalInterrupt13 | 93 | Local interrupt 13 |
| LocalInterrupt14 | 94 | Local interrupt 14 |
| LocalInterrupt15 | 95 | Local interrupt 15 |
| LocalInterrupt16 | 96 | Local interrupt 16 |
| LocalInterrupt17 | 97 | Local interrupt 17 |
| LocalInterrupt18 | 98 | Local interrupt 18 |
| LocalInterrupt19 | 99 | Local interrupt 19 |
| LocalInterrupt20 | 100 | Local interrupt 20 |
| LocalInterrupt21 | 101 | Local interrupt 21 |
| LocalInterrupt22 | 102 | Local interrupt 22 |
| LocalInterrupt23 | 103 | Local interrupt 23 |
| LocalInterrupt24 | 104 | Local interrupt 24 |
| LocalInterrupt25 | 105 | Local interrupt 25 |
| LocalInterrupt26 | 106 | Local interrupt 26 |
| LocalInterrupt27 | 107 | Local interrupt 27 |
| LocalInterrupt28 | 108 | Local interrupt 28 |
| LocalInterrupt29 | 109 | Local interrupt 29 |
| LocalInterrupt30 | 110 | Local interrupt 30 |
| LocalInterrupt31 | 111 | Local interrupt 31 |

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 4 commands.
This level in the model hierarchy has 4 register groups:

| **Group name** | Registers |
|---|---|
| Core | 33 |
| Machine_Control_and_Status | 190 |
| CLINT | 3 |
| Integration_support | 1 |

Table 11.1: Register groups

This level in the model hierarchy has no children.

# Chapter 12

# Model Commands

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

## 12.1 Level 1: Hart

### 12.1.1 getCSRIndex

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

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

Table 12.1: getCSRIndex command arguments

### 12.1.2 isync

specify instruction address range for synchronous execution

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

Table 12.2: isync command arguments

### 12.1.3 itrace

enable or disable instruction tracing

| **Argument** | Type | Description |
|---|---|---|
| -after | Uns64 | apply after this many instructions |
| -enable | Boolean | enable instruction tracing |
| -instructioncount | Boolean | include the instruction number in each trace |
| -memory | String | show memory accesses by this instruction. Argument can be any combination of X (execute), L (load or store access) and S (system) |
| -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 12.3: itrace command arguments

### 12.1.4 listCSRs

#### 12.1.4.1 Argument description

List all CSRs in index order

# 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 Machine_Control_and_Status

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

| **Name** | Bits | Initial-Hex | RW | Description |
|---|---|---|---|---|
| mstatus | 32 | 1800 | rw | Machine Status |
| misa | 32 | 40001106 | rw | ISA and Extensions |
| mie | 32 | 0 | rw | Machine Interrupt Enable |
| mtvec | 32 | 0 | rw | Machine Trap-Vector Base-Address |
| mtvt | 32 | 0 | rw | Machine CLIC Trap-Vector Base-Address |
| mcountinhibit | 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 | 80 | rw | Machine Interrupt Pending |
| mnxti | 32 | 0 | rw | Machine Interrupt Handler Address/Enable |
| mintstatus | 32 | 0 | rw | Machine Interrupt Status |
| 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 |
| 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 |
| mcycle | 32 | 0 | rw | Machine Cycle Counter |
| minstret | 32 | 0 | rw | Machine Instructions Retired |
| mhpmcounter3 | 32 | 0 | rw | Machine Performance Monitor Counter 3 |
| mhpmcounter4 | 32 | 0 | rw | Machine Performance Monitor Counter 4 |
| mhpmcounter5 | 32 | 0 | rw | Machine Performance Monitor Counter 5 |
| mhpmcounter6 | 32 | 0 | rw | Machine Performance Monitor Counter 6 |
| mhpmcounter7 | 32 | 0 | rw | Machine Performance Monitor Counter 7 |
| mhpmcounter8 | 32 | 0 | rw | Machine Performance Monitor Counter 8 |
| mhpmcounter9 | 32 | 0 | rw | Machine Performance Monitor Counter 9 |
| mhpmcounter10 | 32 | 0 | rw | Machine Performance Monitor Counter 10 |
| mhpmcounter11 | 32 | 0 | rw | Machine Performance Monitor Counter 11 |
| mhpmcounter12 | 32 | 0 | rw | Machine Performance Monitor Counter 12 |
| mhpmcounter13 | 32 | 0 | rw | Machine Performance Monitor Counter 13 |
| mhpmcounter14 | 32 | 0 | rw | Machine Performance Monitor Counter 14 |
| mhpmcounter15 | 32 | 0 | rw | Machine Performance Monitor Counter 15 |
| mhpmcounter16 | 32 | 0 | rw | Machine Performance Monitor Counter 16 |
| mhpmcounter17 | 32 | 0 | rw | Machine Performance Monitor Counter 17 |
| mhpmcounter18 | 32 | 0 | rw | Machine Performance Monitor Counter 18 |
| mhpmcounter19 | 32 | 0 | rw | Machine Performance Monitor Counter 19 |
| mhpmcounter20 | 32 | 0 | rw | Machine Performance Monitor Counter 20 |
| mhpmcounter21 | 32 | 0 | rw | Machine Performance Monitor Counter 21 |
| mhpmcounter22 | 32 | 0 | rw | Machine Performance Monitor Counter 22 |
| mhpmcounter23 | 32 | 0 | rw | Machine Performance Monitor Counter 23 |
| mhpmcounter24 | 32 | 0 | rw | Machine Performance Monitor Counter 24 |
| mhpmcounter25 | 32 | 0 | rw | Machine Performance Monitor Counter 25 |
| mhpmcounter26 | 32 | 0 | rw | Machine Performance Monitor Counter 26 |
| mhpmcounter27 | 32 | 0 | rw | Machine Performance Monitor Counter 27 |
| mhpmcounter28 | 32 | 0 | rw | Machine Performance Monitor Counter 28 |
| mhpmcounter29 | 32 | 0 | rw | Machine Performance Monitor Counter 29 |
| mhpmcounter30 | 32 | 0 | rw | Machine Performance Monitor Counter 30 |
| mhpmcounter31 | 32 | 0 | rw | Machine Performance Monitor Counter 31 |
| mcycleh | 32 | 0 | rw | Machine Cycle Counter High |
| minstreth | 32 | 0 | rw | Machine Instructions Retired High |
| mhpmcounterh3 | 32 | 0 | rw | Machine Performance Monitor Counter High 3 |
| mhpmcounterh4 | 32 | 0 | rw | Machine Performance Monitor Counter High 4 |
| mhpmcounterh5 | 32 | 0 | rw | Machine Performance Monitor Counter High 5 |
| mhpmcounterh6 | 32 | 0 | rw | Machine Performance Monitor Counter High 6 |
| mhpmcounterh7 | 32 | 0 | rw | Machine Performance Monitor Counter High 7 |
| mhpmcounterh8 | 32 | 0 | rw | Machine Performance Monitor Counter High 8 |
| mhpmcounterh9 | 32 | 0 | rw | Machine Performance Monitor Counter High 9 |
| mhpmcounterh10 | 32 | 0 | rw | Machine Performance Monitor Counter High 10 |
| mhpmcounterh11 | 32 | 0 | rw | Machine Performance Monitor Counter High 11 |

| | | | | |
|---|---|---|---|---|
| mhpmcounterh12 | 32 | 0 | rw | Machine Performance Monitor Counter High 12 |
| mhpmcounterh13 | 32 | 0 | rw | Machine Performance Monitor Counter High 13 |
| mhpmcounterh14 | 32 | 0 | rw | Machine Performance Monitor Counter High 14 |
| mhpmcounterh15 | 32 | 0 | rw | Machine Performance Monitor Counter High 15 |
| mhpmcounterh16 | 32 | 0 | rw | Machine Performance Monitor Counter High 16 |
| mhpmcounterh17 | 32 | 0 | rw | Machine Performance Monitor Counter High 17 |
| mhpmcounterh18 | 32 | 0 | rw | Machine Performance Monitor Counter High 18 |
| mhpmcounterh19 | 32 | 0 | rw | Machine Performance Monitor Counter High 19 |
| mhpmcounterh20 | 32 | 0 | rw | Machine Performance Monitor Counter High 20 |
| mhpmcounterh21 | 32 | 0 | rw | Machine Performance Monitor Counter High 21 |
| mhpmcounterh22 | 32 | 0 | rw | Machine Performance Monitor Counter High 22 |
| mhpmcounterh23 | 32 | 0 | rw | Machine Performance Monitor Counter High 23 |
| mhpmcounterh24 | 32 | 0 | rw | Machine Performance Monitor Counter High 24 |
| mhpmcounterh25 | 32 | 0 | rw | Machine Performance Monitor Counter High 25 |
| mhpmcounterh26 | 32 | 0 | rw | Machine Performance Monitor Counter High 26 |
| mhpmcounterh27 | 32 | 0 | rw | Machine Performance Monitor Counter High 27 |
| mhpmcounterh28 | 32 | 0 | rw | Machine Performance Monitor Counter High 28 |
| mhpmcounterh29 | 32 | 0 | rw | Machine Performance Monitor Counter High 29 |
| mhpmcounterh30 | 32 | 0 | rw | Machine Performance Monitor Counter High 30 |
| mhpmcounterh31 | 32 | 0 | rw | Machine Performance Monitor Counter High 31 |
| cycle | 32 | 0 | r- | Cycle Counter |
| 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 |
| instreth | 32 | 0 | r- | Instructions Retired High |
| hpmcounterh3 | 32 | 0 | r- | Performance Monitor High 3 |
| hpmcounterh4 | 32 | 0 | r- | Performance Monitor High 4 |
| hpmcounterh5 | 32 | 0 | r- | Performance Monitor High 5 |

| | | | | |
|---|---|---|---|---|
| hpmcounterh6 | 32 | 0 | r- | Performance Monitor High 6 |
| hpmcounterh7 | 32 | 0 | r- | Performance Monitor High 7 |
| hpmcounterh8 | 32 | 0 | r- | Performance Monitor High 8 |
| hpmcounterh9 | 32 | 0 | r- | Performance Monitor High 9 |
| hpmcounterh10 | 32 | 0 | r- | Performance Monitor High 10 |
| hpmcounterh11 | 32 | 0 | r- | Performance Monitor High 11 |
| hpmcounterh12 | 32 | 0 | r- | Performance Monitor High 12 |
| hpmcounterh13 | 32 | 0 | r- | Performance Monitor High 13 |
| hpmcounterh14 | 32 | 0 | r- | Performance Monitor High 14 |
| hpmcounterh15 | 32 | 0 | r- | Performance Monitor High 15 |
| hpmcounterh16 | 32 | 0 | r- | Performance Monitor High 16 |
| hpmcounterh17 | 32 | 0 | r- | Performance Monitor High 17 |
| hpmcounterh18 | 32 | 0 | r- | Performance Monitor High 18 |
| hpmcounterh19 | 32 | 0 | r- | Performance Monitor High 19 |
| hpmcounterh20 | 32 | 0 | r- | Performance Monitor High 20 |
| hpmcounterh21 | 32 | 0 | r- | Performance Monitor High 21 |
| hpmcounterh22 | 32 | 0 | r- | Performance Monitor High 22 |
| hpmcounterh23 | 32 | 0 | r- | Performance Monitor High 23 |
| hpmcounterh24 | 32 | 0 | r- | Performance Monitor High 24 |
| hpmcounterh25 | 32 | 0 | r- | Performance Monitor High 25 |
| hpmcounterh26 | 32 | 0 | r- | Performance Monitor High 26 |
| hpmcounterh27 | 32 | 0 | r- | Performance Monitor High 27 |
| hpmcounterh28 | 32 | 0 | r- | Performance Monitor High 28 |
| hpmcounterh29 | 32 | 0 | r- | Performance Monitor High 29 |
| hpmcounterh30 | 32 | 0 | r- | Performance Monitor High 30 |
| hpmcounterh31 | 32 | 0 | r- | Performance Monitor High 31 |
| mvendorid | 32 | 489 | r- | Vendor ID |
| marchid | 32 | 80000002 | r- | Architecture ID |
| mimpid | 32 | 4210427 | r- | Implementation ID |
| mhartid | 32 | 0 | r- | Hardware Thread ID |

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

### 13.1.3 CLINT

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

| **Name** | Bits | Initial-Hex | RW | Description |
|---|---|---|---|---|
| msip | 8 | 0 | rw | CLINT msip |
| mtimecmp | 64 | 0 | rw | CLINT mtimecmp |
| mtime | 64 | 0 | rw | CLINT mtime |

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

### 13.1.4 Integration_support

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

| **Name** | Bits | Initial-Hex | RW | Description |
|---|---|---|---|---|
| commercial | 8 | 0 | r- | Commercial feature in use |

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