RISC-V Configuration Validator Documentation

Release beta

InCore Semiconductors Pvt. Ltd.

Aug 03, 2019

CONTENTS

CHAPTER

ONE

INTRODUCTION

RISCV-Config (RISCV Configuration Leagalizer) is a YAML based framework which can be used to validate the specifications of a RISC-V implementation against the RISC-V privileged and unprivileged ISA spec and generate standard specification yaml file.

Caution: This is still a work in progress and non-backward compatible changes are expected to happen.

For more information on the official RISC-V spec please visit: RISC-V Specs

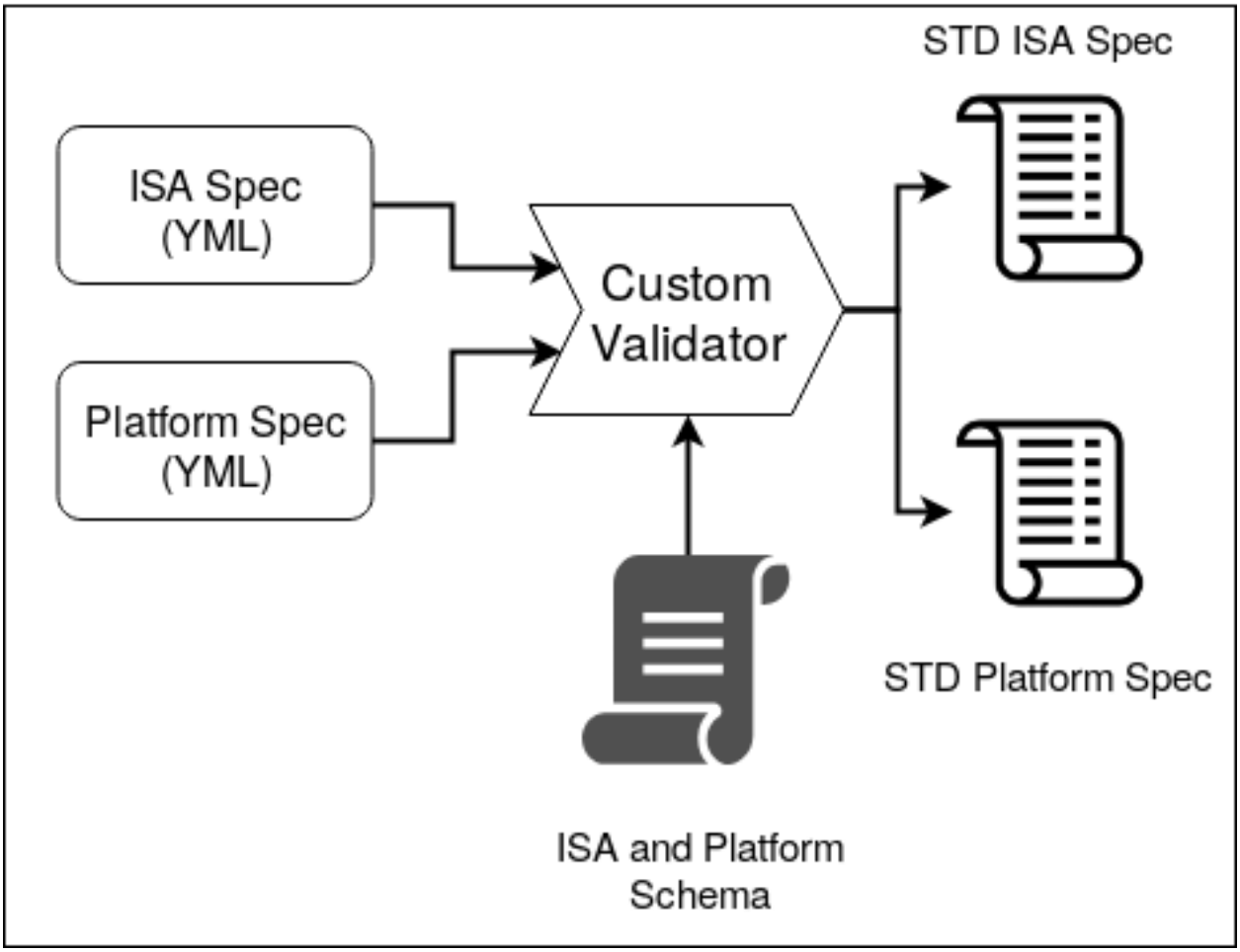
RISCV-Config [Repository]

CHAPTER

TWO

OVERVIEW

The following diagram captures the overall-flow of RISCV-Config.



The user is required to provide 2 YAML files as input:

1. **ISA Spec**: This YAML file is meant to capture the ISA related features implemented by the user. Details of this input file can be found here : *ISA YAML Spec*.
2. **Platform Spec**: This YAML file is meant to capture the platform specific features implemented by the user. Details of this input file can be found here : *Platform YAML Spec*.

2.1 Working:

The ISA and Platform spec are first checked by the validator for any inconsistencies. Checks like 'F' to exist for 'D' are performed by the validator. The validator exits with an error if any illegal configuration for the spec is provided. Once the validator checks pass, two separate standard yaml files are generated, one for each input type. These standard yaml files contain all fields elaborated and additional info for each node. While the user need not specify all the fields in the input yaml files, the validator will assign defaults to those fields and generate a standard exhaustive yaml for both ISA and Platform spec.

CHAPTER

THREE

QUICKSTART

3.1 Installation and Setup

1. Install riscv_config

   Before proceding further please ensure *pip* and *python (>3.7.0)* is installed and configured.

   In case you have issues installing python-3.7, we recommend using *pyenv*.

   Installing instructions for pyenv:

```
#!/bin/sh
curl -L https://raw.githubusercontent.com/yyuu/pyenv-installer/master/bin/
↪pyenv-installer | bash
echo "export PATH=\"/home/$USER/.pyenv/bin:$PATH\"" >> ~/.bashrc
pyenv install 3.7.0
pyenv global 3.7.0
pip install --upgrade pip
```

   You can simply use pip and python for 3.7 by default.

   Support exists for python versions > 3.7.0 only. Please ensure correct version before proceding further.

   - Install using pip(For users):

```
pip3 install riscv_config
```

   - Clone from git(For developers):

```
git clone https://gitlab.com/incoresemi/riscv_config.git
cd riscv_config
pip3 install -r requirements.txt
```

3.2 Usage

- For users

```
riscv-config [-h] --isa_spec YAML --platform_spec YAML [--verbose]

RISC-V Configuration Validator

optional arguments:
```

(continues on next page)

(continued from previous page)

```
  --isa_spec YAML, -ispec YAML
                        The YAML which contains the ISA specs.
  --platform_spec YAML, -pspec YAML
                        The YAML which contains the Platfrorm specs.
  --verbose             debug | info | warning | error
  -h, --help            show this help message and exit
```

- For developers

```
cd riscv_config/

python3 -m riscv_config.main -h
usage: [-h] --isa_spec YAML --platform_spec YAML [--verbose]

RISC-V Configuration Validator

optional arguments:
  --isa_spec YAML, -ispec YAML
                        The YAML which contains the ISA specs.
  --platform_spec YAML, -pspec YAML
                        The YAML which contains the Platfrorm specs.
  --verbose             debug | info | warning | error
  -h, --help            show this help message and exit
```

3.3 Example

- For users

```
git clone https://gitlab.com/incoresemi/riscv_config.git

cd riscv_config/

riscv-config -ispec ./examples/template_isa.yaml -pspec ./examples/template_
↪platform.yaml
```

- For developers

```
cd riscv_config/

python3 -m riscv_config.main -ispec ./examples/template_isa.yaml -pspec ./
↪examples/template_platform.yaml
```

CHAPTER

FOUR

YAML SPECIFICATIONS

This section provides details of the ISA and Platform spec YAML files that need to be provided by the user.

4.1 WARL field Restriction Proposal

Since the RISC-V privilege spec indicates several CSRs and sub-fields of CSRs to be WARL (Write-Any-Read-Legal), it is now necessary to provide a scheme of WARL functions which can be used to precisely define the functionality of any such WARL field/register.

The following proposal for WARL functions was made by **Allen Baum (: esperanto)** and has been adopted in this framework.

1. **Range** (*range-warl-func*)

- represented as a list of 2 or 1 element list where each represents a set.
- Legal values are defined as every value present in the union of disjoint sets represented in the list.
- Each set is represented as (lower,upper) i.e any value>=lower and value<=upper belongs to the set.
- When an illegal value is written (*WriteVal*) to this field, the next valid value of the field can be deduced based on the following modes(*range-update-warl-func*):
  - Unchanged: The value remains unchanged
  - Nextup: ceiling(*WriteVal*) i.e. the next larger or the largest element of the list
  - Nextdown: floor(*WriteVal*) i.e. the next smallest or the smallest element of the list
  - Nearup: celing(*WriteVal*) i.e. the closest element in the list, with the larger element being chosen in case of a tie.
  - Neardown: floor(*WriteVal*) i.e. the closes element in the list, with the smaller element being chosen in case of a tie
  - Largest: maximum of all legal values
  - Smallest: minimum of all legal values
  - Addr:

```
if ( WriteVal < base || WriteVal > bound)
   return Flip-MSB of field
```

Example:

```
range:
  rangelist: [[256,300],[25],[30],[350,390]]
  mode: Addr
```

2. **Bitmask** (*bitmask-warl-func*)

- This function is represented with 2 fields: the *mask* and the *default*
- For the read only positions, the corresponding bits are cleared (=0) in the *mask* and the rest of the bits are set (=1).
- In the *default* field the values for the read only bits are given (= 0 or 1) and the rest of the bits are cleared (=0).

Example:

```
bitmask:
  mask: 0x214102D
  default: 0x100
```

4.2 ISA YAML Spec

This section describes each node of the ISA-YAML. For each node, we have identified the fields required from the user and also the various constraints involved.

All fields accept values as integers or hexadecimals(can be used interchangeably) unless specified otherwise.

An elaborate example of the full-fledge ISA-YAML file can be found here: ISA-YAML

Vendor **Description**: Vendor name.

Examples:

```
Vendor: Shakti
Vendor: Incoresemi
```

Constraints:

- None

Device **Description**: Device Name.

Examples:

```
Device: E-Class
Device: C-Class
```

Constraints:

- None

ISA **Description**: Takes input a string representing the ISA supported by the implementation. All extension names (other than Zext) should be mentioned in upper-case. Z extensions should begin with an upper-case 'Z' followed by lower-case extension name (without Camel casing)

Examples:

```
ISA: RV32IMA
ISA: RV64IMAFDCZifencei
```

Constraints:

- Certain extensions are only valid in certain user-spec version. For, eg. Zifencei is available only in user-spec 2.3 and above.
- The ISA string must be specified in the manner as specified in the specifications(like subsequent Z extensions must be separated with an '_')

User_Spec_Version **Description**: Version number of User/Non-priveleged ISA specification as string. Please enclose the version in "" to avoid type mismatches.

Examples:

```
User_Spec_Version: "2.2"
User_Spec_Version: "2.3"
```

Constraints:

- should be a valid version later than 2.2

Privilege_Spec_Version **Description**: Version number of Priveleged ISA specification as string. Please enclose the version in "" to avoid type mismatches.

Examples:

```
Privilege_Spec_Version: "1.10"
Privilege_Spec_Version: "1.11"
```

Constraints:

- should be a valid version later than 1.10

---

hw_data_misaligned_support **Description**: A boolean value indicating whether hardware support for misaligned load/store requests exists.

Examples:

```
hw_data_misaligned_support: True
hw_data_misaligned_support: False
```

Constraints: - None

---

misa **Description**: Describes the fields of the *misa* CSR. A user needs to provide the following fields:

- implemented: A boolean value indicating if the *misa* has been implemented or not.
- **MXL: needs to be described as *range-warl-func***
  - **range:**
    - rangelist: a list of 2 or 1 element lists which specify the lower and upper bounds of the range of values MXL field can take.(Allowed [1,2,3])
    - mode : a string describing one of the *range-update-warl-func*.

- **Extensions: is described as a *bitmask-warl-func* indicating the valid extensions.**
  - **bitmask:**
    - mask : a 26 bit wide value providing the *mask* field of the *bitmask-warl-func*.
    - default : a 26 bit wide value providing the *default* field of the *bitmask-warl-func*.

Examples:

```
misa:
 implemented: True
 MXL:
     range:
         rangelist : [[1, 2]]
         mode  : 'Unchanged'
 Extensions:
     bitmask:
         mask:   0x12D
         default:  0x000
```

Constraints:

- The maximum value specified in MXL should not be greater than the XLEN/32 specified in the ISA field.
- All extensions defined in the ISA field, cannot be inferred as read-only-0 values.
- All extensions not defined in the ISA field should be inferred as read-only-0 values.
- The default and mask fields of Extensions should be only 26-bits wide.

---

mvendorid

Description: Stores the VendorId.

- implemented: A boolean field indicating that the register has been implemented.
- id: A XLEN bit wide value equal to the VendorId

Examples:

```
mvendorid:
   implemented: True
   id: 0x458
```

Constraints:

- This field should be only XLEN bits wide

---

marchid

Description: Stores the ArchitectureId.

- implemented: A boolean field indicating that the register has been implemented.
- id: A XLEN bit wide value equal to the ArchitectureId

Examples:

```
marchid:
   implemented: True
   id: 0x458
```

Constraints:

- This field should be only XLEN bits wide

---

mimpid

Description: Stores the ImplementationId.

- implemented: A boolean field indicating that the register has been implemented.
- id: A XLEN bit wide value equal to the ImplementationId

Examples:

```
mimpid:
   implemented: True
   id: 0x458
```

Constraints:

- This field should be only XLEN bits wide

---

mhartid **Description**: Specifies the Hart Id of the current specs.

Examples:

```
mhartid: 0
```

Constraints:

- The value should be less than the maximum value supported by XLEN bits.

---

mstatus

Description: Specifies the fields of the *mstatus* register.

- **SD:**
  - is_hardwired: A boolean value indicating if platform has hardwired this field to 0.
- **XS:**
  - is_hardwired: A boolean value indicating if platform has hardwired this field to 0.
- **FS:**
  - **range: (range-warl-func)**
    - rangelist: a list of 2 or 1 element lists which specify the lower and upper bounds of the disjoint set of values FS field can take.(Allowed [0,1,2,3])
    - mode : A string describing one of the *range-update-warl-func*.
- **MPP:**
  - **range: (range-warl-func)**
    - rangelist: a list of 2 or 1 element lists which specify the lower and upper bounds of the disjoint set of values MPP field can take.(Allowed [0,1,3])
    - mode : A string describing one of the *range-update-warl-func*.
- **SXL: This field doesnt exist in systems where XLEN = 32**

  - is_hardwired : A boolean value indicating if platform has hardwired this field to some other field.
  - hardwired_field: A string field indicating to which field SXL is hardwired. (Allowed MXL)
  - **range: (range-warl-func) when not hardwired to MXL**
    - rangelist: a list of 2 or 1 element lists which specify the lower and upper bounds of the disjoint set of values SXL field can take.(Allowed [0,1,2,3])
    - mode : A string describing one of the *range-update-warl-func*.

- **UXL: This field doesnt exist in systems where XLEN = 32**
  - is_hardwired : A boolean value indicating if platform has hardwired this field to some other field.
  - hardwired_field: A string field indicating to which field UXL is hardwired. (Allowed [MXL,SXL])
  - **range: (range-warl-func)**
    - rangelist: a list of 2 or 1 element lists which specify the lower and upper bounds of the disjoint set of values UXL field can take.(Allowed [0,1,2,3])
    - mode : A string describing one of the *range-update-warl-func*.

Examples:

```
mstatus:
   SD:
       is_hardwired: False
   XS:
       is_hardwired: False
   FS:
       range:
           rangelist: [[0,3]]
           mode: "Unchanged"
   MPP:
        range:
           rangelist: [[0],[3]]
           mode: "Unchanged"
   SXL:
        is_hardwired: False
        hardwired_field: "MXL"
   UXL:
        is_hardwired: False
        hardwired_field: "MXL"
```

Constraints:

- if XS is hardwired to 0 and FS is hardwired to 0, then SD should also be hardwired to 0
- if FS field is_hardwired is True, then the range field is ignored by the tests.
- No mode corresponding to a value in MPP must be unsupported (i.e. 1 cannot be present without the 'S' extension and '0' without the 'U' extension)
- No value in SXL or UXL must exceed XLEN/32.
- In 32 bit systems the SXL and UXL fields have no meaning and are ignored while testing.
- In systems other than RV32 for SXL and UXL 0 is allowed to be present only if the corresponding modes are not supported.
- In systems other than RV32 for SXL and UXL if 0 is present in the range list then no other entries are allowed.

---

mtvec

Description: Specifies the fields of the *mtvec* register.

- **BASE: needs to be described as *range-warl-func* providing two integers specifying the range of legal values.**
  - **range: (range-warl-func)**
    - rangelist: a list of 2 or 1 element lists which specify the lower and upper bounds of the disjoint set of values BASE field can take.
    - mode : A string describing one of the *range-update-warl-func*.
- **MODE: needs to be described as *range-warl-func***
  - **range: (range-warl-func)**
    - rangelist: a list of 2 or 1 element lists which specify the lower and upper bounds of the disjoint set of values MODE field can take.(Allowed [0,1])
    - mode : A string describing one of the *range-update-warl-func*.

Examples:

```
mtvec:
   BASE:
       range:
           rangelist: [[1000]]
           mode: 'Unchanged'
   MODE:
        range:
           rangelist: [[0,1]]
           mode: "Unchanged"
```

Constraints:

- The maximum in the list of values specified for MODE cannot exceed 1.
- No value in the list of legal values can exceed 2^(XLEN-2)-1 for BASE.

---

mideleg

Description: needs to be described as *bitmask-warl-func* indicating delegatable interrupts

- **bitmask:**
  - mask : a XLEN bit wide value providing the *mask* field of the *bitmask-warl-func*.
  - default : a XLEN bit wide value providing the *default* field of the *bitmask-warl-func*.

Examples:

```
mideleg:
   bitmask:
       mask: 0xFFFFFFFF
       default: 0x00
```

Constraints:

- No bit can be hardwired to 1 i.e the any bit which is 0 in the mask cannot be 1 in the default.

- The mask and default cannot exceed the maximum value which can be represented by XLEN bits(2^XLEN-1).

---

medeleg

Description: needs to be described as *bitmask-warl-func* indicating delegatable exceptions

- **bitmask:**
  - mask : a XLEN bit wide value providing the *mask* field of the *bitmask-warl-func*.
  - default : a XLEN bit wide value providing the *default* field of the *bitmask-warl-func*.

Examples:

```
medeleg:
   bitmask:
       mask: 0xFFFFF7FF
       default: 0x00
```

Constraints:

- No bit can be hardwired to 1 i.e the any bit which is 0 in the mask cannot be 1 in the default.
- The mask and default cannot exceed the maximum value which can be represented by XLEN bits(2^XLEN-1).
- The 11th bit needs to be hardwired to 0 ie. 0 in position 11 in both mask and default.

---

mip

Description: needs to be described as *bitmask-warl-func* indicating allowed interrupts.

- **bitmask:**
  - mask : a XLEN bit wide value providing the *mask* field of the *bitmask-warl-func*.
  - default : a XLEN bit wide value providing the *default* field of the *bitmask-warl-func*.

Examples:

```
mip:
   bitmask:
       mask: 0xFFFFFFFF
       default: 0x00
```

Constraints:

- No bit can be hardwired to 1 i.e the any bit which is 0 in the mask cannot be 1 in the default.
- The mask and default cannot exceed the maximum default which can be represented by XLEN bits(2^XLEN-1).

---

mie

Description: needs to be described as *bitmask-warl-func* indicating allowed interrupts.

- **bitmask:**
  - mask : a XLEN bit wide value providing the *mask* field of the *bitmask-warl-func*.

  - default : a XLEN bit wide value providing the *default* field of the *bitmask-warl-func*.

Examples:

```
mie:
   bitmask:
       mask: 0xFFFFFFFF
       default: 0x00
```

Constraints:

- No bit can be hardwired to 1 i.e the any bit which is 0 in the mask cannot be 1 in the default.
- The mask and default cannot exceed the maximum value which can be represented by XLEN bits(2^XLEN-1).

---

mepc

Description: needs to be described as *bitmask-warl-func* or a *range-warl-func* indicating the range of

legal values allowed

- **bitmask:**
  - mask : a XLEN bit wide value in hexa-decimal providing the *mask* field of the *bitmask-warl-func*.
  - default : a XLEN bit wide value in hexa-decimal providing the *default* field of the *bitmask-warl-func*.
- **range: (range-warl-func)**
  - rangelist: a list of 2 or 1 element lists which specify the lower and upper bounds of the disjoint set of values mepc can take.
  - mode : A string describing one of the *range-update-warl-func*.

Examples:

```
mepc:
   range:
       rangelist: [[0xFFFFFFFF,0x80000]]
       mode: "Unchanged"
```

Constraints:

- The mask and default cannot exceed the maximum value which can be represented by XLEN bits(2^XLEN-1) (in case of *bitmask-warl-func*).
- No legal value can exceed the maximum value which can be represented by XLEN bits(2^XLEN-1) (in case of *range-warl-func*).

---

mcountinhibit

Description: needs to be described as *bitmask-warl-func* indicating supported counters.

- **bitmask:**
  - mask : a 32 bit wide value providing the *mask* field of the *bitmask-warl-func*.
  - default : a 32 bit wide value providing the *default* field of the *bitmask-warl-func*.

Examples:

```
mcountinhibit:
   bitmask:
       mask: 0xFFFFFFFF
       default: 0x00
```

Constraints:

- The mask and default cannot exceed the maximum value which can be represented by XLEN bits(2^XLEN-1).

---

mcounteren

Description: needs to be described as *bitmask-warl-func* indicating supported counters.

- **bitmask:**
  - mask : a 32 bit wide value providing the *mask* field of the *bitmask-warl-func*.
  - default : a 32 bit wide value providing the *default* field of the *bitmask-warl-func*.
- is_hardwired: a boolean value indicating whether the register is hardwired or not.
- hardwired_val: an integer indicating to what value the register is hardwired to.

Examples:

```
mcounteren:
   bitmask:
       mask: 0xFFFFFFFF
       default: 0x00
```

Constraints:

- The mask and default cannot exceed the maximum value which can be represented by XLEN bits(2^XLEN-1).

---

mcycle

Description:

- is_hardwired: a string indicating whether the register is hardwired or not.

Other counters (minstret,mhpmcounter3 - mhpmcounter31) are analogously defined.

Examples:

```
mcycle:
    is_hardwired: True
    hardwired_val: 0
```

Constraints:

- None

---

stvec

Description: Specifies the fields of the *stvec* register.

- **BASE: needs to be described as *range-warl-func* providing two integers specifying the range of legal values.**

  - **range: (range-warl-func)**
    - rangelist: a list of 2 or 1 element lists which specify the lower and upper bounds of the disjoint set of values BASE field can take.
    - mode : A string describing one of the *range-update-warl-func*.
- **MODE: needs to be described as *range-warl-func***
  - **range: (range-warl-func)**
    - rangelist: a list of 2 or 1 element lists which specify the lower and upper bounds of the disjoint set of values MODE field can take.(Allowed [0,1])
    - mode : A string describing one of the *range-update-warl-func*.

Examples:

```
stvec:
   BASE:
       range:
           rangelist: [[1000]]
           mode: 'Unchanged'
   MODE:
        range:
           rangelist: [[0,1]]
           mode: "Unchanged"
```

Constraints:

- The maximum in the list of values specified for MODE cannot exceed 1.
- No value in the list of legal values can exceed 2^(XLEN-2) for BASE.

---

sip

Description: needs to be described as *bitmask-warl-func* indicating allowed interrupts in 'S' mode.

- **bitmask:**
  - mask : a XLEN bit wide value providing the *mask* field of the *bitmask-warl-func*.
  - default : a XLEN bit wide value providing the *default* field of the *bitmask-warl-func*.

Examples:

```
sip:
   bitmask:
       mask: 0xFFFFFFFF
       default: 0x00
```

Constraints:

- No bit can be hardwired to 1 i.e the any bit which is 0 in the mask cannot be 1 in the default.
- The mask and default cannot exceed the maximum default which can be represented by XLEN bits(2^XLEN-1).

---

sie

Description: needs to be described as *bitmask-warl-func* indicating allowed interrupts in 'S' mode.

- **bitmask:**
  - mask : a XLEN bit wide value providing the *mask* field of the *bitmask-warl-func*.
  - default : a XLEN bit wide value providing the *default* field of the *bitmask-warl-func*.

Examples:

```
sie:
   bitmask:
       mask: 0xFFFFFFFF
       default: 0x00
```

Constraints:

- No bit can be hardwired to 1 i.e the any bit which is 0 in the mask cannot be 1 in the default.
- The mask and default cannot exceed the maximum value which can be represented by XLEN bits(2^XLEN-1).

---

scounteren

Description: needs to be described as *bitmask-warl-func* indicating supported counters in 'S' mode.

- **bitmask:**
  - mask : a 32 bit wide value providing the *mask* field of the *bitmask-warl-func*.
  - default : a 32 bit wide value providing the *default* field of the *bitmask-warl-func*.
- is_hardwired: a boolean value indicating whether the register is hardwired or not.
- hardwired_val: an integer indicating to what value the register is hardwired to.

Examples:

```
scounteren:
   bitmask:
       mask: 0xFFFFFFFF
       default: 0x00
```

Constraints:

- The mask and default cannot exceed the maximum value which can be represented by XLEN bits(2^XLEN-1).

---

sepc

Description: needs to be described as *bitmask-warl-func* or a *range-warl-func* indicating the range of

legal values allowed

- **bitmask:**
  - mask : a XLEN bit wide value in hexa-decimal providing the *mask* field of the *bitmask-warl-func*.
  - default : a XLEN bit wide value in hexa-decimal providing the *default* field of the *bitmask-warl-func*.
- **range: (range-warl-func)**
  - rangelist: a list of 2 or 1 element lists which specify the lower and upper bounds of the disjoint set of values sepc can take.

  - mode : A string describing one of the *range-update-warl-func*.

Examples:

```
sepc:
   range:
       rangelist: [[0xFFFFFFFF,0x80000]]
       mode: "Unchanged"
```

Constraints:

- The mask and default cannot exceed the maximum value which can be represented by XLEN bits(2^XLEN-1) (in case of *bitmask-warl-func*).
- No legal value can exceed the maximum value which can be represented by XLEN bits(2^XLEN-1) (in case of *range-warl-func*).

---

satp

Description: This field describes the *satp* register

- **MODE:**
  - **range: (range-warl-func)**
    - rangelist: a list of 2 or 1 element lists which specify the lower and upper bounds of the disjoint set of values MODE can take.
    - mode : A string describing one of the *range-update-warl-func*.
- **ASID:**
  - **bitmask:**
    - mask : a ASIDLEN bit wide value in hexa-decimal providing the *mask* field of the *bitmask-warl-func*.
    - default : a ASIDLEN bit wide value in hexa-decimal providing the *default* field of the *bitmask-warl-func*.
  - **range: (range-warl-func)**
    - rangelist: a list of 2 or 1 element lists which specify the lower and upper bounds of the disjoint set of values ASID can take.
    - mode : A string describing one of the *range-update-warl-func*.
- **PPN:**
  - **bitmask:**
    - mask : a value in hexa-decimal providing the *mask* field of the *bitmask-warl-func*.
    - default : a value in hexa-decimal providing the *default* field of the *bitmask-warl-func*.
  - **range: (range-warl-func)**
    - rangelist: a list of 2 or 1 element lists which specify the lower and upper bounds of the disjoint set of values PPN can take.
    - mode : A string describing one of the *range-update-warl-func*.

Examples:

```
satp:
  MODE:
    range:
      rangelist: [[0],[8]]
      mode: "Unchanged"
  ASID:
    bitmask:
      mask: 0x1FF
      default: 0x00
  PPN:
    bitmask:
      mask: 0xFFFFFFFFFFF
      default: 0x00
```

Constraints:

- On 32 bit systems ASIDMAX is 9 and on 64 bit systems ASIDMAX is 16.
- On 32 bit systems MODE can take any value in [0,1] and on 64 bit systems MODE can take any value in [0,8,9].
- Rangelist of MODE cannot have the entry 9 without 8 being present.
- PPN is 22 bits wide in 32 bit systems and 44 bits wide in 64 bit systems.

4.3 Platform YAML Spec

This section describes each node of the PLATFORM-YAML. For each node, we have identified the fields required from the user and also the various constraints involved.

An eloborate example of the full-fledge PLATFORM-YAML file can be found here: PLATFORM-YAML

reset

Description: Stores the value for the reset vector. It can either be a label or an address.

- label: A string field equal to the label in the assembly code.
- address: A value equal to the absolute address where the vector is present.

Examples:

```
reset:
   label: reset_vector

reset:
   address: 0x8000000
```

Constraints:

- None

---

nmi

Description: Stores the value for the nmi vector. It can either be a label or an address.

- label: A string field equal to the label in the assembly code.

- address: A value equal to the absolute address where the vector is present.

Examples:

```
nmi:
   label: nmi_vector

nmi:
   address: 0x8000000
```

Constraints:

- None

---

mtime

Description: Stores the fields for memory mapped *mtime* register.

- implemented: A boolean field indicating that the register has been implemented.
- address: A value equal to the physical address at which the register is present.

Examples:

```
mtime:
   implemented: True
   address: 0x458
```

Constraints:

- None

---

mtimecmp

Description: Stores the fields for memory mapped *mtimecmp* register.

- implemented: A boolean field indicating that the register has been implemented.
- address: A value equal to the physical address at which the register is present.

Examples:

```
mtimecmp:
   implemented: True
   address: 0x458
```

Constraints:

- None

---

mcause

Description: Stores the fields for the *mcause* register.

- implemented: A boolean field indicating that the register has been implemented.
- values: The list of exception values greater than 16 as assumed by the platform as integers.

Examples:

```
mcause:
   implemented: True
   value: [16,17,20]
```

Constraints:

- None

---

mtval

Description: Stores the fields for *mtval* register.

- **behaviour: A dictionary type to specify which of the exceptions modify the mtval reg**
  - e0: A string type describing the behaviour of exception 0.
  - e1: A string type describing the behaviour of exception 1.
  - e2: A string type describing the behaviour of exception 2.
  - e3: A string type describing the behaviour of exception 3.
  - e4: A string type describing the behaviour of exception 4.
  - e5: A string type describing the behaviour of exception 5.
  - e6: A string type describing the behaviour of exception 6.
  - e7: A string type describing the behaviour of exception 7.
  - e8: A string type describing the behaviour of exception 8.
  - e9: A string type describing the behaviour of exception 9.
  - e10: A string type describing the behaviour of exception 10.
  - e11: A string type describing the behaviour of exception 11.
  - e12: A string type describing the behaviour of exception 12.
  - e13: A string type describing the behaviour of exception 13.
  - e15: A string type describing the behaviour of exception 15.

Examples:

```
TBD: Provide a concrete use-case for the above.
```

Constraints:

- None

---

mhpmevent3

Description: is a *warl-func* and can be specified as any of the three WARL functions.

- **range: (range-warl-func)**
  - rangelist: a list of 2 or 1 element lists which specify the lower and upper bounds of the disjoint set of values mphmevent3 can take.(Allowed [0,1,3])
  - mode : A string describing one of the *range-update-warl-func*.
- **distinct:**

  - values: A list of legal values that the register can take.(Less than 2^XLEN)
  - mode : A string describing one of the *distinct-update-warl-func*.
- is_hardwired: A boolean field indicating whether the register is hardwired or not.

Other event registers (mhpmevent4 - mhpmevent31) are analogously defined.

Examples:

```
mhpmevent3:
   is_hardwired: True

mhpmevent3:
   range:
      rangelist: [[0],[5,6],[8,9]]
      mode: "Unchanged"
```

Constraints:

- The mask and default (incase of bitmask func) cannot exceed the maximum value which can be represented by XLEN bits.
- No legal value can exceed the maximum value which can be represented by XLEN bits.

---

scause

Description: Stores the fields for the *scause* register.

- implemented: A boolean field indicating that the register has been implemented.
- values: The list of exception values greater than 16 as assumed by the platform as integers.

Examples:

```
scause:
   implemented: True
   value: [16,17,20]
```

Constraints:

- None

---

stval

Description: Stores the fields for *stval* register.

- **behaviour: A dictionary type to specify which of the exceptions modify the mtval reg**
  - e0: A string type describing the behaviour of exception 0.
  - e1: A string type describing the behaviour of exception 1.
  - e2: A string type describing the behaviour of exception 2.
  - e3: A string type describing the behaviour of exception 3.
  - e4: A string type describing the behaviour of exception 4.
  - e5: A string type describing the behaviour of exception 5.
  - e6: A string type describing the behaviour of exception 6.
  - e7: A string type describing the behaviour of exception 7.

- e8: A string type describing the behaviour of exception 8.
- e9: A string type describing the behaviour of exception 9.
- e10: A string type describing the behaviour of exception 10.
- e11: A string type describing the behaviour of exception 11.
- e12: A string type describing the behaviour of exception 12.
- e13: A string type describing the behaviour of exception 13.
- e15: A string type describing the behaviour of exception 15.

Examples:

```
TBD: Provide a concrete use-case for the above.
```

Constraints:

- None

CHAPTER

FIVE

CODE DOCUMENTATION

riscv_config.checker.**add_def_setters**(*schema_yaml*)
Function to set the default setters for various fields in the schema

riscv_config.checker.**check_specs**(*isa_spec*, *platform_spec*, *work_dir*)
Function to perform ensure that the isa and platform specifications confirm to their schemas. The Cerberus module is used to validate that the specifications confirm to their respective schemas.

Parameters

- **isa_spec** (*str*) – The path to the DUT isa specification yaml file.
- **platform_spec** (*str*) – The path to the DUT platform specification yaml file.

Raises **ValidationError** – It is raised when the specifications violate the schema rules.

Returns A tuple with the first entry being the path to normalized isa file and the second being path to the platform spec file.

riscv_config.checker.**errPrint**(*foo*, *space=' '*)
Function to petty print the error from cerberus.

riscv_config.checker.**imp_normalise**(*foo*)
Function to trim the dictionary. Any node with implemented field set to false is trimmed of all the other nodes.

Parameters **foo** (*dict*) – The dictionary to be trimmed.

Returns The trimmed dictionary.

riscv_config.checker.**iset**()
Function to check and set defaults for all "implemented" fields which are dependent on the xlen.

riscv_config.checker.**miedelegset**()
Function to set "implemented" value for mideleg regisrer.

riscv_config.checker.**nosset**()
Function to check and set defaults for all fields which are dependent on the presence of 'S' extension and have a hardwired value of 0.

riscv_config.checker.**nouset**()
Function to check and set defaults for all fields which are dependent on the presence of 'U' extension and have a hardwired value of 0.

riscv_config.checker.**twset**()
Function to check and set value for tw field in misa.

riscv_config.checker.**uieset**(*doc*)
Function to check and set value for uie field in misa.

riscv_config.checker.**upieset**(*doc*)
Function to check and set value for upie field in misa.

class riscv_config.schemaValidator.**schemaValidator**(**args*, ***kwargs*)
Custom validator for schema having the custom rules necessary for implementation and checks.

__init__(**args*, ***kwargs*)
The arguments will be treated as with this signature:

__init__(self, schema=None, ignore_none_values=False, allow_unknown=False, require_all=False, purge_unknown=False, purge_readonly=False, error_handler=errors.BasicErrorHandler)

_check_with_capture_isa_specifics(*field*, *value*)
Function to extract and store ISA specific information(such as xlen,user spec version and extensions present) and check whether the dependencies in ISA extensions are satisfied.

_check_with_ext_check(*field*, *value*)
Function to check whether the bitmask given for the Extensions field in misa is valid.

_check_with_hardwirecheck(*field*, *value*)
Function to check that none of the bits in the field are hardwired to 1

_check_with_len_check(*field*, *value*)
Function to check whether the given value is less than XLEN/32(For check).

_check_with_max_length(*field*, *value*)
Function to check whether the given value is less than the maximum value that can be stored(2^xlen-1).

_check_with_medelegcheck(*field*, *value*)
Function to check that the input given for medeleg satisfies the constraints

_check_with_mpp_check(*field*, *value*)
Function to check whether the modes specified in MPP field in mstatus is supported

_check_with_priv_version_check(*field*, *value*)
Function to check whether the Privileged spec version specified is valid or not.

_check_with_rangecheck(*field*, *value*)
Function to check whether the inputs in range type in WARL fields are valid.

_check_with_sxl_check(*field*, *value*)
Function to check whether the input list for SXL field is valid.

_check_with_user_version_check(*field*, *value*)
Function to check whether the User spec version specified is valid or not.

_check_with_uxl_check(*field*, *value*)
Function to check whether the input list for UXL field is valid.

_check_with_xcause_check(*field*, *value*)
Function to verify the inputs for mcause.

_check_with_xtveccheck(*field*, *value*)
Function to check whether the inputs in range type in mtvec are valid.

5.1 Utils

class riscv_config.utils.**ColoredFormatter**(**args*, ***kwargs*)
Class to create a log output which is colored based on level.

__init__(**args*, ***kwargs*)
Initialize the formatter with specified format strings.

Initialize the formatter either with the specified format string, or a default as described above. Allow for specialized date formatting with the optional datefmt argument. If datefmt is omitted, you get an ISO8601-like (or RFC 3339-like) format.

Use a style parameter of '%', '{' or '$' to specify that you want to use one of %-formatting, `str.format()` ({}) formatting or `string.Template` formatting in your format string.

Changed in version 3.2: Added the `style` parameter.

format(*record*)
Format the specified record as text.

The record's attribute dictionary is used as the operand to a string formatting operation which yields the returned string. Before formatting the dictionary, a couple of preparatory steps are carried out. The message attribute of the record is computed using LogRecord.getMessage(). If the formatting string uses the time (as determined by a call to usesTime(), formatTime() is called to format the event time. If there is exception information, it is formatted using formatException() and appended to the message.

class riscv_config.utils.**SortingHelpFormatter**(*prog*, *indent_increment=2*, *max_help_position=24*, *width=None*)

riscv_config.utils.**setup_logging**(*log_level*)
Setup logging

Verbosity decided on user input

Parameters **log_level** (*str*) – User defined log level

CHAPTER

SIX

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

r

INDEX

Symbols

A

C

E

F

I

M

N

R

S

T

U