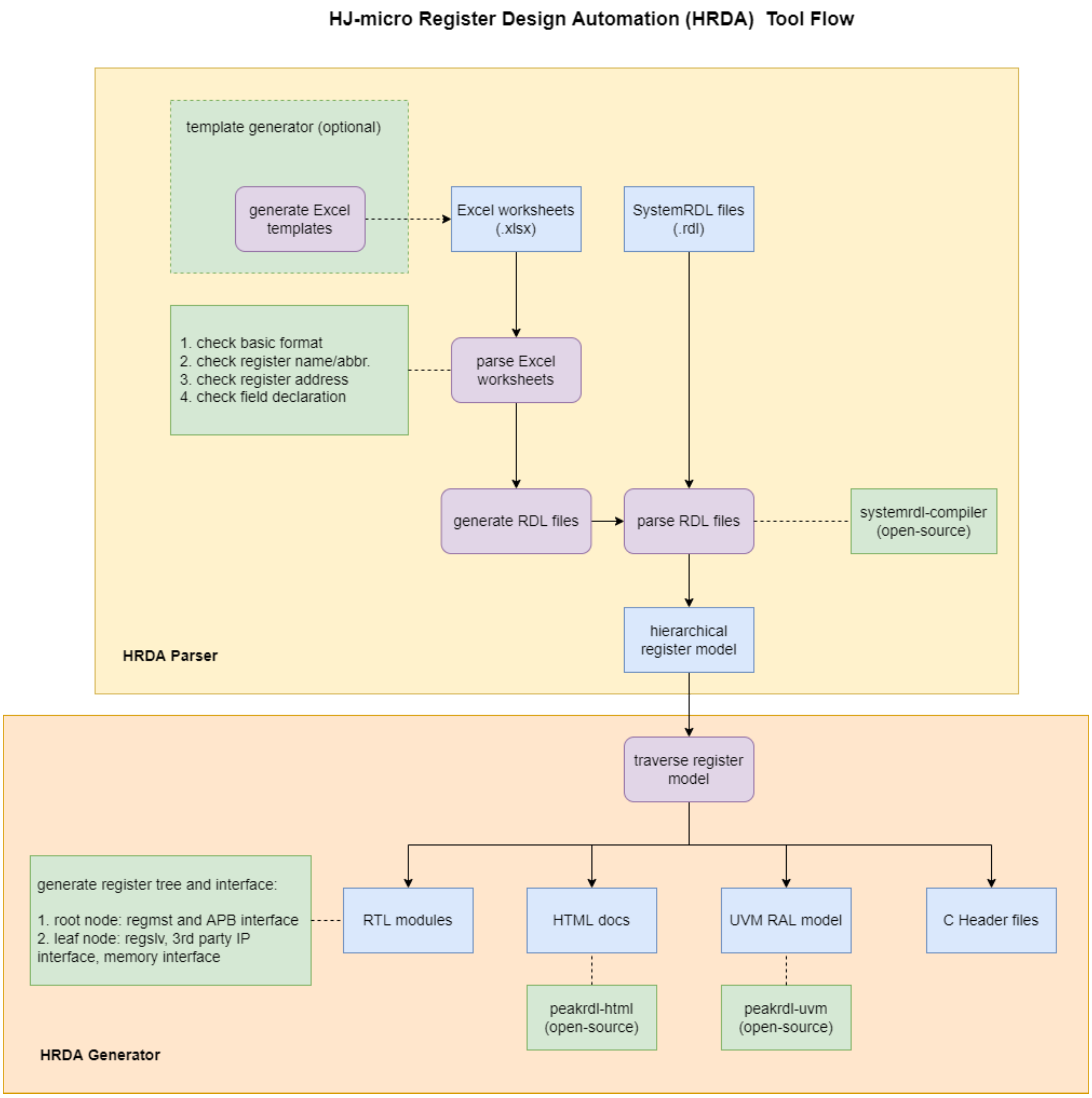
HJ-micro Register Design Automation Tool (HRDA Tool)

Introduction

HJ-micro Register design Automation (HRDA) Tool is a command-line register automation tool developed by Python, which can be divided into two major parts: front-end and back-end. The front-end comprises template generation which supports for generating register description templates in Excel worksheet (.xlsx) format, and Parser which can parse the **input Excel Worksheet or SystemRDL (.rdl) descriptions** with semantic and predefined rule check. The back-end comprises generator abilities supporting for generating register RTL (Verilog/SystemVerilog) Modules, HTML documents, UVM RAL models and C header files.

For modules with few registers and simple address mapping, Excel worksheet is recommended. For some complicated modules with a large amount of registers and fancy mappings, SystemRDL is more expressive and flexible.

The overall tool flow is shown as the figure below.


HJ-micro Register Design Automation (HRDA) Tool Flow
template generator (optional)
generate Excel templates
Excel worksheets (.xlsx)
SystemRDL files (.rdl)
1. check basic format
2. check register name/abbr.
3. check register address
4. check field declaration
parse Excel worksheets
generate RDL files
parse RDL files
systemrdl-compiler (open-source)
hierarchical register model
HRDA Parser
traverse register model
generate register tree and interface:
1. root node: regmst and APB interface
2. leaf node: regslv, 3rd party IP interface, memory interface
RTL modules
HTML docs
UVM RAL model
C Header files
peakrdl-html (open-source)
peakrdl-uvm (open-source)
HRDA Generator

Register Template Generator

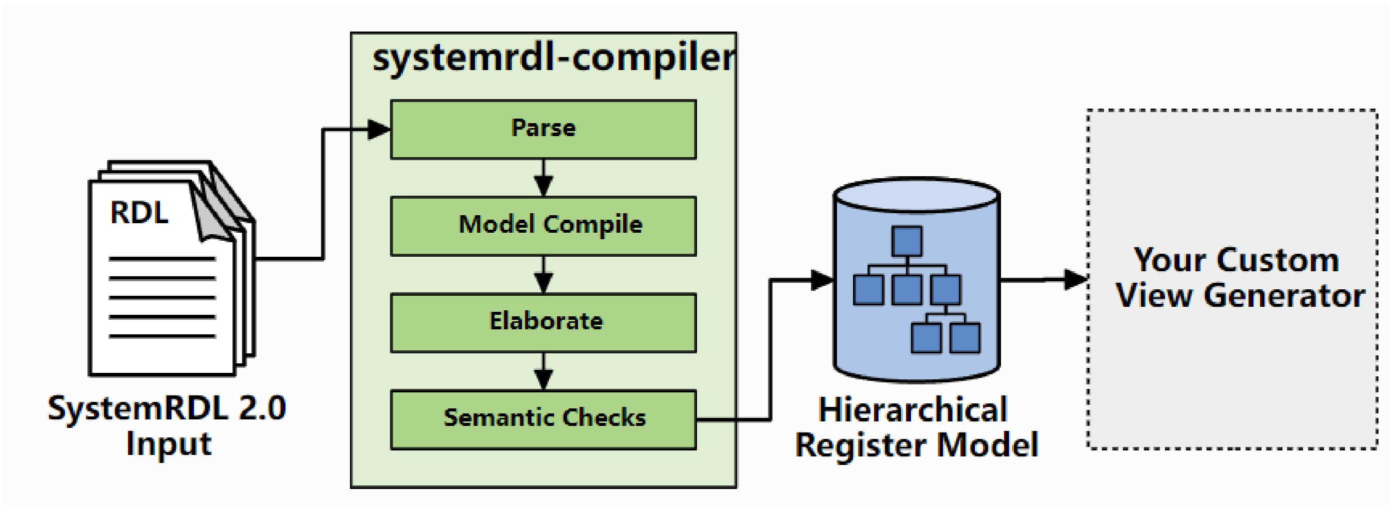
Excel Parser

The Excel parser check all Excel files provided by the designer, including basic format and design rules, and then converts the parsed register specification model into SystemRDL code, which will be submitted to the `SystemRDL Compiler` later. Intermediate SystemRDL code generation also allows the designer to add more complicated features supported by SystemRDL.

To learn what rules are checked, see Chapter **Excel Worksheet Guideline**. If any of rules are violated, Excel parser will raise error and error message will display the position where error occurs.

SystemRDL Parser/Compiler

SystemRDL parser relies on an open-source project `SystemRDL Compiler`, see the link in section 3.1 for detailed information. SystemRDL Compiler is able to parse, compile and check RDL input files followed by SystemRDL 2.0 Spec to generate a traversable hierarchical register model, with the basic workflow shown in the following diagram.

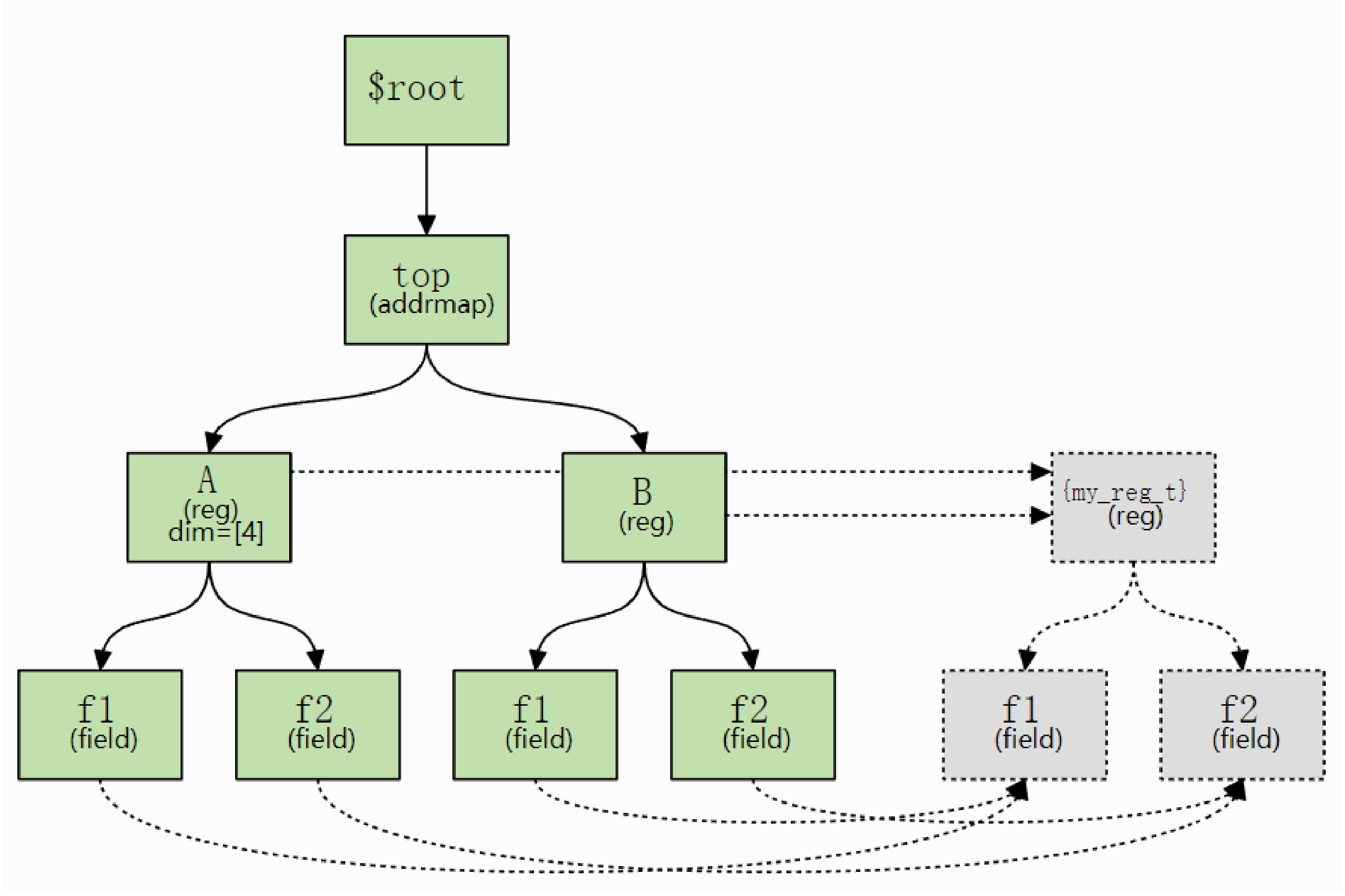


Simple example:

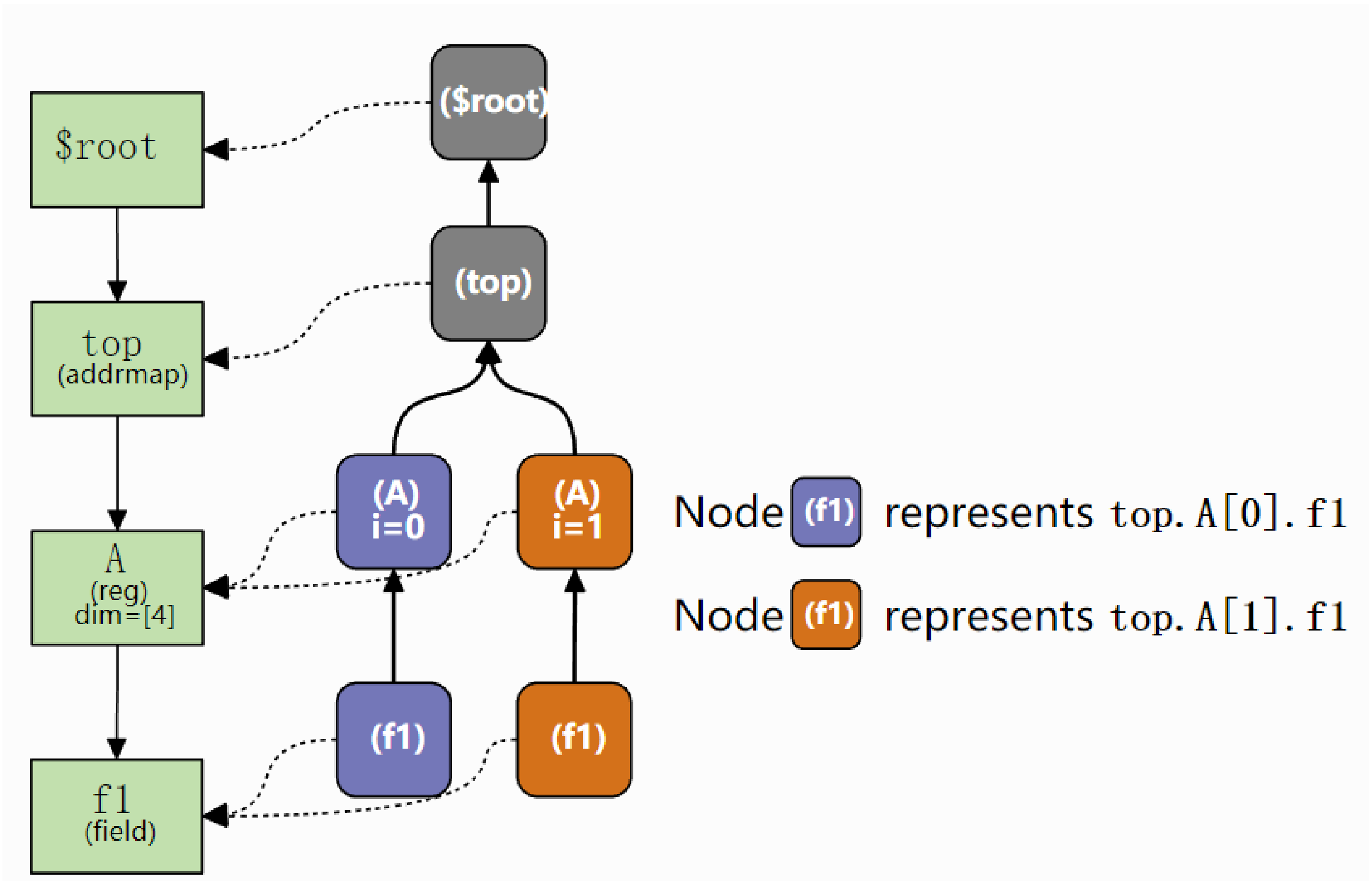
```
reg my_reg_t {
    field {} f1;
    field {} f2;
};

addrmap top {
    my_reg_t A[4];
    my_reg_t B;
};
```

Once compiled, the register model can be described like this:



or another Node overlay:



The model bridges the front-end and the back-end of this tool. The front-end parser ultimately generates this model, and everything on the back-end is based on this model as input.

> For a detailed description of this model, see SystemRDL Compiler Documentation : https://systemrdl-compiler.readthedocs.io/en/stable/index.html

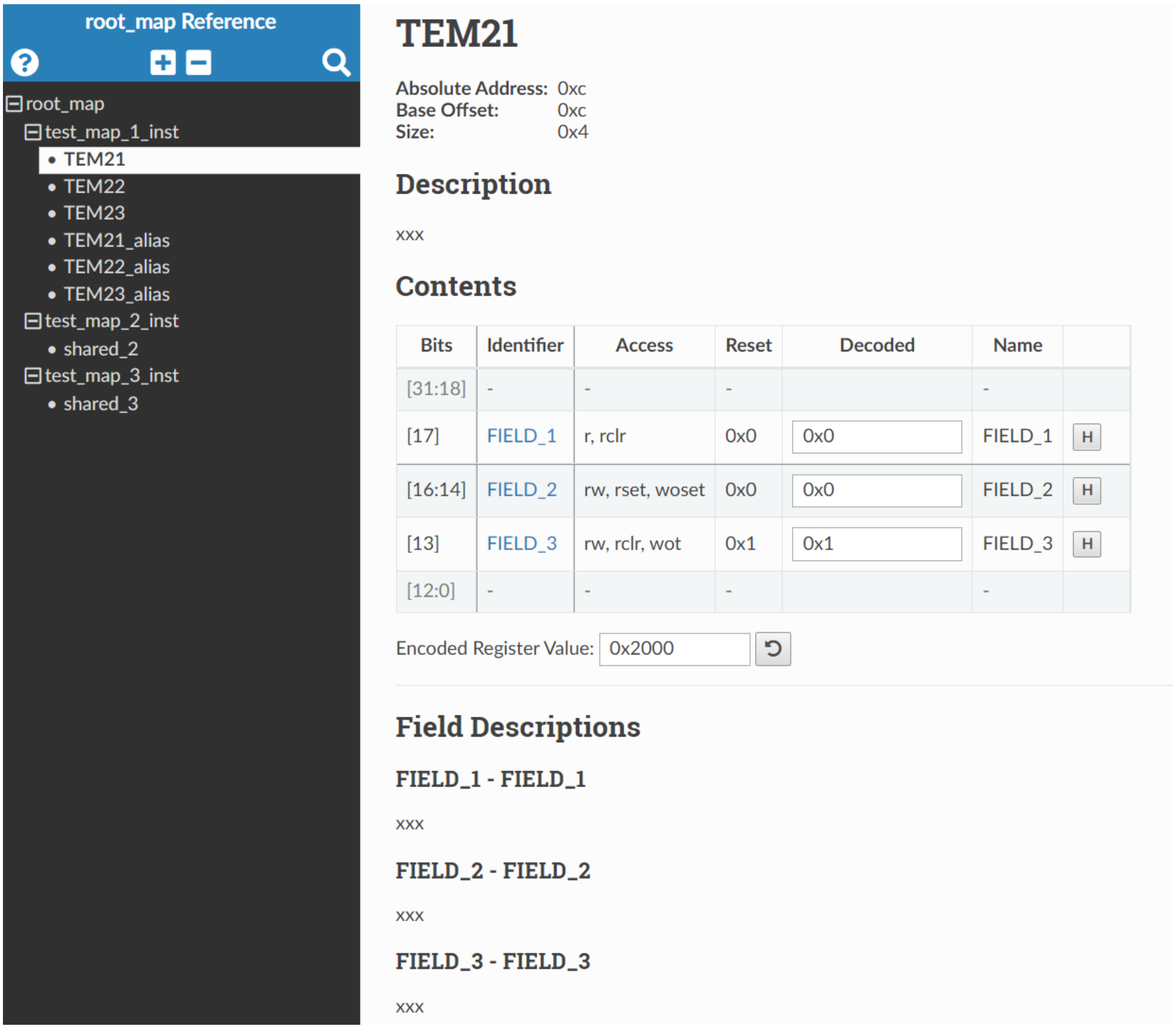
RTL Generator

The RTL Generator is the core functionality of HRDA. It traverses the hierarchical register model and generate corresponding RTL modules.

For detailed RTL architecture information, see Chapter 2.

HTML Generator

The HTML generator relies on the open-source project `PeakRDL-html`, see the link in section 3.1 for detailed information. A simple example of exported HTML is shown below.



TEM21

Absolute Address: 0xc
Base Offset: 0xc
Size: 0x4

Description

xxx

Contents

Bits	Identifier	Access	Reset	Decoded	Name	
[31:18]	-	-	-		-	
[17]	FIELD_1	r, rclr	0x0	0x0	FIELD_1	H
[16:14]	FIELD_2	rw, rset, woset	0x0	0x0	FIELD_2	H
[13]	FIELD_3	rw, rclr, wot	0x1	0x1	FIELD_3	H
[12:0]	-	-	-		-	

Encoded Register Value: 0x2000

Field Descriptions

FIELD_1 - FIELD_1

xxx

FIELD_2 - FIELD_2

xxx

FIELD_3 - FIELD_3

xxx

UVM RAL Generator

The export of the UVM register model relies on the open-source project `PeakRDL-uvm`, see the link in section 3.1 for detailed information.

C Header Generator (TBD)

RTL Architecture

SystemRDL Coding Guideline

SystemRDL is a language for the design and delivery of intellectual property (IP) products used in designs. SystemRDL semantics supports the entire life-cycle of registers from specification, model generation, and design verification to maintenance and documentation. Registers are not just limited to traditional configuration registers, but can also refer to register arrays and memories.

This chapter is based on the SystemRDL 2.0 Specification. In other words, it specifies a subset of SystemRDL syntax and features to use, and some pre-defined properties under this framework. What's more significant, **HRDA Tool only interpret SystemRDL features mentioned in this chapter, namely other features are not supported and make no sense in the tool back-end generation process**.

General Rules

Components

A component in SystemRDL is the basic building block or a container which contains properties that further describe the component’s behavior. There are several structural components in SystemRDL: **field**, **reg**, **mem**, **regfile**, and **addrmap**. All structural components are supported in HRDA Tool, and their mappings to RTL module are as follows:

- **field**: describes fields in registers
- **reg**: describes registers

- **regfile**: pack registers together with support of address allocation
- **addrmap**: similar to **regfile** on packing register and allocating addresses. Additionally, it defines an RTL code generation boundary. Each definition of **addrmap** with ~gencode~ property set will be generated to an ~regslv~ module.

Additionally, HRDA does not support non-structural components, such as **signal**. But signals are indeed used to describe field synchronous resets in a special way: defining a user-defined property, see Chapter

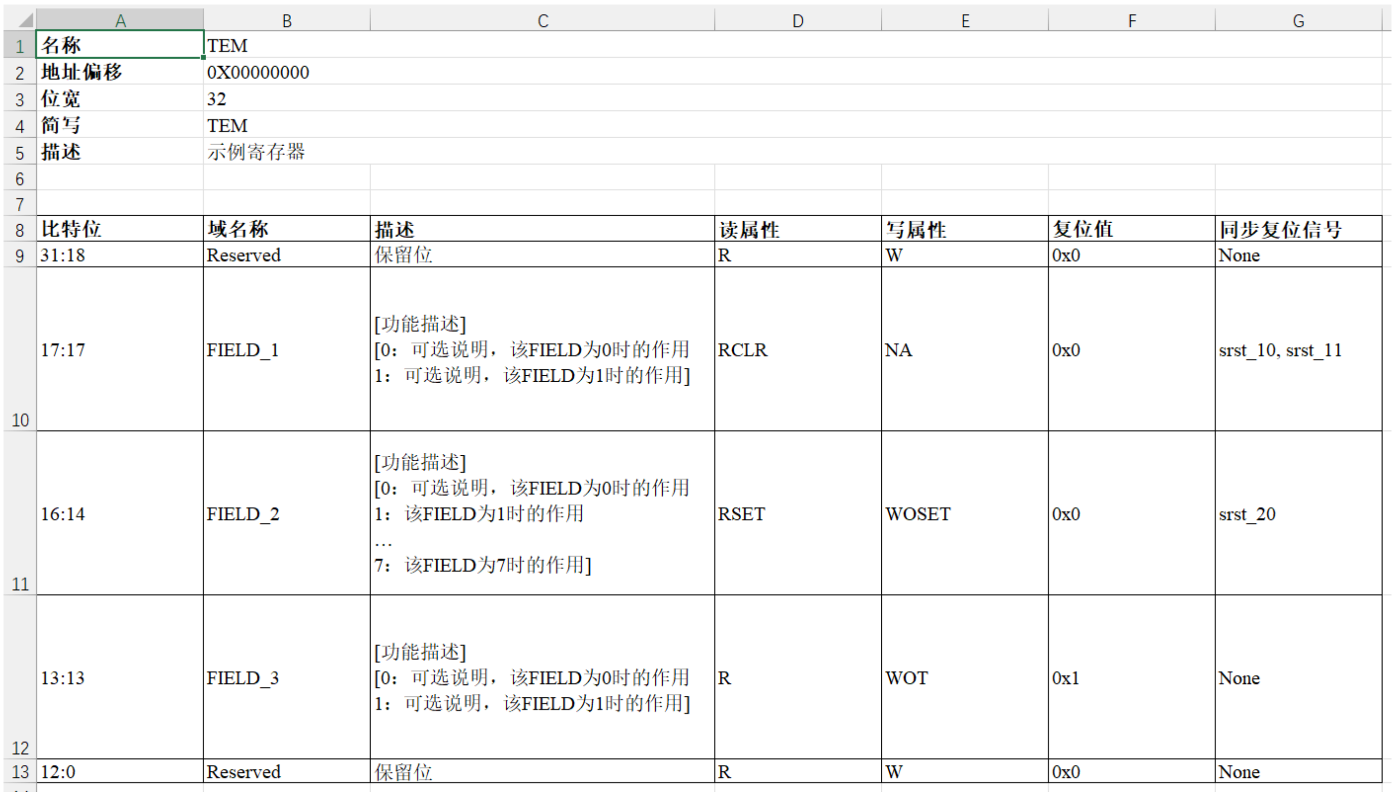
Correspondence to RTL

(TBD)

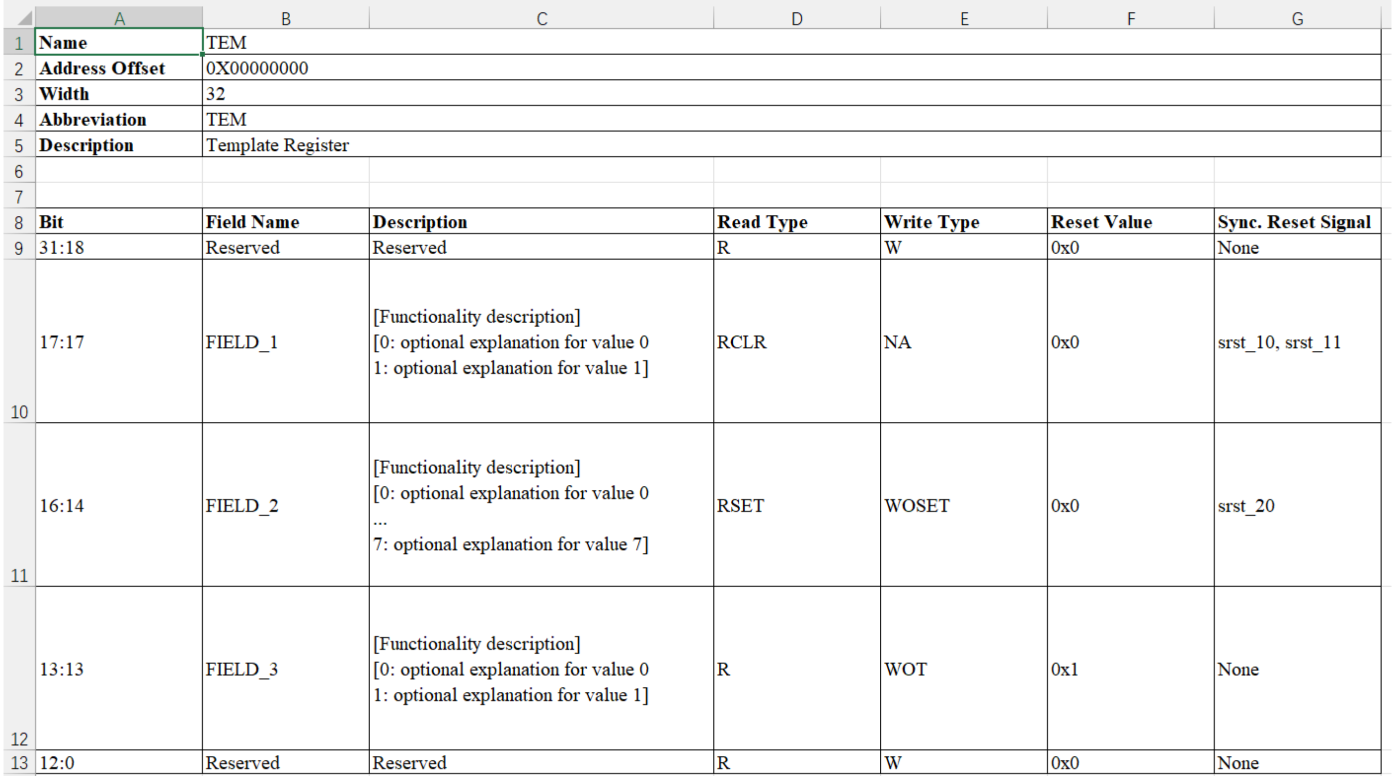
Excel Worksheet Guideline

Table Format

An example Excel worksheet that describes only one register is shown below, in two language (cn/en) versions.



	A	B	C	D	E	F	G
1	**名称**	TEM					
2	**地址偏移**	0X00000000					
3	**位宽**	32					
4	**简写**	TEM					
5	**描述**	示例寄存器					
6							
7							
8	**比特位**	**域名称**	**描述**	**读属性**	**写属性**	**复位值**	**同步复位信号**
9	31:18	Reserved	保留位	R	W	0x0	None
10	17:17	FIELD_1	[功能描述] [0：可选说明，该FIELD为0时的作用 1：可选说明，该FIELD为1时的作用]	RCLR	NA	0x0	srst_10, srst_11
11	16:14	FIELD_2	[功能描述] [0：可选说明，该FIELD为0时的作用 1：该FIELD为1时的作用 … 7：该FIELD为7时的作用]	RSET	WOSET	0x0	srst_20
12	13:13	FIELD_3	[功能描述] [0：可选说明，该FIELD为0时的作用 1：可选说明，该FIELD为1时的作用]	R	WOT	0x1	None
13	12:0	Reserved	保留位	R	W	0x0	None



Name	TEM
Address Offset	0X00000000
Width	32
Abbreviation	TEM
Description	Template Register

Bit	**Field Name**	**Description**	**Read Type**	**Write Type**	**Reset Value**	**Sync. Reset Signal**
31:18	Reserved	Reserved	R	W	0x0	None
17:17	FIELD_1	[Functionality description] [0: optional explanation for value 0 1: optional explanation for value 1]	RCLR	NA	0x0	srst_10, srst_11
16:14	FIELD_2	[Functionality description] [0: optional explanation for value 0 ... 7: optional explanation for value 7]	RSET	WOSET	0x0	srst_20
13:13	FIELD_3	[Functionality description] [0: optional explanation for value 0 1: optional explanation for value 1]	R	WOT	0x1	None
12:0	Reserved	Reserved	R	W	0x0	None

Designers can refer to this template generated by Template Generator, and edit to extend it, like arrange several tables corresponding to more than one registers in the worksheet in a way that a few blank lines separate each table.

Register elements are as follows.

- Register Name: consistent with the `name` attribute in SystemRDL. It is used to help understand register functionality which will be shown on HTML documents.
- Address Offset: each Excel worksheet is mapped to an `addrmap` component in SystemRDL and has a independent base address. Therefore, the address offset value filled in by the designer is based on the current worksheet's base address. It is recommended to start addressing from `0X0`.
- Register Bitwidth: currently only `32 bit` or `64 bit` is supported. If 32-bit bus interface is used to connected to the whole system, the snapshot feature will be implemented in 64-bit registers.
- Register Abbreviation: consistent with the register instance name in SystemRDL and in RTL modules.
- Register Description: consistent with the `desc` attribute in the SystemRDL. It is used to help understand register functionality which will be shown on HTML documents.
- Fields: define all fields including `Reserved`, listed in lines one by one.

    - Bit Range: indicates the location of the field in the form of `xx:xx`.
    - Field Name: corresponds to the field instance name of the generated RTL, also consistent with the `name` attribute in SystemRDL.
    - Field Description: consistent with the `desc` attribute in SystemRDL.
    - Read Attribute (Read Type): consistent with the `onread` attribute in SystemRDL. `R`, `RCLR` and `RSET` are supported.
    - Write Attribute (Write Type): consistent with the `onwrite` attribute in SystemRDL. `W`, `WOC`, `WOS`, `WOT`, `WZC`, `WZS`, `WZT` are supported.
    - Reset value: field reset value for synchronous and generic asynchronous reset signals.
    - Synchronous Reset Signals: In addition to the generic asynchronous reset by default, declaration of independent, one or more synchronous reset signals are supported.

Degisners should keep items mentioned above complete.

Rules

Follows are rules that designers should not violate when editing Excel worksheets.

- **BASIC_FORMAT :** Basic format constrained by regular expressions.
    1. the base address must be hexdecimal and prefixed with `0X(x)`
    2. the address offset must be hexdecimal and prefixed with `0X(x)`
    3. the register bitwidth can only be `32 bit` or `64 bit`.
    4. supported field read and write attributes: `R`, `RCLR`, `RSET`, `W`, `WOC`, `WOS`, `WOT`, `WZC`, `WZS`, `WZT`
    5. field bit range is in `xx:xx` format
    6. the reset value is hexdecimal and prefixed with `0X(x)`
    7. field synchronous reset signals is `None` if there is none, or there can be one or more, separated by `,` in the case of more than one

- **REG_ADDR :** Legality of the assignment of register address offsets.
    1. address offset is by integral times of the register byte length (called `regalign` method in SystemRDL)
    2. no address overlap is allowed in the same Excel worksheet
- **FIELD_DEFINITION :** Legality of field definitions.
    1. the bit order of multiple fields should be arranged from high to low
    2. the bit range of each field should be arranged in `[high_bit]:[low_bit]` order
    3. field bit range no overlap (3.1), and no omission (3.2)
    4. the reset value cannot exceed the maximum value which field can represent
    5. no duplicate field name except for `Reserved`
    6. the synchronous reset signal of the `Reserved` field should be `None`.
    7. no duplicate synchronous reset signal name in one field.

Tool Flow Guideline

Environment and dependencies

- Available OS: Windows/Linux
- Python Version 3.7+
    - systemrdl-compiler: https://github.com/SystemRDL/systemrdl-compiler
    - PeakRDL-html: https://github.com/SystemRDL/PeakRDL-html
    - PeakRDL-uvm: https://github.com/SystemRDL/PeakRDL-uvm

Command options and arguments

- `-h,--help`

Show help information.

- `-v, --version`

  Show `RDA Tool` version.

- `excel_template`

  Subcommand to generate register specification templates in Excel worksheet (.xlsx) format with the following command options.

  `-h, --help`

  Show help information for this subcommand.

  `-d,--dir [DIR]`

  Specify the location of the directory where the template will be generated, the default is the current directory.

  `-n,--name [NAME]`

  Specify the file name of the generated template, if there is a duplicate name, it will be automatically distinguished by a number, the default is `template.xlsx`.

  `-rnum [RNUM]`

  Specify the number of registers to be included in the generated template, default is `1`.

  `-rname [TEM1 TEM2 ...]`

  Specify the name of the register in the generated template, the default is `TEM`, the default name and abbreviation are the same.

  `-l, --language [cn | en]`

  Specify the language format of the generated template: `cn/en`, default is `cn`.

- `parse`

  Sub-command to check the syntax and rules of the input Excel(.xlsx) and SystemRDL(.rdl) files, and compile them into the hierarchical model defined in `systemrdl-compiler`, with the following command options.

  `-h, --help`

Show help information for this subcommand.

`-f, --file [FILE1 FILE2 ...]`

Specify the input Excel(.xlsx)/SystemRDL(.rdl) files, support multiple, mixed input files at the same time, error will be reported if any of input files do not exist.

`-l, --list [LIST_FILE]`

Specify a text-based file list including all files to be read. Parser will read and parse files in order, if the file list or any file in it does not exist, an error will be reported.

> The `-f, --file` or `-l, --list` options must be used but not at the same time. If so, warning message will be reported and parser will ignore the `-l, --list` option.

`-g, --generate`

Explicitly specifying this option parses and converts all input Excel (.xlsx) files to SystemRDL (.rdl) files one by one, with separate `addrmap` for each Excel worksheet. When the input is all Excel (.xlsx) files, parser generates an additional SystemRDL (.rdl) file containing the top-level `addrmap`, which instantiates all child `addrmaps`.

If this option is not used, Parser will only conduct rule check and parse, thus no additional files will be generated.

`-m, --module [MODULE_NAME]`

If `-g, --generate` option is specified, this option specifies top-level `addrmap` name and top-level RDL file name to be generated for subsequent analysis and further modification. See detailed information in Chapter 2.4: RTL generator.

`-gdir, --gen_dir [GEN_DIR]`

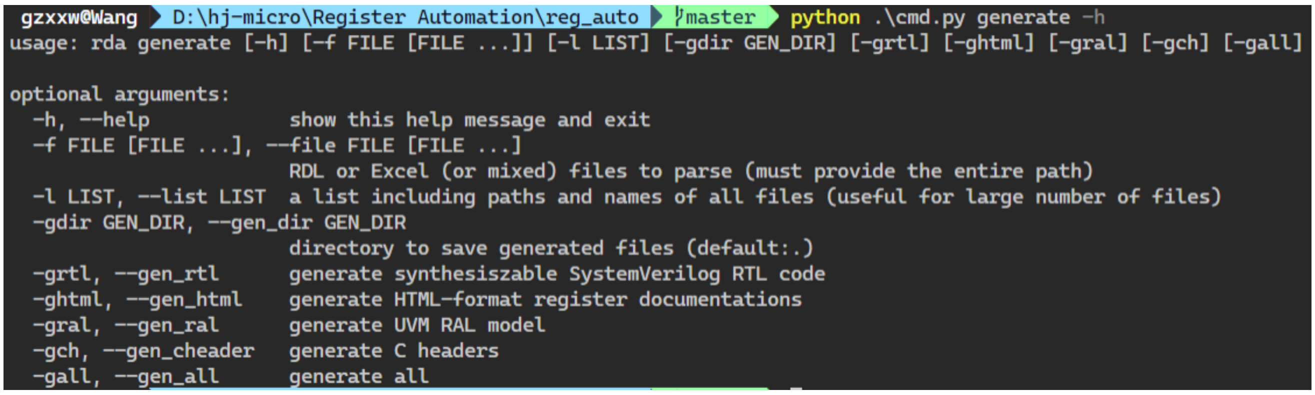
When using the `-g, --generate` option, this option specifies the directory where the files are generated, the default is the current directory.

- `generate`

  subcommand for generating RTL Module, HTML Docs, UVM RAL, C Header Files, with the following command options.

`-h, --help`

Show help information for this subcommand.



```
gzxxw@Wang  D:\hj-micro\Register Automation\reg_auto  master  python .\cmd.py generate -h
usage: rda generate [-h] [-f FILE [FILE ...]] [-l LIST] [-gdir GEN_DIR] [-grtl] [-ghtml] [-gral] [-gch] [-gall]

optional arguments:
  -h, --help            show this help message and exit
  -f FILE [FILE ...], --file FILE [FILE ...]
                        RDL or Excel (or mixed) files to parse (must provide the entire path)
  -l LIST, --list LIST  a list including paths and names of all files (useful for large number of files)
  -gdir GEN_DIR, --gen_dir GEN_DIR
                        directory to save generated files (default:.)
  -grtl, --gen_rtl      generate synthesiszable SystemVerilog RTL code
  -ghtml, --gen_html    generate HTML-format register documentations
  -gral, --gen_ral      generate UVM RAL model
  -gch, --gen_cheader   generate C headers
  -gall, --gen_all      generate all
```

`-f, --file [FILE1 FILE2 ...]`

Specify the input Excel(.xlsx)/SystemRDL(.rdl) files, support multiple, mixed input files at the same time, error will be reported if any of input files do not exist.

`-l, --list [LIST_FILE]`

Specify a text-based file list including all files to be read. Parser will read and parse files in order, if the file list or any file in it does not exist, an error will be reported.

> The `-f, --file` or `-l, --list` options must be used but not at the same time. If so, warning message will be reported and parser will ignore the `-l, --list` option.

`-m, --module [MODULE_NAME]`

Used for the situation where all input files are Excel worksheets. Like `-m` option in `parse` sub-command, this option specifies top-level `addrmap` name and top-level RDL file name to be generated for subsequent analysis and further modification. See detailed information in Chapter 2.4: RTL generator.

`-gdir, --gen_dir [dir]`

Specify the directory where the generated files will be stored. If the directory does not exist, an error will be reported. Default is the current directory.

`-grtl, --gen_rtl`

Specify this option explicitly to generate RTL Module code.

`-ghtml, --gen_html`

Specify this option explicitly to generate the register description in HTML format.

`-gral, --gen_ral`

Specify this option explicitly to generate the UVM RAL verification model.

`-gch,--gen_cheader`

Specifying this option explicitly generates the register C header file.

`-gall,--gen_all`

Specifying this option explicitly generates all of the above files.

Example

> Switch to the source directory of the tool, or add the executable `hrda` to `PATH`, or use the `module` tool for configuration.

- Generate the register template in Excel format.

```
mkdir test
hrda excel_template -n test.xlsx -rnum 3 -rname tem1 tem2 tem3
```

- Parse the register description in Excel format and generate the corresponding RDL file.

```
hrda parse -f test/test.xlsx -g -gdir . /test -m test_top
# another method: edit and save a list file
hrda parse -l test.list -g -gdir . /test -m test_top
```

- Generate using Generator

```
hrda generate -f test.xlsx -gdir . /test -grtl -ghtml -gral -gch
# another method: edit and save a list file
hrda generate -l test.list -gdir . /test -gall
```

> If execution of `hrda` fails, check that `hrda` is in `PATH` and that the Python version and dependencies mentioned in section 3.1 are satisfied.

Miscellaneous