SpyDrNet Reference

Release 1.8.0

BYU Configurable Computing Lab

Apr 23, 2021

CONTENTS

Release 1.8.0

Date Apr 23, 2021

CHAPTER

ONE

INTRODUCTION

Welcome to SpyDrNet, a tool that will help you analyze and transform netlists. SpyDrNet is developed and maintained by the Configurable Computing Lab of Brigham Young University. This tool is related to the BYU EDIF Tools and is considered to be the next generation tool for FPGA netlist analysis and transformation.

1.1 SpyDrNet Getting Started

Installation

This package will be available on Python Package Index shortly. Once it is, the stable release of SpyDrNet can be installed using `pip`:

```
> pip install spydrnet
```

To install from PyPI with all optional dependicies use:

```
> pip install spydrnet[all]
```

SpyDrNet can also be installed from a source archive:

```
> pip install spydrnet-1.0.0.tar.gz
```

Or a built distribution:

```
> pip install spydrnet-1.0.0-py3-none-any.whl
```

If a development environment is desired, the project can be installed in editable mode from the project directory:

```
> pip install -e .
```

Tool Flow

Netlists flow through SpyDrNet in a three step process (see Fig. 1.1). First, they are parsed by a *parser* into an intermediate representation (IR). Second, their IR is analyzed and transformed. Finally, their IR is composed by a *composer* back into a netlist format that a 3rd-party tool can use. This flow is inspired by LLVM and Pandoc. LLVM has a similar flow for compiling computer programs and Pandoc has a similar flow for converting document formats. Using this flow, SpyDrNet is designed to be able to work on any netlist.

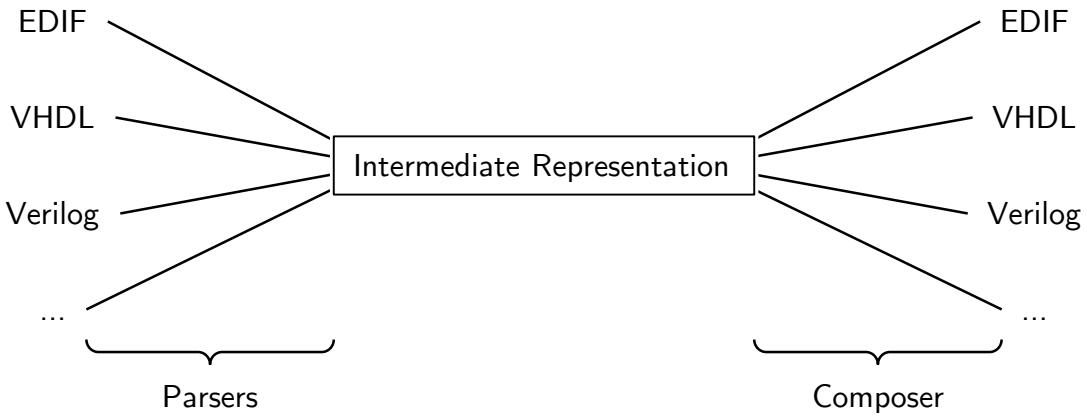

Fig. 1.1: Flow

SpyDrNet can be used to create a netlist from scratch. Thus, the API can be used to implement a parser and composer for arbitrary formats.

Parsing a netlist

SpyDrNet currently includes a parser for EDIF:

```
>>> netlist = sdn.parse('<netlist_filename>.edf')
```

Composing a netlist

SpyDrNet currently includes a composer for EDIF:

```
>>> sdn.compose('<filename>.edf', netlist)
```

Loading SpyDrNet

To load `spydrnet`, import it into a Python interactive interpreter or code:

```
>>> import spydrnet as sdn
>>>
```

At this point, SpyDrNet features and functionality are accessible via `sdn.<function/feature>`. The abbreviation of `sdn` is used throughout code examples to reference the `spydrnet` package.

Intermediate Representation Basics

Digital designs for FPGAs are represented as netlists, a list of components and connections. Netlists come from various vendors in many different formats. SpyDrNet allows you to look at and alter a netlist in a language inspecific way. SpyDrNet parses a netlist into an intermediate represention (IR) that is designed to be easily traversed and effortlessly manipulated. SpyDrNet aims to provide the tools you need to accomplish the netlist analysis and transformation tasks you have in mind without having to reinvent the wheel. Fig. 1.2 shows a summary of the SpyDrNet intermediate representation (IR).

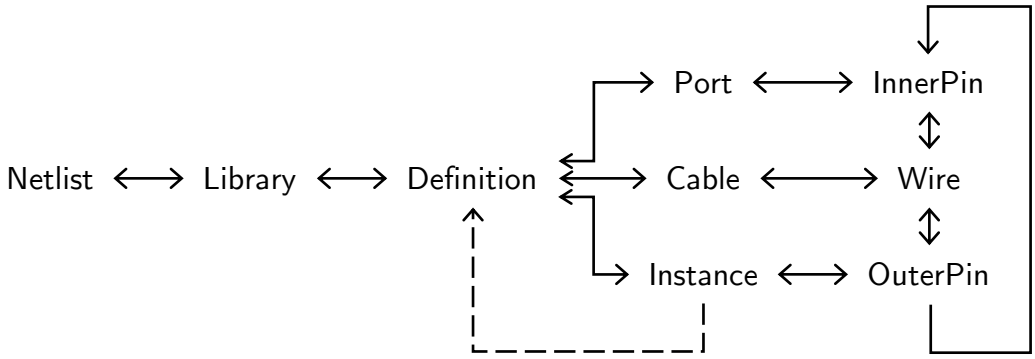

Fig. 1.2: Intermediate Representaion

SpyDrNet's intermediate representation of netlists (IR) is what sets it apart for other EDA tools. The IR is structured to house netlists in a generic way while allowing for format specific constructs to be preserved.

`Element` Most IR classes inherit from this Python class. Objects of this class are referred to as a netlist elements. A netlist element contains a dictionary for storing data specific to itself. This is accomplished using Python get/set item functions, (see *Element Data*).

`Netlist` This class of Python objects is the netlist element with the highest level of organization (a whole netlist). It contains an ordered collection of libraries and any data associated with the netlist as a whole.

`Library` This netlist element contains an ordered collection of cell or module definitions associated with a library.

`Definition` A definition outlines the contents of each component that can be instantiated elsewhere in the design. It holds information that is pertinant to all instances of itself including subcomponents ports and connections

`Instance` This element holds pointers to the definition which it instances, and contains its own set of pins to be connected to within its parent definition.

`Bundle` The Bundle class is a parent class of Ports and Cables because each can be thought of as an array. This class defines the structure that helps us properly represent array objects in netlists including the width, direction (to or downto) and starting index. As a parent class this class is not directly instantiated in netlist.

`Port` The Port element inherits from Bundles and can be thought of as containing the information on how a Definition connects the outside world to the elements it contains.

`Cable` Cables are bundles of connectors between components within a definition. They connect ports to their destination pins

`Pin` The pin class is also a parent class, inherited from by the inner pin and outer pin objects. Unlike the Element and Bundle objects, Pins are useful because they can hide some of the implementation details of the underlying inner pins and outer pins.

`InnerPin` These pins are collected in Ports and are contained on the inside of the definitions. There is one set of inner pins per definition but they could refer to several sets of OuterPins

`OuterPin` These pins are collected on instances. They let us distinguish between connections to multiple instances of a single definition. These objects remove the need to carefuly track hierarcy while navegating a netlist.

`Wire` Wires are grouped inside cables and are elements that help hold connection information between single pins on instances within a definition and within it's ports.

More detail on the IR is provided in *API Summary*.

1.2 Aracnid Etymology

Spiders are masters at spinning webs. These webs often created like nets are stronger than steel when stretched and much more elastic. SpyDrNet aims to give end users the ability to pass these traits on to their netlists by enabling reliability and other applications through generic analysis and transformations on netlist. Of course this is just scratching the surface of the ways in which this name is applicable to the tool. Finding these fun meanings is (as it is said in academia) left as an exercise to the curious reader. For now we would rather discuss what this tool can be used to do.

1.3 Other Information

SpyDrNet is part of a rising ecosystem of free and open source software (FOSS) for FPGA development. Consider MyHDL, PyEDA, Yosys, LiveHD, ABC, BLIF, RapidWright, RapidSmith, RapidSmith2, JHDL, BYU EDIF Tools, VQM, and Project X-ray.

CHAPTER

TWO

ELEMENT DATA

Each netlist element allows for arbitrary data to be associated with it. This is accomplished using a Python `dict`.

2.1 Keys

Keys are ought to be strings with `.` seperated namespaces. Element properties from the originating netlist format belong in the root namespace (without any `.`). The NULL namespace (keys with a leading `.`) is reserved for use by SpyDrNet. For example:

```
>>> element['.NAME'] = "name_of_element"
```

The `'.NAME'` key in this example is the key `NAME` in the NULL namespace. The key is reserved for the reference name of elements.

Language specific constructs can be sored under key entries within the namespace of the specific language. For example:

```
>>> netlist[EDIF.edifVersion] = (2, 0, 0)
```

The key value par stores the EDIF version used by an EDIF netlist.

2.2 Setting Data

Data is set using the Python `__setitem__` magic function meaning that data can be set using this syntax:

```
>>> element['<key>'] = value
```

2.3 Getting Data

Data can be read through itteration:

```
>>> for key in element:
>>>     print(key, element[key])
>>>
```

Or by using the using the Python `__getitem__` magic function by itself:

```
>>> print(element['<key>'])
value
```

A read only view of the data dictionary can be obtained from `element.data`. The returned object acts and feels like a Python dictionary, but mutator functions are disabled. This allows for the automated managment of the dictionary.

2.4 Deleting Data

Entries in the dictionary can be deleted using the *__delitem__* magic function as follows:

```
>>> del element['<key>']
```

CHAPTER

THREE

API SUMMARY

The SpyDrNet API can be used to create, analyze, and transform a netlist. Netlist are represented in memory in an Intermediate Representation. Fig. 3.1 show the representation of a simple circuit in the SpyDrNet Intermediate Representation. If you would like an example of using the SpyDrNet tool to create a netlist like this, click here

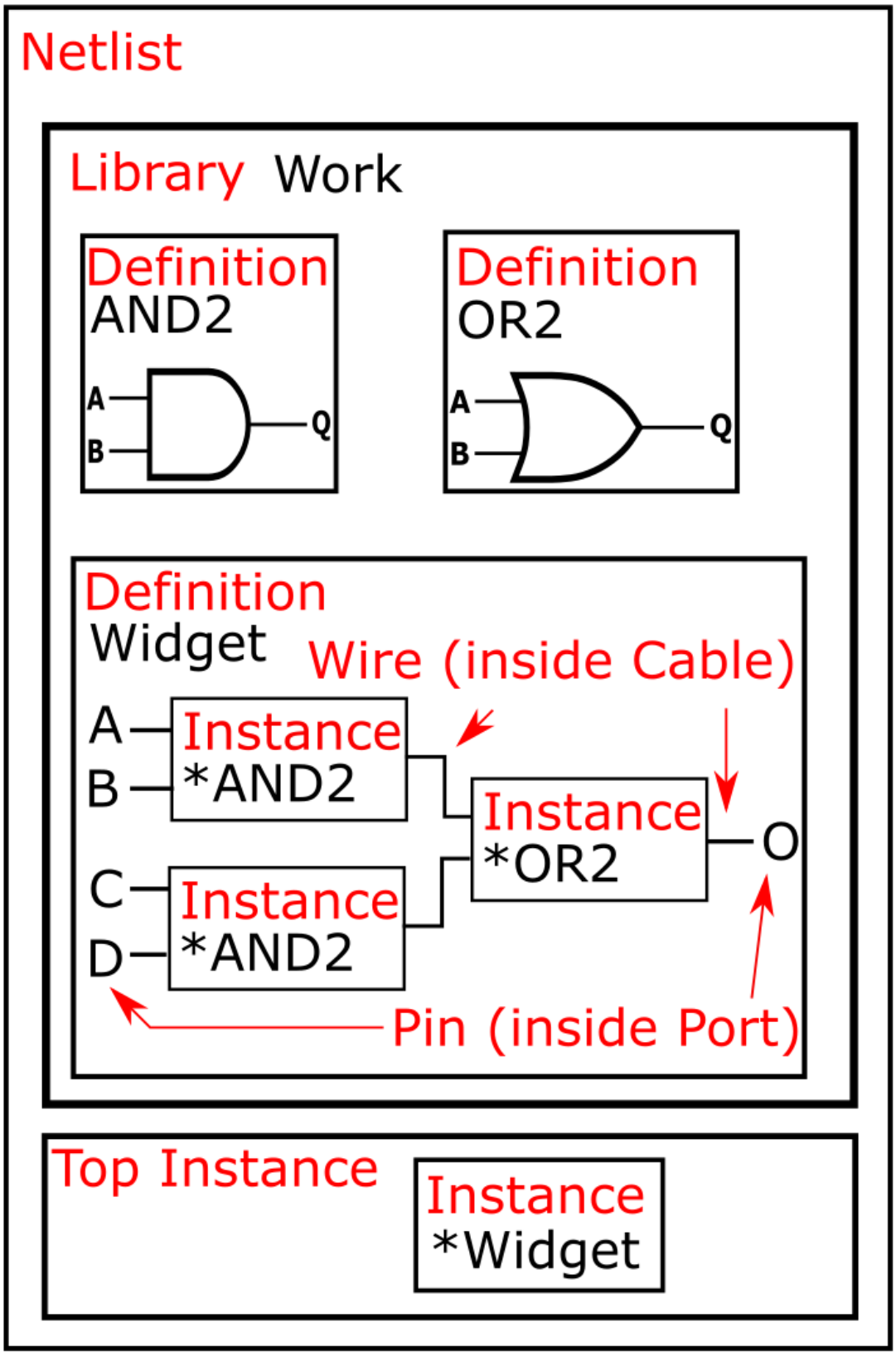

Fig. 3.1: Example Netlist in the Intermediate Representaion

The API calls documented here can be used in Python as follows:

```
>>> # create an empty netlist and add an empty library to it
>>> import spydrnet as sdn
>>> netlist = sdn.ir.Netlist()
>>> library = netlist.create_library()
>>>
```

Similarly if the parser is used the calls can be made in the same way:

```
>>> # parse an edif file in and add an empty library to the netlist.
>>> import spydrnet as sdn
>>> netlist = sdn.parse('four_bit_counter.edf')
```

```
>>> library = netlist.create_library
>>>
```

3.1 Basic object types

3.1.1 Netlist

Overview

class spydrnet.**Netlist**(*name=None*, *properties=None*)

Represents a netlist object.

Contains a top level instance and libraries

Examples

After importing the spydrnet package, we can initialize the netlist from scratch

```
>>> import spydrnet as sdn
>>> netlist = sdn.Netlist()
```

We can also initalize an top instance for the netlist. For more info: *Instance*

```
>>> top_instance = sdn.Instance()
>>> netlist.top_instance = top_instance
```

This is the way to set up a top level definition for the netlist

```
>>> top_definition = sdn.Definition()
>>> netlist.top_instance = top_definition
```

We can initialize a “primatives” and a “work” library this way:

```
>>> primitives_library = netlist.create_library()
>>> primitives_library['EDIF.identifier'] = 'hdi_primitives'
```

```
>>> work_library = netlist.create_library()
>>> work_library['EDIF.identifier'] = 'work'
```

Methods

Netlist.__init__([name, properties])	creates an empty object of type netlist
Netlist.compose(*args, **kwargs)	Compose a netlist into a file format.
Netlist.libraries	Get a list of all libraries included in the netlist.
Netlist.top_instance	Get the top instance in the netlist.
Netlist.create_library([name, properties])	Create a library and add it to the netlist and return that library
Netlist.add_library(library[, position])	add an already existing library to the netlist.
Netlist.remove_library(library)	Removes the given library if it is in the netlist
Netlist.remove_libraries_from(libraries)	Removes all the given libraries from the netlist.
Netlist.clone()	API safe clone on a netlist.
Netlist.get_netlists(...)	Shortcut to *get_netlists()*.
Netlist.get_libraries(...)	Shortcut to *get_libraries()*.
Netlist.get_definitions(...)	Shortcut to *get_definitions()*.
Netlist.get_instances(...)	Shortcut to *get_instances()*.
Netlist.get_ports(...)	Shortcut to *get_ports()*.
Netlist.get_pins(...)	Shortcut to *get_pins()*.
Netlist.get_cables(...)	Shortcut to *get_cables()*.
Netlist.get_wires(...)	Shortcut to *get_wires()*.
Netlist.get_hinstances(...)	Shortcut to *get_hinstances()*.
Netlist.get_hports(...)	Shortcut to *get_hports()*.
Netlist.get_hpins(...)	Shortcut to *get_hpins()*.
Netlist.get_hcables(...)	Shortcut to *get_hcables()*.
Netlist.get_hwires(...)	Shortcut to *get_hwires()*.

spydrnet.Netlist.__init__

Netlist.**__init__**(*name=None*, *properties=None*)

creates an empty object of type netlist

Parameters

- **name - (str) the name of this instance**
- **properties - (dict) the dictionary which holds the properties**

spydrnet.Netlist.compose

Netlist.**compose**(**args*, ***kwargs*)
Compose a netlist into a file format.

Compose(filename). Shortcut to *compose()*.

spydrnet.Netlist.libraries

property Netlist.**libraries**
Get a list of all libraries included in the netlist.

spydrnet.Netlist.top_instance

property Netlist.**top_instance**
Get the top instance in the netlist.

Returns The top level instance in the environment

Return type *Instance*

spydrnet.Netlist.create_library

Netlist.**create_library**(*name=None*, *properties=None*)
Create a library and add it to the netlist and return that library

Parameters

- **name - (str) the name of the library**
- **properties - (dict) the dictionary which holds the properties of the library**

spydrnet.Netlist.add_library

Netlist.**add_library**(*library*, *position=None*)
add an already existing library to the netlist.

This library should not belong to another netlist. Use remove_library from other netlists before adding

Parameters

- **library - Library** – The library to be added to the netlist.
- **position - int, (default None)** – When set, it is the index at which to add the library in the libraries list.

spydrnet.Netlist.remove_library

Netlist.**remove_library**(*library*)
Removes the given library if it is in the netlist

Parameters **library - Library** – The library to be removed.

spydrnet.Netlist.remove_libraries_from

Netlist.**remove_libraries_from**(*libraries*)
Removes all the given libraries from the netlist.

All libraries must be in the netlist.

Parameters **libraries - Set** – Libraries to be removed.

spydrnet.Netlist.clone

Netlist.**clone**()
API safe clone on a netlist.

This clone function should act just the way you would expect All references are internal to the netlist that has been cloned.

spydrnet.Netlist.get_netlists

Netlist.**get_netlists**(...)
Shortcut to *get_netlists()*.

spydrnet.Netlist.get_libraries

Netlist.**get_libraries**(...)
Shortcut to *get_libraries()*.

spydrnet.Netlist.get_definitions

Netlist.**get_definitions**(...)
Shortcut to *get_definitions()*.

spydrnet.Netlist.get_instances

Netlist.**get_instances**(...)
Shortcut to *get_instances()*.

spydrnet.Netlist.get_ports

Netlist.**get_ports**(...)
Shortcut to *get_ports()*.

spydrnet.Netlist.get_pins

Netlist.**get_pins**(...)
Shortcut to *get_pins()*.

spydrnet.Netlist.get_cables

Netlist.**get_cables**(...)
Shortcut to *get_cables()*.

spydrnet.Netlist.get_wires

Netlist.**get_wires**(...)
Shortcut to *get_wires()*.

spydrnet.Netlist.get_hinstances

Netlist.**get_hinstances**(...)
Shortcut to *get_hinstances()*.

spydrnet.Netlist.get_hports

Netlist.**get_hports**(...)
Shortcut to *get_hports()*.

spydrnet.Netlist.get_hpins

Netlist.**get_hpins**(...)
Shortcut to *get_hpins()*.

spydrnet.Netlist.get_hcables

Netlist.**get_hcables**(...)
Shortcut to *get_hcables()*.

spydrnet.Netlist.get_hwires

Netlist.**get_hwires**(...)
Shortcut to *get_hwires()*.

3.1.2 Library

Overview

class spydrnet.**Library**(*name=None*, *properties=None*)
Represents a library object.

Contains a pointer to parent netlist and definitions.

Methods

Library.__init__([name, properties])	creates an empty object of type Library
Library.netlist	Get the netlist that contains this library
Library.definitions	Return a list of all the definitions that are included in this library
Library.create_definition([name, properties])	Create a definition, add it to the library, and return the definition
Library.add_definition(definition[, position])	Add an existing definition to the library.
Library.remove_definition(definition)	Remove the given definition from the library.
Library.remove_definitions_from(definitions)	Remove a set of definitions from the library.
Library.clone()	Clone the library in an API safe manner.
Library.get_netlists(...)	Shortcut to *get_netlists()*.
Library.get_libraries(...)	Shortcut to *get_libraries()*.
Library.get_definitions(...)	Shortcut to *get_definitions()*.
Library.get_instances(...)	Shortcut to *get_instances()*.
Library.get_ports(...)	Shortcut to *get_ports()*.
Library.get_pins(...)	Shortcut to *get_pins()*.
Library.get_cables(...)	Shortcut to *get_cables()*.
Library.get_wires(...)	Shortcut to *get_wires()*.
Library.get_hinstances(...)	Shortcut to *get_hinstances()*.
Library.get_hports(...)	Shortcut to *get_hports()*.
Library.get_hpins(...)	Shortcut to *get_hpins()*.
Library.get_hcables(...)	Shortcut to *get_hcables()*.
Library.get_hwires(...)	Shortcut to *get_hwires()*.

spydrnet.Library.__init__

Library.**__init__**(*name=None*, *properties=None*)
creates an empty object of type Library

Parameters

- **name - (str) the name of this instance**
- **properties - (dict) the dictionary which holds the properties**

spydrnet.Library.netlist

property Library.**netlist**
Get the netlist that contains this library

spydrnet.Library.definitions

property Library.**definitions**
Return a list of all the definitions that are included in this library

spydrnet.Library.create_definition

Library.**create_definition**(*name=None*, *properties=None*)
Create a definition, add it to the library, and return the definition

Parameters

- **name - (str) the name of this instance**
- **properties - (dict) the dictionary which holds the properties**

spydrnet.Library.add_definition

Library.**add_definition**(*definition*, *position=None*)
Add an existing definition to the library.

The definition must not belong to a library including this one.

Parameters

- **definition - Definition** – The defintion to add to the library
- **position - int, (default None)** – the index in the library list at which to add the definition

spydrnet.Library.remove_definition

Library.**remove_definition**(*definition*)
Remove the given definition from the library.

Parameters **definition - Definition** – The definition to be removed.

spydrnet.Library.remove_definitions_from

Library.**remove_definitions_from**(*definitions*)
Remove a set of definitions from the library.

All definitions provided must be in the library.

Parameters **Definitions - Set of Definition type objects** – The definitions to be removed

spydrnet.Library.clone

Library.**clone**()
Clone the library in an API safe manner.

The following describes the structure of the returned object:

- the instances that pointed to reference definitions within the library will have updated references
- the instances that pointed to reference definitions outside the library will maintain their definitions
- the references lists (of definitions) both inside and outsde the library will be updated to reflect the change
- all definitions are cloned within the library.

spydrnet.Library.get_netlists

Library.**get_netlists**(...)
Shortcut to *get_netlists()*.

spydrnet.Library.get_libraries

Library.**get_libraries**(...)
Shortcut to *get_libraries()*.

spydrnet.Library.get_definitions

Library.**get_definitions**(...)
Shortcut to *get_definitions()*.

spydrnet.Library.get_instances

Library.**get_instances**(...)
Shortcut to *get_instances()*.

spydrnet.Library.get_ports

Library.**get_ports**(...)
Shortcut to *get_ports()*.

spydrnet.Library.get_pins

Library.**get_pins**(...)
Shortcut to *get_pins()*.

spydrnet.Library.get_cables

Library.**get_cables**(...)
Shortcut to *get_cables()*.

spydrnet.Library.get_wires

Library.**get_wires**(...)
Shortcut to *get_wires()*.

spydrnet.Library.get_hinstances

Library.**get_hinstances**(...)
Shortcut to *get_hinstances()*.

spydrnet.Library.get_hports

Library.**get_hports**(...)
Shortcut to *get_hports()*.

spydrnet.Library.get_hpins

Library.**get_hpins**(...)
Shortcut to *get_hpins()*.

spydrnet.Library.get_hcables

Library.**get_hcables**(...)
Shortcut to *get_hcables()*.

spydrnet.Library.get_hwires

Library.**get_hwires**(...)
Shortcut to *get_hwires()*.

3.1.3 Definition

Overview

class spydrnet.**Definition**(*name=None*, *properties=None*)
Represents a definition of a cell, module, entity/architecture, or paralleled structure object.

Contains a pointer to parent library, ports, cables, and instances.

Methods

Definition.__init__([name, properties])	Creates an empty object of type definition
Definition.is_leaf()	Check to see if this definition represents a leaf cell.

spydrnet.Definition.__init__

Definition.**__init__**(*name=None*, *properties=None*)

Creates an empty object of type definition

Parameters

- **name - (str) the name of this instance**
- **properties - (dict) the dictionary which holds the properties**

spydrnet.Definition.is_leaf

Definition.**is_leaf**()

Check to see if this definition represents a leaf cell.

Leaf cells are cells with no children instances or no children cables. Blackbox cells are considered leaf cells as well as direct pass through cells with cables only

Getter Functions

Definition.__init__([name, properties])	Creates an empty object of type definition
Definition.get_netlists(...)	Shortcut to *get_netlists()*.
Definition.library	Get the library that contains this definition
Definition.ports	Get the ports that are instanced in this definition
Definition.cables	Get the cables that are instanced in this definition.
Definition.children	Return a list of all instances instantiated in this definition
Definition.references	Get a list of all the instances of this definition
Definition.is_leaf()	Check to see if this definition represents a leaf cell.
Definition.create_port([name, properties, ...])	Create a port, add it to the definition, and return that port.
Definition.add_port(port[, position])	Add a preexisting port to the definition.
Definition.remove_port(port)	Remove a port from the definition.
Definition.remove_ports_from(ports)	Remove a set of ports from the definition.
Definition.create_child([name, properties, ...])	Create an instance to add to the definition, add it, and return the instance.
Definition.add_child(instance[, position])	Add an existing instance to the definition.
Definition.remove_child(child)	Remove an instance from the definition.
Definition.remove_children_from(children)	Remove a set of instances from the definition.
Definition.create_cable([name, properties, ...])	Create a cable, add it to the definition, and return the cable.
Definition.add_cable(cable[, position])	Add a cable to the definition.
Definition.remove_cable(cable)	Remove a cable from the definition.

continues on next page

Table 3.4 – continued from previous page

Definition.remove_cables_from(cables)	Remove a set of cables from the definition.
Definition.clone()	Clone the definition in an api safe way.
Definition.get_netlists(...)	Shortcut to *get_netlists()*.
Definition.get_libraries(...)	Shortcut to *get_libraries()*.
Definition.get_definitions(...)	Shortcut to *get_definitions()*.
Definition.get_instances(...)	Shortcut to *get_instances()*.
Definition.get_ports(...)	Shortcut to *get_ports()*.
Definition.get_pins(...)	Shortcut to *get_pins()*.
Definition.get_cables(...)	Shortcut to *get_cables()*.
Definition.get_wires(...)	Shortcut to *get_wires()*.
Definition.get_hinstances(...)	Shortcut to *get_hinstances()*.
Definition.get_hports(...)	Shortcut to *get_hports()*.
Definition.get_hpins(...)	Shortcut to *get_hpins()*.
Definition.get_hcables(...)	Shortcut to *get_hcables()*.
Definition.get_hwires(...)	Shortcut to *get_hwires()*.

spydrnet.Definition.get_netlists

Definition.**get_netlists**(...)
Shortcut to *get_netlists()*.

spydrnet.Definition.library

property Definition.**library**
Get the library that contains this definition

spydrnet.Definition.ports

property Definition.**ports**
Get the ports that are instanced in this definition

spydrnet.Definition.cables

property Definition.**cables**
Get the cables that are instanced in this definition.

spydrnet.Definition.children

property Definition.**children**
Return a list of all instances instantiated in this definition

spydrnet.Definition.references

property Definition.**references**
Get a list of all the instances of this definition

spydrnet.Definition.create_port

Definition.**create_port**(*name=None*, *properties=None*, *is_downto=None*, *is_scalar=None*, *lower_index=None*, *direction=None*)
Create a port, add it to the definition, and return that port.

Parameters

- **name - (str) the name of this instance**
- **properties - (dict) the dictionary which holds the properties**
- **id_downto - (bool) set the downto status. Downto is False if the right index is higher than the left one, True otherwise**
- **is_scalar - (bool) set the scalar status. Return True if the item is a scalar False otherwise.**
- **lower_index - (int) get the value of the lower index of the array.**
- **direction - (Enum) Define the possible directions for a given port. (UNDEFINED, INOUT, IN, OUT)**

spydrnet.Definition.add_port

Definition.**add_port**(*port*, *position=None*)
Add a preexisting port to the definition.

This port must not be a member of any definition

Parameters

- **port - (Port) the port to add to the definition**
- **position - (int, default None) the index in the port list at which to add the port**

spydrnet.Definition.remove_port

Definition.**remove_port**(*port*)
Remove a port from the definition.

his port must be a member of the definition in order to be removed

Parameters port - (Port) the port to be removed

spydrnet.Definition.remove_ports_from

Definition.**remove_ports_from**(*ports*)

Remove a set of ports from the definition.

All these ports must be included in the definition

Parameters ports - (Set containing Port type objects) the ports to remove from the definition

spydrnet.Definition.create_child

Definition.**create_child**(*name=None*, *properties=None*, *reference=None*)

Create an instance to add to the definition, add it, and return the instance.

This functions calls the add_child funciton.

Parameters

- **name - (str) the name of this instance**
- **properties - (dict) the dictionary which holds the properties**

Example

To create a child:

```
>>> definition = sdn.Definition()
>>> child_instance = definition.create_child()
>>> child_instance.name = "child_instance"
>>> child_instance.reference = reference_definition
```

To create a child with optional parameters

```
>>> child_instance = definition.create_child(name="child_instance",␣
↪reference=reference_definition)
```

The reference of the instance is the definition that initialized this instance.

spydrnet.Definition.add_child

Definition.**add_child**(*instance*, *position=None*)

Add an existing instance to the definition.

This instance must not already be included in a definition. It add the instance as a child into the given position. Append to the end of the list if no position is given. It will set the parent of the instance to this definition.

Parameters

- **instance - (Instance) the instance to add as a child of the definition**
- **position - (int, default None) the index in the children list at which to add the instance.**

spydrnet.Definition.remove_child

Definition.**remove_child**(*child*)
Remove an instance from the definition.

The instance must be a member of the definition already

Parameters **instance - (Instance) the instance to be removed from the definition**

spydrnet.Definition.remove_children_from

Definition.**remove_children_from**(*children*)
Remove a set of instances from the definition.

All instances must be members of the definition

Parameters **children - (Set of Instance type objects) the children to be removed from the definition**

spydrnet.Definition.create_cable

Definition.**create_cable**(*name=None*, *properties=None*, *is_downto=None*, *is_scalar=None*, *lower_index=None*)
Create a cable, add it to the definition, and return the cable.

Parameters

- **name - (str) the name of this instance**
- **properties - (dict) the dictionary which holds the properties**
- **id_downto - (bool) set the downto status. Downto is False if the right index is higher than the left one, True otherwise**
- **is_scalar - (bool) set the scalar status. Return True if the item is a scalar False otherwise.**
- **lower_index - (int) get the value of the lower index of the array.**

spydrnet.Definition.add_cable

Definition.**add_cable**(*cable*, *position=None*)
Add a cable to the definition.

The cable must not already be a member of another definition.

Parameters

- **cable - (Cable) the cable to be added**
- **position - (int, default None) the position in the cable list at which to add the cable**

spydrnet.Definition.remove_cable

Definition.**remove_cable**(*cable*)
Remove a cable from the definition.

The cable must be a member of the definition.

Parameters cable - (Cable) the cable to be removed from the definition

spydrnet.Definition.remove_cables_from

Definition.**remove_cables_from**(*cables*)
Remove a set of cables from the definition.

The cables must be members of the definition

Parameters cables - (Set of Cable type objects) the cables to be remove from the definition

spydrnet.Definition.clone

Definition.**clone**()
Clone the definition in an api safe way.

The cloned object will have the following properties

- the definition will be orphaned and will not belong to any library
- each of the sub elements of the definition will also be cloned and the connection structure between them will be updated.
- the cloned instances will still point to the reference to which the pointed before. They will also be members of the references list of those definitions.

spydrnet.Definition.get_libraries

Definition.**get_libraries**(...)
Shortcut to *get_libraries()*.

spydrnet.Definition.get_definitions

Definition.**get_definitions**(...)
Shortcut to *get_definitions()*.

spydrnet.Definition.get_instances

Definition.**get_instances**(...)
Shortcut to *get_instances()*.

spydrnet.Definition.get_ports

Definition.**get_ports**(...)
Shortcut to *get_ports()*.

spydrnet.Definition.get_pins

Definition.**get_pins**(...)
Shortcut to *get_pins()*.

spydrnet.Definition.get_cables

Definition.**get_cables**(...)
Shortcut to *get_cables()*.

spydrnet.Definition.get_wires

Definition.**get_wires**(...)
Shortcut to *get_wires()*.

spydrnet.Definition.get_hinstances

Definition.**get_hinstances**(...)
Shortcut to *get_hinstances()*.

spydrnet.Definition.get_hports

Definition.**get_hports**(...)
Shortcut to *get_hports()*.

spydrnet.Definition.get_hpins

Definition.**get_hpins**(...)
Shortcut to *get_hpins()*.

spydrnet.Definition.get_hcables

Definition.**get_hcables**(...)
Shortcut to *get_hcables()*.

spydrnet.Definition.get_hwires

Definition.**get_hwires**(...)
Shortcut to *get_hwires()*.

3.1.4 Instance

Overview

class spydrnet.**Instance**(*name=None*, *properties=None*)
Netlist instance of a netlist definition.

Instances are literally instances of definitions and they reside inside definitions. Function names have been set to adjust for the potential confusion that could arise because instances both have a parent definition and have definitions which they reference.

Variables

- ***parent*** – the parent of the object. Parent is the definition of the instance that contains the current instance.
- ***reference*** – the item of the object. Reference is the definition of the instance that instantiated or defined the current instance.

Methods

Instance.__init__([name, properties])	Creates an empty object of type instance.
Instance.parent	Get the definition that contains this instance
Instance.reference	Get the definition that this instance is instantiating
Instance.get_ports(...)	Shortcut to *get_ports()*.
Instance.pins	Get the pins on this instance.
Instance.clone()	Clone the instance in an api safe way.
Instance.get_netlists(...)	Shortcut to *get_netlists()*.
Instance.get_libraries(...)	Shortcut to *get_libraries()*.
Instance.get_definitions(...)	Shortcut to *get_definitions()*.
Instance.get_instances(...)	Shortcut to *get_instances()*.
Instance.get_ports(...)	Shortcut to *get_ports()*.
Instance.get_pins(...)	Shortcut to *get_pins()*.
Instance.get_cables(...)	Shortcut to *get_cables()*.
Instance.get_wires(...)	Shortcut to *get_wires()*.
Instance.get_hinstances(...)	Shortcut to *get_hinstances()*.
Instance.get_hports(...)	Shortcut to *get_hports()*.
Instance.get_hpins(...)	Shortcut to *get_hpins()*.
Instance.get_hcables(...)	Shortcut to *get_hcables()*.
Instance.get_hwires(...)	Shortcut to *get_hwires()*.

spydrnet.Instance.__init__

Instance.**__init__**(*name=None*, *properties=None*)
Creates an empty object of type instance.

Parameters

- **name - (str) the name of this instance**
- **properties - (dict) the dictionary which holds the properties**

spydrnet.Instance.parent

property Instance.**parent**
Get the definition that contains this instance

spydrnet.Instance.reference

property Instance.**reference**
Get the definition that this instance is instantiating

spydrnet.Instance.get_ports

Instance.**get_ports**(...)
Shortcut to *get_ports()*.

spydrnet.Instance.pins

property Instance.**pins**
Get the pins on this instance.

spydrnet.Instance.clone

Instance.**clone**()
Clone the instance in an api safe way. This call will return a cloned instance that has the following properties:

- the pins in the instance will all be disconnected from wires but they will maintain their references to inner pins
- the instance references is the same as the cloned object
- the reference's references list contains this instance
- the instance is orphaned (no longer a child of the definition to which the cloned definition belonged

spydrnet.Instance.get_netlists

Instance.**get_netlists**(...)
Shortcut to *get_netlists()*.

spydrnet.Instance.get_libraries

Instance.**get_libraries**(...)
Shortcut to *get_libraries()*.

spydrnet.Instance.get_definitions

Instance.**get_definitions**(...)
Shortcut to *get_definitions()*.

spydrnet.Instance.get_instances

Instance.**get_instances**(...)
Shortcut to *get_instances()*.

spydrnet.Instance.get_pins

Instance.**get_pins**(...)
Shortcut to *get_pins()*.

spydrnet.Instance.get_cables

Instance.**get_cables**(...)
Shortcut to *get_cables()*.

spydrnet.Instance.get_wires

Instance.**get_wires**(...)
Shortcut to *get_wires()*.

spydrnet.Instance.get_hinstances

Instance.**get_hinstances**(...)
Shortcut to *get_hinstances()*.

spydrnet.Instance.get_hports

Instance.**get_hports**(...)
Shortcut to *get_hports()*.

spydrnet.Instance.get_hpins

Instance.**get_hpins**(...)
Shortcut to *get_hpins()*.

spydrnet.Instance.get_hcables

Instance.**get_hcables**(...)
Shortcut to *get_hcables()*.

spydrnet.Instance.get_hwires

Instance.**get_hwires**(...)
Shortcut to *get_hwires()*.

3.1.5 Port

Overview

class spydrnet.**Port**(*name=None*, *properties=None*, *is_downto=None*, *is_scalar=None*, *lower_index=None*, *direction=None*)
Located on the inside of a definition.

Ports contain information about the quantity and directon of pins that go into and out of the defined struture when instanced.

Methods

Port.__init__([name, properties, is_downto, ...])	Setup an empty port
Port.direction	Gets the direction of the port.
Port.pins	Get a list of the pins that are in the port
Port.create_pin()	Create a pin and add it to the port.
Port.add_pin(pin[, position])	Add a pin to the port at the given position.
Port.remove_pin(pin)	Remove the given pin from the port.
Port.remove_pins_from(pins)	Remove several pins from the port at once.
Port.clone()	Clone the port in an api safe way.
Port.get_netlists(...)	Shortcut to *get_netlists()*.
Port.get_libraries(...)	Shortcut to *get_libraries()*.
Port.get_definitions(...)	Shortcut to *get_definitions()*.
Port.get_instances(...)	Shortcut to *get_instances()*.
Port.get_ports(...)	Shortcut to *get_ports()*.
Port.get_pins(...)	Shortcut to *get_pins()*.

continues on next page

Table 3.6 – continued from previous page

Port.get_cables(...)	Shortcut to *get_cables()*.
Port.get_wires(...)	Shortcut to *get_wires()*.
Port.get_hinstances(...)	Shortcut to *get_hinstances()*.
Port.get_hports(...)	Shortcut to *get_hports()*.
Port.get_hpins(...)	Shortcut to *get_hpins()*.
Port.get_hcables(...)	Shortcut to *get_hcables()*.
Port.get_hwires(...)	Shortcut to *get_hwires()*.

spydrnet.Port.__init__

Port.**__init__**(*name=None*, *properties=None*, *is_downto=None*, *is_scalar=None*, *lower_index=None*, *direction=None*)

Setup an empty port

Parameters

- **name - (str) the name of this instance**
- **properties - (dict) the dictionary which holds the properties**
- **id_downto - (bool) set the downto status. Downto is False if the right index is higher than the left one, True otherwise**
- **is_scalar - (bool) set the scalar status. Return True if the item is a scalar False otherwise.**
- **lower_index - (int) get the value of the lower index of the array.**
- **direction - (Enum) Define the possible directions for a given port. (UNDEFINED, INOUT, IN, OUT)**

spydrnet.Port.direction

property Port.**direction**

Gets the direction of the port.

This will be a variable of type Port.Direction

spydrnet.Port.pins

property Port.**pins**

Get a list of the pins that are in the port

spydrnet.Port.create_pin

Port.**create_pin**()

Create a pin and add it to the port.

return: the inner_pin created

spydrnet.Port.add_pin

Port.**add_pin**(*pin*, *position=None*)
Add a pin to the port at the given position.

Parameters

- **pin - (Pin) the pin to be added to the port.**
- **position - (int, default None) the index at which to add the pin**

spydrnet.Port.remove_pin

Port.**remove_pin**(*pin*)
Remove the given pin from the port.

The pin must belong to the port in order to be removed. Wires are disconnected from the pin that is removed.

Parameters pin - (Pin) a pin to be removed from the port.

spydrnet.Port.remove_pins_from

Port.**remove_pins_from**(*pins*)
Remove several pins from the port at once.

The wires are disconnected from the pins that are removed.

Parameters pins - (List of Pin objects) a list of all pins to be removed from the port.

spydrnet.Port.clone

Port.**clone**()
Clone the port in an api safe way.

The following rules will be observed:

- all the pins will be disconnected from wires
- the port will be orphaned
- all pins will belong to the returned port
- direction, downto, is_scalar, lower_index will all be maintained

spydrnet.Port.get_netlists

Port.**get_netlists**(...)
Shortcut to *get_netlists()*.

spydrnet.Port.get_libraries

Port.**get_libraries**(...)
Shortcut to *get_libraries()*.

spydrnet.Port.get_definitions

Port.**get_definitions**(...)
Shortcut to *get_definitions()*.

spydrnet.Port.get_instances

Port.**get_instances**(...)
Shortcut to *get_instances()*.

spydrnet.Port.get_ports

Port.**get_ports**(...)
Shortcut to *get_ports()*.

spydrnet.Port.get_pins

Port.**get_pins**(...)
Shortcut to *get_pins()*.

spydrnet.Port.get_cables

Port.**get_cables**(...)
Shortcut to *get_cables()*.

spydrnet.Port.get_wires

Port.**get_wires**(...)
Shortcut to *get_wires()*.

spydrnet.Port.get_hinstances

Port.**get_hinstances**(...)
Shortcut to *get_hinstances()*.

spydrnet.Port.get_hports

Port.**get_hports**(...)
Shortcut to *get_hports()*.

spydrnet.Port.get_hpins

Port.**get_hpins**(...)
Shortcut to *get_hpins()*.

spydrnet.Port.get_hcables

Port.**get_hcables**(...)
Shortcut to *get_hcables()*.

spydrnet.Port.get_hwires

Port.**get_hwires**(...)
Shortcut to *get_hwires()*.

3.1.6 InnerPin

Overview

class spydrnet.**InnerPin**
Pins that correspond to definitions.

These pins can be thought of as on the inside of a definition. There can be many outer pins for each inner pin

Methods

InnerPin.__init__()	Initialize self.
InnerPin.port	Return the port that the inner pin is a part of.
InnerPin.wire	Get the wire the pin is connected to.
InnerPin.clone()	Clone the inner pin in an api safe way.
InnerPin.get_netlists(...)	Shortcut to *get_netlists()*.
InnerPin.get_libraries(...)	Shortcut to *get_libraries()*.
InnerPin.get_definitions(...)	Shortcut to *get_definitions()*.
InnerPin.get_instances(...)	Shortcut to *get_instances()*.
InnerPin.get_ports(...)	Shortcut to *get_ports()*.
InnerPin.get_pins(...)	Shortcut to *get_pins()*.
InnerPin.get_cables(...)	Shortcut to *get_cables()*.
InnerPin.get_wires(...)	Shortcut to *get_wires()*.
InnerPin.get_hinstances(...)	Shortcut to *get_hinstances()*.
InnerPin.get_hports(...)	Shortcut to *get_hports()*.
InnerPin.get_hpins(...)	Shortcut to *get_hpins()*.
InnerPin.get_hcables(...)	Shortcut to *get_hcables()*.

continues on next page

Table 3.7 – continued from previous page

InnerPin.get_hwires(...)	Shortcut to *get_hwires()*.

spydrnet.InnerPin.__init__

InnerPin.**__init__**()
Initialize self. See help(type(self)) for accurate signature.

spydrnet.InnerPin.port

property InnerPin.**port**
Return the port that the inner pin is a part of.

This object cannot be modified directly by the end user.

spydrnet.InnerPin.wire

property InnerPin.**wire**
Get the wire the pin is connected to. This value cannot be modified directly by the end user.

spydrnet.InnerPin.clone

InnerPin.**clone**()
Clone the inner pin in an api safe way.

The following conditions will be met:

- The inner pin will be orphaned from any ports
- The pin will not be connected to any wires
- The pin will not be referenced to by any wires or outer pins

spydrnet.InnerPin.get_netlists

InnerPin.**get_netlists**(...)
Shortcut to *get_netlists()*.

spydrnet.InnerPin.get_libraries

InnerPin.**get_libraries**(...)
Shortcut to *get_libraries()*.

spydrnet.InnerPin.get_definitions

InnerPin.**get_definitions**(...)
Shortcut to *get_definitions()*.

spydrnet.InnerPin.get_instances

InnerPin.**get_instances**(...)
Shortcut to *get_instances()*.

spydrnet.InnerPin.get_ports

InnerPin.**get_ports**(...)
Shortcut to *get_ports()*.

spydrnet.InnerPin.get_pins

InnerPin.**get_pins**(...)
Shortcut to *get_pins()*.

spydrnet.InnerPin.get_cables

InnerPin.**get_cables**(...)
Shortcut to *get_cables()*.

spydrnet.InnerPin.get_wires

InnerPin.**get_wires**(...)
Shortcut to *get_wires()*.

spydrnet.InnerPin.get_hinstances

InnerPin.**get_hinstances**(...)
Shortcut to *get_hinstances()*.

spydrnet.InnerPin.get_hports

InnerPin.**get_hports**(...)
Shortcut to *get_hports()*.

spydrnet.InnerPin.get_hpins

InnerPin.**get_hpins**(...)
Shortcut to *get_hpins()*.

spydrnet.InnerPin.get_hcables

InnerPin.**get_hcables**(...)
Shortcut to *get_hcables()*.

spydrnet.InnerPin.get_hwires

InnerPin.**get_hwires**(...)
Shortcut to *get_hwires()*.

3.1.7 OuterPin

Overview

class spydrnet.**OuterPin**(*instance=None*, *inner_pin=None*)
Pins that correspond to instances. These pins can be thought of as on the outside of an instance. There can be many outer pins for each inner pin

Methods

OuterPin.__init__([instance, inner_pin])	create an OuterPin.
OuterPin.from_instance_and_inner_pin(...)	Create an outer pin associated with a given inner_pin and instance object.
OuterPin.instance	Return the instance with which this pin is associated
OuterPin.inner_pin	get the inner pin associated with this outer pin
OuterPin.clone()	Clone the pin in an api safe way.
OuterPin.get_netlists(...)	Shortcut to *get_netlists()*.
OuterPin.get_libraries(...)	Shortcut to *get_libraries()*.
OuterPin.get_definitions(...)	Shortcut to *get_definitions()*.
OuterPin.get_instances(...)	Shortcut to *get_instances()*.
OuterPin.get_ports(...)	Shortcut to *get_ports()*.
OuterPin.get_pins(...)	Shortcut to *get_pins()*.
OuterPin.get_cables(...)	Shortcut to *get_cables()*.
OuterPin.get_wires(...)	Shortcut to *get_wires()*.
OuterPin.get_hinstances(...)	Shortcut to *get_hinstances()*.
OuterPin.get_hports(...)	Shortcut to *get_hports()*.
OuterPin.get_hpins(...)	Shortcut to *get_hpins()*.
OuterPin.get_hcables(...)	Shortcut to *get_hcables()*.
OuterPin.get_hwires(...)	Shortcut to *get_hwires()*.

spydrnet.OuterPin.__init__

OuterPin.**__init__**(*instance=None*, *inner_pin=None*)
create an OuterPin.

Parameters

- **instance - (Instance) the instance with which to associate this outper pin.**
- **inner_pin - (InnerPin) a definition's inner pin to be associated with this instance outer pin.**

spydrnet.OuterPin.from_instance_and_inner_pin

static OuterPin.**from_instance_and_inner_pin**(*instance*, *inner_pin*)
Create an outer pin associated with a given inner_pin and instance object.

Parameters

- **instance - (Instance) the instance to associate with this pin**
- **inner_pin - (InnerPin) the inner pin with which to associate this outer pin**

spydrnet.OuterPin.instance

property OuterPin.**instance**
Return the instance with which this pin is associated

spydrnet.OuterPin.inner_pin

property OuterPin.**inner_pin**
get the inner pin associated with this outer pin

spydrnet.OuterPin.clone

OuterPin.**clone**()
Clone the pin in an api safe way.

The following conditions will be met with the returned outer pin:

- the pin will not be connected to any wires
- the pin will be orphaned from any instance
- the pin will not be connected to any inner pins

spydrnet.OuterPin.get_netlists

OuterPin.**get_netlists**(...)
Shortcut to *get_netlists()*.

spydrnet.OuterPin.get_libraries

OuterPin.**get_libraries**(...)
Shortcut to *get_libraries()*.

spydrnet.OuterPin.get_definitions

OuterPin.**get_definitions**(...)
Shortcut to *get_definitions()*.

spydrnet.OuterPin.get_instances

OuterPin.**get_instances**(...)
Shortcut to *get_instances()*.

spydrnet.OuterPin.get_ports

OuterPin.**get_ports**(...)
Shortcut to *get_ports()*.

spydrnet.OuterPin.get_pins

OuterPin.**get_pins**(...)
Shortcut to *get_pins()*.

spydrnet.OuterPin.get_cables

OuterPin.**get_cables**(...)
Shortcut to *get_cables()*.

spydrnet.OuterPin.get_wires

OuterPin.**get_wires**(...)
Shortcut to *get_wires()*.

spydrnet.OuterPin.get_hinstances

OuterPin.**get_hinstances**(...)
Shortcut to *get_hinstances()*.

spydrnet.OuterPin.get_hports

OuterPin.**get_hports**(...)
Shortcut to *get_hports()*.

spydrnet.OuterPin.get_hpins

OuterPin.**get_hpins**(...)
Shortcut to *get_hpins()*.

spydrnet.OuterPin.get_hcables

OuterPin.**get_hcables**(...)
Shortcut to *get_hcables()*.

spydrnet.OuterPin.get_hwires

OuterPin.**get_hwires**(...)
Shortcut to *get_hwires()*.

3.1.8 Cable

Overview

class spydrnet.**Cable**(*name=None*, *properties=None*, *is_downto=None*, *is_scalar=None*, *lower_index=None*)
Representino of several wires in a collection.

Much like Ports cable extend the bundle class, giving them indexing ability they represent several wires in a collection or bus that are generally related. This could be thought of much like vector types in VHDL ie std_logic_vector (7 downto 0)

Methods

Cable.__init__([name, properties, ...])	Create a cable with no wires and default values for a bundle.
Cable.wires	Gets a list of wires that are in this cable
Cable.create_wire()	Creates a wire and adds it to the cable.
Cable.add_wire(wire[, position])	Adds a wire to the cable at the given position.
Cable.create_wires(wire_count)	Creates wire_count wires for this cable and adds them to it.
Cable.remove_wire(wire)	removes the given wire from the cable and return it.

continues on next page

Table 3.9 – continued from previous page

Cable.remove_wires_from(wires)	Remove all wires given from the cable.
Cable.clone()	Clone the Cable and all of its wires in an api safe way the following will be true of the returned cable
Cable.get_netlists(...)	Shortcut to *get_netlists()*.
Cable.get_libraries(...)	Shortcut to *get_libraries()*.
Cable.get_definitions(...)	Shortcut to *get_definitions()*.
Cable.get_instances(...)	Shortcut to *get_instances()*.
Cable.get_ports(...)	Shortcut to *get_ports()*.
Cable.get_pins(...)	Shortcut to *get_pins()*.
Cable.get_cables(...)	Shortcut to *get_cables()*.
Cable.get_wires(...)	Shortcut to *get_wires()*.
Cable.get_hinstances(...)	Shortcut to *get_hinstances()*.
Cable.get_hports(...)	Shortcut to *get_hports()*.
Cable.get_hpins(...)	Shortcut to *get_hpins()*.
Cable.get_hcables(...)	Shortcut to *get_hcables()*.
Cable.get_hwires(...)	Shortcut to *get_hwires()*.

spydrnet.Cable.__init__

Cable.**__init__**(*name=None*, *properties=None*, *is_downto=None*, *is_scalar=None*, *lower_index=None*)

Create a cable with no wires and default values for a bundle.

Parameters

- **name - (str) the name of this instance**
- **properties - (dict) the dictionary which holds the properties**
- **id_downto - (bool) set the downto status. Downto is False if the right index is higher than the left one, True otherwise**
- **is_scalar - (bool) set the scalar status. Return True if the item is a scalar False otherwise.**
- **lower_index - (int) get the value of the lower index of the array.**

spydrnet.Cable.wires

property Cable.**wires**

Gets a list of wires that are in this cable

spydrnet.Cable.create_wire

Cable.**create_wire**()

Creates a wire and adds it to the cable. Returns the wire that was created

spydrnet.Cable.add_wire

Cable.**add_wire**(*wire*, *position=None*)

Adds a wire to the cable at the given position. This wire must not belong to a cable already

Parameters

- **wire - (Wire) the wire to be added to the cable. This wire must not belong to any other cable.**
- **position - (int, default None) the index in the wires list at which to add the wire.**

spydrnet.Cable.create_wires

Cable.**create_wires**(*wire_count*)

Creates wire_count wires for this cable and adds them to it.

Parameters wire_count - (int) the number of wires to be added to the cable.

spydrnet.Cable.remove_wire

Cable.**remove_wire**(*wire*)

removes the given wire from the cable and return it. The wire must belong to this cable

Parameters wire - (Wire) the wire to be removed from the cable.

spydrnet.Cable.remove_wires_from

Cable.**remove_wires_from**(*wires*)

Remove all wires given from the cable. Each must be a member of this cable.

Parameters wires - (List of Wire objects) wires to be removed from the cable.

spydrnet.Cable.clone

Cable.**clone**()

Clone the Cable and all of its wires in an api safe way the following will be true of the returned cable

- The cable will be orphaned from any definitions
- the wires in the cable will not be connected to any pins
- is_downto, is_scalar, lower_index will be maintained
- the wires in the cable will all have the cable set as the parent

spydrnet.Cable.get_netlists

Cable.**get_netlists**(...)
Shortcut to *get_netlists()*.

spydrnet.Cable.get_libraries

Cable.**get_libraries**(...)
Shortcut to *get_libraries()*.

spydrnet.Cable.get_definitions

Cable.**get_definitions**(...)
Shortcut to *get_definitions()*.

spydrnet.Cable.get_instances

Cable.**get_instances**(...)
Shortcut to *get_instances()*.

spydrnet.Cable.get_ports

Cable.**get_ports**(...)
Shortcut to *get_ports()*.

spydrnet.Cable.get_pins

Cable.**get_pins**(...)
Shortcut to *get_pins()*.

spydrnet.Cable.get_cables

Cable.**get_cables**(...)
Shortcut to *get_cables()*.

spydrnet.Cable.get_wires

Cable.**get_wires**(...)
Shortcut to *get_wires()*.

spydrnet.Cable.get_hinstances

Cable.**get_hinstances**(...)
Shortcut to *get_hinstances()*.

spydrnet.Cable.get_hports

Cable.**get_hports**(...)
Shortcut to *get_hports()*.

spydrnet.Cable.get_hpins

Cable.**get_hpins**(...)
Shortcut to *get_hpins()*.

spydrnet.Cable.get_hcables

Cable.**get_hcables**(...)
Shortcut to *get_hcables()*.

spydrnet.Cable.get_hwires

Cable.**get_hwires**(...)
Shortcut to *get_hwires()*.

3.1.9 Wire

Overview

class spydrnet.**Wire**
Represents a wire object

Methods

Wire.__init__()	Initialize self.
Wire.cable	The cable that the wire contains
Wire.pins	The a list of pins that the wire is connected to
Wire.connect_pin(pin[, position])	Connects a pin to the wire
Wire.disconnect_pin(pin)	Disconnect a pin from the wire
Wire.disconnect_pins_from(pins)	Disconnect a list of pins from the wire
Wire.clone()	clone wire in an api safe way.
Wire.get_netlists(...)	Shortcut to *get_netlists()*.
Wire.get_libraries(...)	Shortcut to *get_libraries()*.
Wire.get_definitions(...)	Shortcut to *get_definitions()*.
Wire.get_instances(...)	Shortcut to *get_instances()*.
Wire.get_ports(...)	Shortcut to *get_ports()*.

continues on next page

Table 3.10 – continued from previous page

Wire.get_pins(...)	Shortcut to *get_pins()*.
Wire.get_cables(...)	Shortcut to *get_cables()*.
Wire.get_wires(...)	Shortcut to *get_wires()*.
Wire.get_hinstances(...)	Shortcut to *get_hinstances()*.
Wire.get_hports(...)	Shortcut to *get_hports()*.
Wire.get_hpins(...)	Shortcut to *get_hpins()*.
Wire.get_hcables(...)	Shortcut to *get_hcables()*.
Wire.get_hwires(...)	Shortcut to *get_hwires()*.

spydrnet.Wire.__init__

Wire.**__init__**()
Initialize self. See help(type(self)) for accurate signature.

spydrnet.Wire.cable

property Wire.**cable**
The cable that the wire contains

spydrnet.Wire.pins

property Wire.**pins**
The a list of pins that the wire is connected to

spydrnet.Wire.connect_pin

Wire.**connect_pin**(*pin*, *position=None*)
Connects a pin to the wire

Parameters value - The pin to connect to

spydrnet.Wire.disconnect_pin

Wire.**disconnect_pin**(*pin*)
Disconnect a pin from the wire

Parameters value - The pin to disconnect

spydrnet.Wire.disconnect_pins_from

Wire.**disconnect_pins_from**(*pins*)
Disconnect a list of pins from the wire

Parameters value - The list of pins to disconnect

spydrnet.Wire.clone

Wire.**clone**()
clone wire in an api safe way.

The following properties can be expected from the returned element:

- The wire is not connected to any pins.
- The wire is orphaned from any cable.
- No pins are connected to the wire

spydrnet.Wire.get_netlists

Wire.**get_netlists**(...)
Shortcut to *get_netlists()*.

spydrnet.Wire.get_libraries

Wire.**get_libraries**(...)
Shortcut to *get_libraries()*.

spydrnet.Wire.get_definitions

Wire.**get_definitions**(...)
Shortcut to *get_definitions()*.

spydrnet.Wire.get_instances

Wire.**get_instances**(...)
Shortcut to *get_instances()*.

spydrnet.Wire.get_ports

Wire.**get_ports**(...)
Shortcut to *get_ports()*.

spydrnet.Wire.get_pins

Wire.**get_pins**(...)
Shortcut to *get_pins()*.

spydrnet.Wire.get_cables

Wire.**get_cables**(...)
Shortcut to *get_cables()*.

spydrnet.Wire.get_wires

Wire.**get_wires**(...)
Shortcut to *get_wires()*.

spydrnet.Wire.get_hinstances

Wire.**get_hinstances**(...)
Shortcut to *get_hinstances()*.

spydrnet.Wire.get_hports

Wire.**get_hports**(...)
Shortcut to *get_hports()*.

spydrnet.Wire.get_hpins

Wire.**get_hpins**(...)
Shortcut to *get_hpins()*.

spydrnet.Wire.get_hcables

Wire.**get_hcables**(...)
Shortcut to *get_hcables()*.

spydrnet.Wire.get_hwires

Wire.**get_hwires**(...)
Shortcut to *get_hwires()*.

3.1.10 HRef

Overview

class spydrnet.util.**HRef**(*item*, *parent=None*)
A hierarchical reference to a specific element in a netlist.

Definitions can be instanced more than once (i.e., multiple instances can reference the same definition). When a definition is instanced more than once, it causes the contents of the definition to be shared. Therefore, any changes to a multi-instanced definition will be reflected in all instances of that definition. Similarly, any references to the contents of a multi-instanced definition refer to the contents of all of the instances and not to the contents of a specific instance. This sharing creates challenges for analyzing and transforming the netlist.

Hierarchical references refer to a netlist element by hierarchical sequence. A hierarchical sequence begins with the the top-instance of netlist (see Netlist.top_instance). The sequence continues with children instances (parent to child) until the instance of interest is reached. The instance of interest is the final instance in the sequence. When the referenced element is an instance, the sequence terminates. When the referenced element is a port, pin, cable, or wire, the sequence continues with those elements until the desired element is specified (e.g., port; port, pin; cable; or cable, wire. In this way, hierarhical elements are uniquely referenced even though the contents of a definition may be shared.

Hierarchical Sequence Examples:

Here are some examples of hierarchical sequences:

- **Top Instance:**
  - `[top_instance]`
- **Top Instance Port**
  - `[top_instance, port]`
- **Top Instance Pin**
  - `[top_instance, port, pin]`
- **Shared Sub-Instance Cable)**
  - `[top_instance, sub_instance_A, sub_instance_C, cable]`
  - `[top_instance, sub_instance_B, sub_instance_C, cable]`
  - `sub_instance_A` and `sub_instance_B` are instances (or children) with the definition referenced by `top_instance`.
  - `sub_instance_A` and `sub_instance_B` reference the same definition, which contains `sub_instance_C`.
  - Even though `cable` is the same element in both sequences, each sequence uniquely references the cable inside `sub_instance_A` and `sub_instance_B` respectively.

Netlist Analysis and Transformation:

Hierarchical references provide unique handles on hierarchical elements. A unique handle allows for such elements to be considered individually even though two hierarchical may point to some of the same elements. This makes it possible, for example, to consider pin connectivity across hierarchy even though the actual pins may be the same.

In some netlist tranformations, it may be desirable to modify the contents of a specific instance without modifying the contents of another instance that refers to the same definition. Hierarchical references make it possible to refer to an instance that should be changed. Once the definition is made unique (see spydrnet.uniquify), then any alterations will only affect the originally specified instance. Hierarchical instances also allow for uniqueness checking (see HRef.is_unique).

Hierarchical Reference Representation:

HRefs represent hierarchy as nodes in a hierarchical tree. The root node is an HRef to the top_instance of a netlist with no parent node. Each HRef contains a pointer to its parent HRef (`None` in the case of the root HRef), a pointer to the element in the netlist that it references, and a hashcode generated from each referenced object.

Storing the hashcode with the object saves on re-computation and allows for quick operations in containers that require Hashable objects. If the hashcode were not stored with the object, it would have to be recalculated for each hash-dependent operation, which could consume a large amount of computational resources depending on the hierarchical depth of the node. Parent and item pointers are immutable. The hashcode of a referenced item

is also immutable. Therefore, the hashcode of a HRef should not change during its existence (even if a netlist transformation renders it invalid).

Use of a Flyweight Pattern:

Due the the nature of hierarchical references, parent nodes can be referenced more than once. Rather than having multiple hierarchical nodes in memory that point to the same hierarchical parent, a flyweight can be used to save on memory. A flyweight pattern is used here to share hierarchical parent nodes. See Flyweight pattern.

Lack of Parent to Child Pointers:

A parent to child pointer requires a lookup diction from each child item to each child hierarchical node. This approach could be taken, but it recreates much of the same information that is available in the original netlist. It was therefore decided to leverage the flyweight pattern rather than explicitly manage all of the necessary child-item to child-node relationships.

item: the item of the object

parent: the parent of the object

Methods

HRef.__init__(item[, parent])	Initialize the href
HRef.get_all_hrefs_of_item(item)	Get all the href of the itsm
HRef.get_all_hrefs_of_instances(instances[, ...])	Assuming all instances are vaild (meaning their reference belongs in a proper library inside a netlist).
HRef.from_sequence(sequence)	Return the href of the sequence
HRef.from_parent_and_item(parent, item)	Return the href with given parent and item
HRef.is_unique	A hierarchical reference must be valid to be unique.
HRef.is_valid	Checks if the href is valid
HRef.name	Stores the name of the href
HRef.get_netlists(...)	Shortcut to *get_netlists()*.
HRef.get_libraries(...)	Shortcut to *get_libraries()*.
HRef.get_definitions(...)	Shortcut to *get_definitions()*.
HRef.get_instances(...)	Shortcut to *get_instances()*.
HRef.get_ports(...)	Shortcut to *get_ports()*.
HRef.get_pins(...)	Shortcut to *get_pins()*.
HRef.get_cables(...)	Shortcut to *get_cables()*.
HRef.get_wires(...)	Shortcut to *get_wires()*.
HRef.get_hinstances(...)	Shortcut to *get_hinstances()*.
HRef.get_hports(...)	Shortcut to *get_hports()*.
HRef.get_hpins(...)	Shortcut to *get_hpins()*.
HRef.get_hcables(...)	Shortcut to *get_hcables()*.
HRef.get_hwires(...)	Shortcut to *get_hwires()*.

spydrnet.util.HRef.__init__

HRef.**__init__**(*item*, *parent=None*)
Initialize the href

Parameters

- **item - the item that the href is reference to**
- **parent - the parent obejct of this href**

spydrnet.util.HRef.get_all_hrefs_of_item

static HRef.**get_all_hrefs_of_item**(*item*)
Get all the href of the itsm

Parameters item - The item to get the href from.

spydrnet.util.HRef.get_all_hrefs_of_instances

static HRef.**get_all_hrefs_of_instances**(*instances*, *netlist=None*)
Assuming all instances are vaild (meaning their reference belongs in a proper library inside a netlist). :param instances: :param netlist: :return:

spydrnet.util.HRef.from_sequence

static HRef.**from_sequence**(*sequence*)
Return the href of the sequence

Parameters sequence - The sequence to get the href from

spydrnet.util.HRef.from_parent_and_item

static HRef.**from_parent_and_item**(*parent*, *item*)
Return the href with given parent and item

Parameters

- **parent - the parent obejct of this href**
- **item - the item that the href is reference to**

spydrnet.util.HRef.is_unique

property HRef.**is_unique**
A hierarchical reference must be valid to be unique. If it is not valid, it may not be unique. : return:

spydrnet.util.HRef.is_valid

property HRef.**is_valid**
Checks if the href is valid

spydrnet.util.HRef.name

property HRef.**name**
Stores the name of the href

spydrnet.util.HRef.get_netlists

HRef.**get_netlists**(...)
Shortcut to *get_netlists()*.

spydrnet.util.HRef.get_libraries

HRef.**get_libraries**(...)
Shortcut to *get_libraries()*.

spydrnet.util.HRef.get_definitions

HRef.**get_definitions**(...)
Shortcut to *get_definitions()*.

spydrnet.util.HRef.get_instances

HRef.**get_instances**(...)
Shortcut to *get_instances()*.

spydrnet.util.HRef.get_ports

HRef.**get_ports**(...)
Shortcut to *get_ports()*.

spydrnet.util.HRef.get_pins

HRef.**get_pins**(...)
Shortcut to *get_pins()*.

spydrnet.util.HRef.get_cables

HRef.**get_cables**(...)
Shortcut to *get_cables()*.

spydrnet.util.HRef.get_wires

HRef.**get_wires**(...)
Shortcut to *get_wires()*.

spydrnet.util.HRef.get_hinstances

HRef.**get_hinstances**(...)
Shortcut to *get_hinstances()*.

spydrnet.util.HRef.get_hports

HRef.**get_hports**(...)
Shortcut to *get_hports()*.

spydrnet.util.HRef.get_hpins

HRef.**get_hpins**(...)
Shortcut to *get_hpins()*.

spydrnet.util.HRef.get_hcables

HRef.**get_hcables**(...)
Shortcut to *get_hcables()*.

spydrnet.util.HRef.get_hwires

HRef.**get_hwires**(...)
Shortcut to *get_hwires()*.

parse(filename)	The parse function is able to parse either a EDIF (.edf) file or a Verilog file (.v)

3.1.11 spydrnet.parse

spydrnet.**parse**(*filename*)

The parse function is able to parse either a EDIF (.edf) file or a Verilog file (.v)

This functions also supports the parsing of .zip files. Such as the ones in support_files folder

Returns The netlist that comes as the result of the parsing of the file if the file was parsed successfully

Return type *Netlist*

Examples

```
>>> import spydrnet as sdn
>>> netlist = sdn.parse('<netlist_filename>.edf')
```

Or we can parse a verilog file

```
>>> netlist = sdn.parse('<netlist_filename>.v')
```

Or a zip file that contains the edif or verilog file

```
>>> netlist = sdn.parse('4bitadder.edf.zip')
```

The following three classes are the classes from which the above elements inherit. They are included here for completeness of documentation and can be used if needed. if the above types will suffice it may be simpler to use them.

Pin()	Pin connects to a single wire.
Bundle()	Parent class of ports and cables.
Element()	Represents an object
FirstClassElement()	Base class of all intermediate representation objects.

3.1.12 spydrnet.ir.Pin

class spydrnet.ir.**Pin**

Pin connects to a single wire.

This class is extended by InnerPin and OuterPin

__init__()

Initialize self. See help(type(self)) for accurate signature.

Methods

__init__()	Initialize self.
get_cables(...)	Shortcut to *get_cables()*.
get_definitions(...)	Shortcut to *get_definitions()*.
get_hcables(...)	Shortcut to *get_hcables()*.
get_hinstances(...)	Shortcut to *get_hinstances()*.
get_hpins(...)	Shortcut to *get_hpins()*.
get_hports(...)	Shortcut to *get_hports()*.

continues on next page

Table 3.14 – continued from previous page

get_hwires(...)	Shortcut to *get_hwires()*.
get_instances(...)	Shortcut to *get_instances()*.
get_libraries(...)	Shortcut to *get_libraries()*.
get_netlists(...)	Shortcut to *get_netlists()*.
get_pins(...)	Shortcut to *get_pins()*.
get_ports(...)	Shortcut to *get_ports()*.
get_wires(...)	Shortcut to *get_wires()*.

Attributes

wire	Get the wire the pin is connected to.

3.1.13 spydrnet.ir.Bundle

class spydrnet.ir.**Bundle**

Parent class of ports and cables.

Since both of these objects represent arrays of objects they both inherit from this parent class.

__init__()

Initialize an element with an empty data dictionary.

Methods

__init__()	Initialize an element with an empty data dictionary.
get(*args, **kwargs)	get the item from the data structure
get_cables(...)	Shortcut to *get_cables()*.
get_definitions(...)	Shortcut to *get_definitions()*.
get_hcables(...)	Shortcut to *get_hcables()*.
get_hinstances(...)	Shortcut to *get_hinstances()*.
get_hpins(...)	Shortcut to *get_hpins()*.
get_hports(...)	Shortcut to *get_hports()*.
get_hwires(...)	Shortcut to *get_hwires()*.
get_instances(...)	Shortcut to *get_instances()*.
get_libraries(...)	Shortcut to *get_libraries()*.
get_netlists(...)	Shortcut to *get_netlists()*.
get_pins(...)	Shortcut to *get_pins()*.
get_ports(...)	Shortcut to *get_ports()*.
get_wires(...)	Shortcut to *get_wires()*.
pop(item)	pop the object from the data structure

Attributes

data	Data stores information about the element
definition	Get the definition that this bundle belongs to.
is_array	This is the logical inverse of is_scalar.
is_downto	Get the downto status of the bundle.
is_scalar	Return True if the item is a scalar False otherwise.
lower_index	Get the value of the lower index of the array.
name	The name of this element

3.1.14 spydrnet.ir.Element

class spydrnet.ir.**Element**

Represents an object

__init__()

Initialize self. See help(type(self)) for accurate signature.

Methods

__init__()	Initialize self.
get_cables(...)	Shortcut to *get_cables()*.
get_definitions(...)	Shortcut to *get_definitions()*.
get_hcables(...)	Shortcut to *get_hcables()*.
get_hinstances(...)	Shortcut to *get_hinstances()*.
get_hpins(...)	Shortcut to *get_hpins()*.
get_hports(...)	Shortcut to *get_hports()*.
get_hwires(...)	Shortcut to *get_hwires()*.
get_instances(...)	Shortcut to *get_instances()*.
get_libraries(...)	Shortcut to *get_libraries()*.
get_netlists(...)	Shortcut to *get_netlists()*.
get_pins(...)	Shortcut to *get_pins()*.
get_ports(...)	Shortcut to *get_ports()*.
get_wires(...)	Shortcut to *get_wires()*.

3.1.15 spydrnet.ir.FirstClassElement

class spydrnet.ir.**FirstClassElement**

Base class of all intermediate representation objects.

An intermediate representation object represents an item in a netlist. Items range in specificity from pins on a port or wires in a cable up to an item that represents the netlist as a whole.

Each element implements a dictionary for storing key-value pairs. The key should be a case sensitive string and the value should be a primitive type (string, integer, float, boolean) or potentially nested collections of primitive types. The purpose of this dictionary is to provide a space for properties and metadata associated with the element.

Key namespaces are separated with a *period* character. If the key is void of a *period* than the key resides in the root namespace. Keys in the root namespace are considered properties. Other keys are considered metadata. For

example '<LANG_OF_ORIGIN>.<METADATA_TAG>':<metadata_value> is considered metadata associated with the netlist's language of origin.

Only data pertinent to the netlist should be stored in this dictionary. Cached data (namespace management, anything that can be recreated from the netlist) should be excluded from this dictionary. The intent of the IR is to house the basis of data for the netlist.

The only key that is reserved is 'NAME'. It is the primary name of the element. NAME may be undefined or inferred, for example, a pin on a port may be nameless, but infer its name for its parent port and position.

__init__()
Initialize an element with an empty data dictionary.

Methods

__init__()	Initialize an element with an empty data dictionary.
get(*args, **kwargs)	get the item from the data structure
get_cables(...)	Shortcut to *get_cables()*.
get_definitions(...)	Shortcut to *get_definitions()*.
get_hcables(...)	Shortcut to *get_hcables()*.
get_hinstances(...)	Shortcut to *get_hinstances()*.
get_hpins(...)	Shortcut to *get_hpins()*.
get_hports(...)	Shortcut to *get_hports()*.
get_hwires(...)	Shortcut to *get_hwires()*.
get_instances(...)	Shortcut to *get_instances()*.
get_libraries(...)	Shortcut to *get_libraries()*.
get_netlists(...)	Shortcut to *get_netlists()*.
get_pins(...)	Shortcut to *get_pins()*.
get_ports(...)	Shortcut to *get_ports()*.
get_wires(...)	Shortcut to *get_wires()*.
pop(item)	pop the object from the data structure

Attributes

data	Data stores information about the element
name	The name of this element

3.2 Getter Functions

get_netlists(obj, ...)	Get netlists *within* an object.
get_libraries(obj, ...)	Get libraries *within* an object.
get_definitions(obj, ...)	Get definitions *within* an object.
get_ports(obj, ...)	Get ports *within* an object.
get_pins(obj, ...)	Get pins *within* an object.
get_cables(obj, ...)	Get cables *within* an object.
get_wires(obj, ...)	Get wires *within* an object.
get_instances(obj, ...)	Get instances *within* an object.
get_hinstances(obj, ...)	Get hierarchical references to instances *within* an object.

continues on next page

Table 3.21 – continued from previous page

get_hports(obj, ...)	Get hierarchical references to ports *within* an object.
get_hpins(obj, ...)	Get hierarchical references to wires *within* an object.
get_hcables(obj, ...)	Get hierarchical references to cables *within* an object.
get_hwires(obj, ...)	Get hierarchical references to wires *within* an object.

3.2.1 spydrnet.get_netlists

spydrnet.**get_netlists**(*obj*, ...)

Get netlists *within* an object.

Parameters

- **obj** (*object, Iterable - required*) – The object or objects associated with this query. Queries return a collection objects associated with the provided object or objects that match the query criteria. For example, *sdn.get_libraries(netlist, ...)* would return all of the libraries associated with the provided netlist that match the additional criteria.
- **patterns** (*str, Iterable - optional, positional or named, (default: wildcard)*) – The search patterns. Patterns can be a single string or an Iterable collection of strings. Patterns can be absolute or they can contain wildcards or regular expressions. If *patterns* is not provided, then it defaults to a wildcard. Patterns are queried against the object property value stored under a specified key. Fast lookups are only attempted on absolute patterns that are not regular expressions and contain no wildcards.
- **key** (*str, optional, (default: ".NAME")*) – This is the key that controls which value is being searched.
- **is_case** (*bool - optional, named, (default: True)*) – Specify if patterns should be treated as case sensitive. Only applies to patterns. Does not alter fast lookup behavior (if namespace policy uses case insensitive indexing, this parameter will not prevent a fast lookup from returning a matching object even if the case is not an exact match).
- **is_re** (*bool - optional, named, (default: False)*) – Specify if patterns are regular expressions. If *False*, a pattern can still contain * and *?* wildcards. A * matches zero or more characters. A *?* matches upto a single character.
- **filter** (*function*) – This is a single input function that can be used to filter out unwanted virtual instances. If not specifed, all matching virtual instances are returned. Otherwise, virtual instances that cause the filter function to evaluate to true are the only items returned.

Returns **netlists** – A generator associated with a particular object

Return type generator

3.2.2 spydrnet.get_libraries

spydrnet.**get_libraries**(*obj*, ...)

Get libraries *within* an object.

Parameters

- **obj** (*object, Iterable - required*) – The object or objects associated with this query. Queries return a collection objects associated with the provided object or objects that match the query criteria. For example, *sdn.get_libraries(netlist, ...)* would return all of the libraries associated with the provided netlist that match the additional criteria.

- **patterns** (*str, Iterable - optional, positional or named, default: wildcard*) – The search patterns. Patterns can be a single string or an Iterable collection of strings. Patterns can be absolute or they can contain wildcards or regular expressions. If *patterns* is not provided, then it defaults to a wildcard. Patterns are queried against the object property value stored under a specified key. Fast lookups are only attempted on absolute patterns that are not regular expressions and contain no wildcards.
- **key** (*str, optional, default: ".NAME"*) – This is the key that controls which value is being searched.
- **is_case** (*bool - optional, named, default: True*) – Specify if patterns should be treated as case sensitive. Only applies to patterns. Does not alter fast lookup behavior (if namespace policy uses case insensitive indexing, this parameter will not prevent a fast lookup from returning a matching object even if the case is not an exact match).
- **is_re** (*bool - optional, named, default: False*) – Specify if patterns are regular expressions. If *False*, a pattern can still contain * and *?* wildcards. A * matches zero or more characters. A *?* matches upto a single character.
- **recursive** (*bool - optional, default: False*) – Specify if search should be recursive or not meaning that sub hierarchical instances within an instance are included or not.
- **selection** (*Selection.{INSIDE, OUTSIDE}, default: INSIDE*) – This parameter determines the instances that are returned based on the definition that is being searched. This parameter only applies to objects that are definitions. If the selection is "INSIDE" (default), then the function will return all of the instances that are inside the definition (i.e., the definition's children) that match the remainder of the search criteria. If the selection is "OUTSIDE", then the function will return all of the instances of the provided definition that match the remainder of the search criteria.
- **filter** (*function*) – This is a single input function that can be used to filter out unwanted virtual instances. If not specifed, all matching virtual instances are returned. Otherwise, virtual instances that cause the filter function to evaluate to true are the only items returned.

Returns **libraries** – The libraries associated with a particular object

Return type generator

3.2.3 spydrnet.get_definitions

spydrnet.**get_definitions**(*obj*, ...)

Get definitions *within* an object.

Parameters

- **obj** (*object, Iterable - required*) – The object or objects associated with this query. Queries return a collection objects associated with the provided object or objects that match the query criteria. For example, *sdn.get_definitions(library, ...)* would return all of the definitions associated with the provided library that match the additional criteria.
- **patterns** (*str, Iterable - optional, positional or named, default: wildcard*) – The search patterns. Patterns can be a single string or an Iterable collection of strings. Patterns can be absolute or they can contain wildcards or regular expressions. If *patterns* is not provided, then it defaults to a wildcard. Patterns are queried against the object property value stored under a specified key. Fast lookups are only attempted on absolute patterns that are not regular expressions and contain no wildcards.
- **key** (*str, optional, default: ".NAME"*) – This is the key that controls which value is being searched.

- **is_case** (*bool - optional, named, default: True*) – Specify if patterns should be treated as case sensitive. Only applies to patterns. Does not alter fast lookup behavior (if namespace policy uses case insensitive indexing, this parameter will not prevent a fast lookup from returning a matching object even if the case is not an exact match).
- **is_re** (*bool - optional, named, default: False*) – Specify if patterns are regular expressions. If *False*, a pattern can still contain * and *?* wildcards. A * matches zero or more characters. A *?* matches upto a single character.
- **recursive** (*bool - optional, default: False*) – Specify if search should be recursive or not meaning that sub hierarchical instances within an instance are included or not.
- **selection** (*Selection.{INSIDE, OUTSIDE}, default: INSIDE*) – This parameter determines the instances that are returned based on the definition that is being searched. This parameter only applies to objects that are definitions. If the selection is "INSIDE" (default), then the function will return all of the instances that are inside the definition (i.e., the definition's children) that match the remainder of the search criteria. If the selection is "OUTSIDE", then the function will return all of the instances of the provided definition that match the remainder of the search criteria.
- **filter** (*function*) – This is a single input function that can be used to filter out unwanted virtual instances. If not specifed, all matching virtual instances are returned. Otherwise, virtual instances that cause the filter function to evaluate to true are the only items returned.

Returns **definitions** – The definitions associated with a particular object or collection of objects.

Return type generator

3.2.4 spydrnet.get_ports

spydrnet.**get_ports**(*obj*, ...)

Get ports *within* an object.

Parameters

- **obj** (*object, Iterable - required*) – The object or objects associated with this query. Queries return a collection objects associated with the provided object or objects that match the query criteria. For example, *sdn.get_ports(definition, . . .)* would return all of the ports associated with the provided definition that match the additional criteria.
- **patterns** (*str, Iterable - optional, positional or named, default: wildcard*) – The search patterns. Patterns can be a single string or an Iterable collection of strings. Patterns can be absolute or they can contain wildcards or regular expressions. If *patterns* is not provided, then it defaults to a wildcard. Patterns are queried against the object property value stored under a specified key. Fast lookups are only attempted on absolute patterns that are not regular expressions and contain no wildcards.
- **key** (*str, optional, default: ".NAME"*) – This is the key that controls which value is being searched.
- **is_case** (*bool - optional, named, default: True*) – Specify if patterns should be treated as case sensitive. Only applies to patterns. Does not alter fast lookup behavior (if namespace policy uses case insensitive indexing, this parameter will not prevent a fast lookup from returning a matching object even if the case is not an exact match).
- **is_re** (*bool - optional, named, default: False*) – Specify if patterns are regular expressions. If *False*, a pattern can still contain * and *?* wildcards. A * matches zero or more characters. A *?* matches upto a single character.

- **filter** (*function*) – This is a single input function that can be used to filter out unwanted virtual instances. If not specifed, all matching virtual instances are returned. Otherwise, virtual instances that cause the filter function to evaluate to true are the only items returned.

Returns **ports** – The ports associated with a particular object or collection of objects.

Return type generator

3.2.5 spydrnet.get_pins

spydrnet.**get_pins**(*obj*, ...)

Get pins *within* an object.

Parameters

- **obj** (*object, Iterable - required*) – The object or objects associated with this query. Queries return a collection objects associated with the provided object or objects that match the query criteria. For example, *sdn.get_ports(definition, ...)* would return all of the ports associated with the provided definition that match the additional criteria.
- **selection** (*Selection.{INSIDE, OUTSIDE}, default: INSIDE*) – Controls the type of pin returned. Setting this parameter to "OUTSIDE" will return the outer pins of the objects queried. For cables, this returns the corresponding pin based on this parameter.
- **filter** (*function*) – This is a single input function that can be used to filter out unwanted virtual instances. If not specifed, all matching virtual instances are returned. Otherwise, virtual instances that cause the filter function to evaluate to true are the only items returned.

Returns **pins** – The pins associated with a particular object or collection of objects.

Return type generator

3.2.6 spydrnet.get_cables

spydrnet.**get_cables**(*obj*, ...)

Get cables *within* an object.

Parameters

- **obj** (*object, Iterable - required*) – The object or objects associated with this query. Queries return a collection objects associated with the provided object or objects that match the query criteria. For example, *sdn.get_cables(definition, ...)* would return all of the cables associated with the provided definition that match the additional criteria.
- **patterns** (*str, Iterable - optional, positional or named, default: wildcard*) – The search patterns. Patterns can be a single string or an Iterable collection of strings. Patterns can be absolute or they can contain wildcards or regular expressions. If *patterns* is not provided, then it defaults to a wildcard. Patterns are queried against the object property value stored under a specified key. Fast lookups are only attempted on absolute patterns that are not regular expressions and contain no wildcards.
- **key** (*str, optional, default: ".NAME"*) – This is the key that controls which value is being searched.
- **is_case** (*bool - optional, named, default: True*) – Specify if patterns should be treated as case sensitive. Only applies to patterns. Does not alter fast lookup behavior (if namespace policy uses case insensitive indexing, this parameter will not prevent a fast lookup from returning a matching object even if the case is not an exact match).

- **is_re** (*bool - optional, named, default: False*) – Specify if patterns are regular expressions. If *False*, a pattern can still contain * and *?* wildcards. A * matches zero or more characters. A *?* matches upto a single character.
- **selection** (*Selection.{INSIDE, OUTSIDE, BOTH, ALL}, default: INSIDE*) – This parameter determines the wires that are returned based on the instance associated with the object that is being searched.
- **recursive** (*bool - optional, default: False*) – Specify if search should be recursive or not meaning that sub hierarchical instances within an instance are included or not.
- **filter** (*function*) – This is a single input function that can be used to filter out unwanted virtual instances. If not specifed, all matching virtual instances are returned. Otherwise, virtual instances that cause the filter function to evaluate to true are the only items returned.

Returns **cables** – The cables associated with a particular object or collection of objects.

Return type generator

3.2.7 spydrnet.get_wires

spydrnet.**get_wires**(*obj*, ...)

Get wires *within* an object.

Parameters

- **obj** (*object, Iterable - required*) – The object or objects associated with this query. Queries return a collection of objects associated with the provided object or objects that match the query criteria. For example, *sdn.get_instances(netlist, . . .)* would return all of the instances *within* the provided definition that match the additional criteria.
- **recursive** (*bool - optional, default: False*) – Specify if search should be recursive or not meaning that sub hierarchical instances within an instance are included or not.
- **selection** (*Selection.{INSIDE, OUTSIDE, BOTH, ALL}, default: INSIDE*) – This parameter determines the wires that are returned based on the instance associated with the object that is being searched.
- **filter** (*function*) – This is a single input function that can be used to filter out unwanted virtual instances. If not specifed, all matching virtual instances are returned. Otherwise, virtual instances that cause the filter function to evaluate to true are the only items returned.

Returns **wires** – The wires associated with a particular object or collection of objects.

Return type generator

3.2.8 spydrnet.get_instances

spydrnet.**get_instances**(*obj*, ...)

Get instances *within* an object.

Parameters

- **obj** (*object, Iterable - required*) – The object or objects associated with this query. Queries return a collection objects associated with the provided object or objects that match the query criteria. For example, *sdn.get_instances(definition, . . .)* would return all of the instances *within* the provided definition that match the additional criteria.

- **patterns** (*str, Iterable - optional, positional or named, default: wildcard*) – The search patterns. Patterns can be a single string or an Iterable collection of strings. Patterns can be absolute or they can contain wildcards or regular expressions. If *patterns* is not provided, then it defaults to a wildcard. Patterns are queried against the object property value stored under a specified key. Fast lookups are only attempted on absolute patterns that are not regular expressions and contain no wildcards.
- **key** (*str, optional, default: ".NAME"*) – This is the key that controls which value is being searched.
- **is_case** (*bool - optional, named, default: True*) – Specify if patterns should be treated as case sensitive. Only applies to patterns. Does not alter fast lookup behavior (if namespace policy uses case insensitive indexing, this parameter will not prevent a fast lookup from returning a matching object even if the case is not an exact match).
- **is_re** (*bool - optional, named, default: False*) – Specify if patterns are regular expressions. If *False*, a pattern can still contain * and *?* wildcards. A * matches zero or more characters. A *?* matches upto a single character.
- **selection** (*Selection.{INSIDE, OUTSIDE}, default: INSIDE*) – This parameter determines the instances that are returned based on the definition or instance that is being searched. This parameter only applies to objects that are definitions. If the selection is "INSIDE" (default), then the function will return all of the instances that are inside the definition (i.e., the definition's children) that match the remainder of the search criteria. If the selection is "OUTSIDE", then the function will return all of the instances of the provided definition that match the remainder of the search criteria.
- **recursive** (*bool - optional, default: False*) – Specify if search should be recursive or not meaning that sub hierarchical instances within an instance are included or not.
- **filter** (*function*) – This is a single input function that can be used to filter out unwanted virtual instances. If not specifed, all matching virtual instances are returned. Otherwise, virtual instances that cause the filter function to evaluate to true are the only items returned.

Returns **cables** – The cables associated with a particular object or collection of objects.

Return type generator

3.2.9 spydrnet.get_hinstances

spydrnet.**get_hinstances**(*obj*, ...)

Get hierarchical references to instances *within* an object.

Parameters

- **obj** (*object, Iterable - required*) – The object or objects associated with this query. Queries return a collection of objects associated with the provided object or objects that match the query criteria. For example, *sdn.get_instances(netlist, . . .)* would return all of the instances *within* the provided definition that match the additional criteria.
- **patterns** (*str, Iterable - optional, positional or named, default: wildcard*) – The search patterns. Patterns can be a single string or an Iterable collection of strings. Patterns can be absolute or they can contain wildcards or regular expressions. If *patterns* is not provided, then it defaults to a wildcard.
- **is_case** (*bool - optional, named, default: True*) – Specify if patterns should be treated as case sensitive. Only applies to patterns. Does not alter fast lookup behavior (if namespace policy uses case insensitive indexing, this parameter will not prevent a fast lookup from returning a matching object even if the case is not an exact match).

- **is_re** (*bool - optional, named, default: False*) – Specify if patterns are regular expressions. If *False*, a pattern can still contain * and *?* wildcards. A * matches zero or more characters. A *?* matches upto a single character.
- **recursive** (*bool - optional, default: False*) – Specify if search should be recursive or not meaning that sub hierarchical instances within an instance are included or not.
- **filter** (*function*) – This is a single input function that can be used to filter out unwanted virtual instances. If not specifed, all matching virtual instances are returned. Otherwise, virtual instances that cause the filter function to evaluate to true are the only items returned.

Returns **href_instances** – The hierarchical references to instances associated with a particular object or collection of objects.

Return type generator

3.2.10 spydrnet.get_hports

spydrnet.**get_hports**(*obj*, ...)

Get hierarchical references to ports *within* an object.

Parameters

- **obj** (*object, Iterable - required*) – The object or objects associated with this query. Queries return a collection of objects associated with the provided object or objects that match the query criteria. For example, *sdn.get_instances(netlist, . . .)* would return all of the instances *within* the provided definition that match the additional criteria.
- **patterns** (*str, Iterable - optional, positional or named, default: wildcard*) – The search patterns. Patterns can be a single string or an Iterable collection of strings. Patterns can be absolute or they can contain wildcards or regular expressions. If *patterns* is not provided, then it defaults to a wildcard.
- **recursive** (*bool - optional, default: False*) – Specify if search should be recursive or not meaning that sub hierarchical pins within an instance are included or not.
- **is_case** (*bool - optional, named, default: True*) – Specify if patterns should be treated as case sensitive. Only applies to patterns. Does not alter fast lookup behavior (if namespace policy uses case insensitive indexing, this parameter will not prevent a fast lookup from returning a matching object even if the case is not an exact match).
- **is_re** (*bool - optional, named, default: False*) – Specify if patterns are regular expressions. If *False*, a pattern can still contain * and *?* wildcards. A * matches zero or more characters. A *?* matches upto a single character.
- **filter** (*function*) – This is a single input function that can be used to filter out unwanted virtual instances. If not specifed, all matching virtual instances are returned. Otherwise, virtual instances that cause the filter function to evaluate to true are the only items returned.

Returns **href_ports** – The hierarchical references to ports associated with a particular object or collection of objects.

Return type generator

3.2.11 spydrnet.get_hpins

spydrnet.**get_hpins**(*obj*, ...)

Get hierarchical references to wires *within* an object.

Parameters

- **obj** (*object, Iterable - required*) – The object or objects associated with this query. Queries return a collection of objects associated with the provided object or objects that match the query criteria. For example, *sdn.get_instances(netlist, . . .)* would return all of the instances *within* the provided definition that match the additional criteria.
- **patterns** (*str, Iterable - optional, positional or named, default: wildcard*) – The search patterns. Patterns can be a single string or an Iterable collection of strings. Patterns can be absolute or they can contain wildcards or regular expressions. If *patterns* is not provided, then it defaults to a wildcard.
- **recursive** (*bool - optional, default: False*) – Specify if search should be recursive or not meaning that sub hierarchical pins within an instance are included or not.
- **is_case** (*bool - optional, named, default: True*) – Specify if patterns should be treated as case sensitive. Only applies to patterns. Does not alter fast lookup behavior (if namespace policy uses case insensitive indexing, this parameter will not prevent a fast lookup from returning a matching object even if the case is not an exact match).
- **is_re** (*bool - optional, named, default: False*) – Specify if patterns are regular expressions. If *False*, a pattern can still contain * and *?* wildcards. A * matches zero or more characters. A *?* matches upto a single character.
- **filter** (*function*) – This is a single input function that can be used to filter out unwanted virtual instances. If not specifed, all matching virtual instances are returned. Otherwise, virtual instances that cause the filter function to evaluate to true are the only items returned.

Returns **href_pins** – The hierarchical references to pins associated with a particular object or collection of objects.

Return type generator

3.2.12 spydrnet.get_hcables

spydrnet.**get_hcables**(*obj*, ...)

Get hierarchical references to cables *within* an object.

Parameters

- **obj** (*object, Iterable - required*) – The object or objects associated with this query. Queries return a collection of objects associated with the provided object or objects that match the query criteria. For example, *sdn.get_instances(netlist, . . .)* would return all of the instances *within* the provided definition that match the additional criteria.
- **patterns** (*str, Iterable - optional, positional or named, default: wildcard*) – The search patterns. Patterns can be a single string or an Iterable collection of strings. Patterns can be absolute or they can contain wildcards or regular expressions. If *patterns* is not provided, then it defaults to a wildcard.
- **is_case** (*bool - optional, named, default: True*) – Specify if patterns should be treated as case sensitive. Only applies to patterns. Does not alter fast lookup behavior (if namespace policy uses case insensitive indexing, this parameter will not prevent a fast lookup from returning a matching object even if the case is not an exact match).

- **is_re** (*bool - optional, named, default: False*) – Specify if patterns are regular expressions. If *False*, a pattern can still contain * and *?* wildcards. A * matches zero or more characters. A *?* matches upto a single character.
- **selection** (*Selection.{INSIDE, OUTSIDE, BOTH, ALL}, default: INSIDE*) – This parameter determines the wires that are returned based on the instance associated with the object that is being searched.
- **recursive** (*bool - optional, default: False*) – Specify if search should be recursive or not meaning that sub hierarchical instances within an instance are included or not.
- **filter** (*function*) – This is a single input function that can be used to filter out unwanted virtual instances. If not specifed, all matching virtual instances are returned. Otherwise, virtual instances that cause the filter function to evaluate to true are the only items returned.

Returns **href_cables** – The hierarchical references to cables associated with a particular object or collection of objects.

Return type generator

3.2.13 spydrnet.get_hwires

spydrnet.**get_hwires**(*obj*, ...)

Get hierarchical references to wires *within* an object.

Parameters

- **obj** (*object, Iterable - required*) – The object or objects associated with this query. Queries return a collection of objects associated with the provided object or objects that match the query criteria. For example, *sdn.get_instances(netlist, . . .)* would return all of the instances *within* the provided definition that match the additional criteria.
- **patterns** (*str, Iterable - optional, positional or named, default: wildcard*) – The search patterns. Patterns can be a single string or an Iterable collection of strings. Patterns can be absolute or they can contain wildcards or regular expressions. If *patterns* is not provided, then it defaults to a wildcard.
- **is_case** (*bool - optional, named, default: True*) – Specify if patterns should be treated as case sensitive. Only applies to patterns. Does not alter fast lookup behavior (if namespace policy uses case insensitive indexing, this parameter will not prevent a fast lookup from returning a matching object even if the case is not an exact match).
- **is_re** (*bool - optional, named, default: False*) – Specify if patterns are regular expressions. If *False*, a pattern can still contain * and *?* wildcards. A * matches zero or more characters. A *?* matches upto a single character.
- **recursive** (*bool - optional, default: False*) – Specify if search should be recursive or not meaning that sub hierarchical instances within an instance are included or not.
- **selection** (*Selection.{INSIDE, OUTSIDE, BOTH, ALL}, default: INSIDE*) – This parameter determines the wires that are returned based on the instance associated with the object that is being searched.
- **filter** (*function*) – This is a single input function that can be used to filter out unwanted virtual instances. If not specifed, all matching virtual instances are returned. Otherwise, virtual instances that cause the filter function to evaluate to true are the only items returned.

Returns **href_wires** – The hierarchical references to wires associated with a particular object or collection of objects.

Return type generator

3.3 Other Functions

compose(netlist, filename)	To compose a file into a netlit format

3.3.1 spydrnet.compose

spydrnet.**compose**(*netlist*, *filename*)
To compose a file into a netlit format

flatten(netlist)	starts at the top instance and brings all the different subelements to the top level.

3.3.2 spydrnet.flatten.flatten

spydrnet.flatten.**flatten**(*netlist*)
starts at the top instance and brings all the different subelements to the top level. and port boundries are redone into one net.

uniquify(netlist)	Make the instances in the netlist unique uniqification is done in place.

3.3.3 spydrnet.uniquify.uniquify

spydrnet.uniquify.**uniquify**(*netlist*)
Make the instances in the netlist unique uniqification is done in place. Each instance will correspond to exactly one definition and each definition will correspond to exactly one instance with the exception of leaf cells. Leaf cells are can be instanced unlimited numbers of times. Any netlist elements that are not instantiated by the top instance will not be modified and may retain duplicate instances Currently there is no guarantee that the original definition names will be maintained, but it is guaranteed that they will be unique within the scope of all hardware that is below the top instance.

Renameing is predictable. the string: _sdn_unique_# will be added to the end of the definition names.

parameter - netlist, the netlist that will be uniquified

returns - no returns

clone(element)	Clone any netlist objects

3.3.4 spydrnet.clone.clone

spydrnet.clone.**clone**(*element*)
Clone any netlist objects

several premises hold while cloning

- the object will be orphaned and not belong to any parent
- the object will maintain internal structure with cloned objects

- the names will be unchanged
- external connections will mostly be severed

Properties

- cloned using python's built in deepcopy functionality.
- expected to be string objects but if you store something else there make sure you override deepcopy on that object.

Instances have some special considerations

- when cloned without the library containing the reference definition the instance will still point to the definition of it's clone.
- in the same case as the above point the references of the definition will be updated accordingly
- when a library is cloned some of the instances may be defined in another library these instances will follow the premises above
- instances defined and referenced in the cloned library will point to the cloned definition

3.4 Shortcuts

3.4.1 GetterShortcuts

Overview

class spydrnet.shortcuts.getter.**GetterShortcuts**

Methods

GetterShortcuts.get_netlists(...)	Shortcut to *get_netlists()*.
GetterShortcuts.get_libraries(...)	Shortcut to *get_libraries()*.
GetterShortcuts.get_definitions(...)	Shortcut to *get_definitions()*.
GetterShortcuts.get_instances(...)	Shortcut to *get_instances()*.
GetterShortcuts.get_ports(...)	Shortcut to *get_ports()*.
GetterShortcuts.get_pins(...)	Shortcut to *get_pins()*.
GetterShortcuts.get_cables(...)	Shortcut to *get_cables()*.
GetterShortcuts.get_wires(...)	Shortcut to *get_wires()*.
GetterShortcuts.get_hinstances(...)	Shortcut to *get_hinstances()*.
GetterShortcuts.get_hports(...)	Shortcut to *get_hports()*.
GetterShortcuts.get_hpins(...)	Shortcut to *get_hpins()*.
GetterShortcuts.get_hcables(...)	Shortcut to *get_hcables()*.
GetterShortcuts.get_hwires(...)	Shortcut to *get_hwires()*.

spydrnet.shortcuts.getter.GetterShortcuts.get_netlists

GetterShortcuts.**get_netlists**(...)
Shortcut to *get_netlists()*.

spydrnet.shortcuts.getter.GetterShortcuts.get_libraries

GetterShortcuts.**get_libraries**(...)
Shortcut to *get_libraries()*.

spydrnet.shortcuts.getter.GetterShortcuts.get_definitions

GetterShortcuts.**get_definitions**(...)
Shortcut to *get_definitions()*.

spydrnet.shortcuts.getter.GetterShortcuts.get_instances

GetterShortcuts.**get_instances**(...)
Shortcut to *get_instances()*.

spydrnet.shortcuts.getter.GetterShortcuts.get_ports

GetterShortcuts.**get_ports**(...)
Shortcut to *get_ports()*.

spydrnet.shortcuts.getter.GetterShortcuts.get_pins

GetterShortcuts.**get_pins**(...)
Shortcut to *get_pins()*.

spydrnet.shortcuts.getter.GetterShortcuts.get_cables

GetterShortcuts.**get_cables**(...)
Shortcut to *get_cables()*.

spydrnet.shortcuts.getter.GetterShortcuts.get_wires

GetterShortcuts.**get_wires**(...)
Shortcut to *get_wires()*.

spydrnet.shortcuts.getter.GetterShortcuts.get_hinstances

GetterShortcuts.**get_hinstances**(...)
Shortcut to *get_hinstances()*.

spydrnet.shortcuts.getter.GetterShortcuts.get_hports

GetterShortcuts.**get_hports**(...)
Shortcut to *get_hports()*.

spydrnet.shortcuts.getter.GetterShortcuts.get_hpins

GetterShortcuts.**get_hpins**(...)
Shortcut to *get_hpins()*.

spydrnet.shortcuts.getter.GetterShortcuts.get_hcables

GetterShortcuts.**get_hcables**(...)
Shortcut to *get_hcables()*.

spydrnet.shortcuts.getter.GetterShortcuts.get_hwires

GetterShortcuts.**get_hwires**(...)
Shortcut to *get_hwires()*.

APPENDIX

A

TUTORIAL

SpyDrNet is a tool for the analysis and transformation of structural netlists. A structural netlist is a static representation of an electronic circuit. A circuit consists of a number of electrical components and their connections. Figure Fig. 1.1 shows a graphical representation of a netlist.

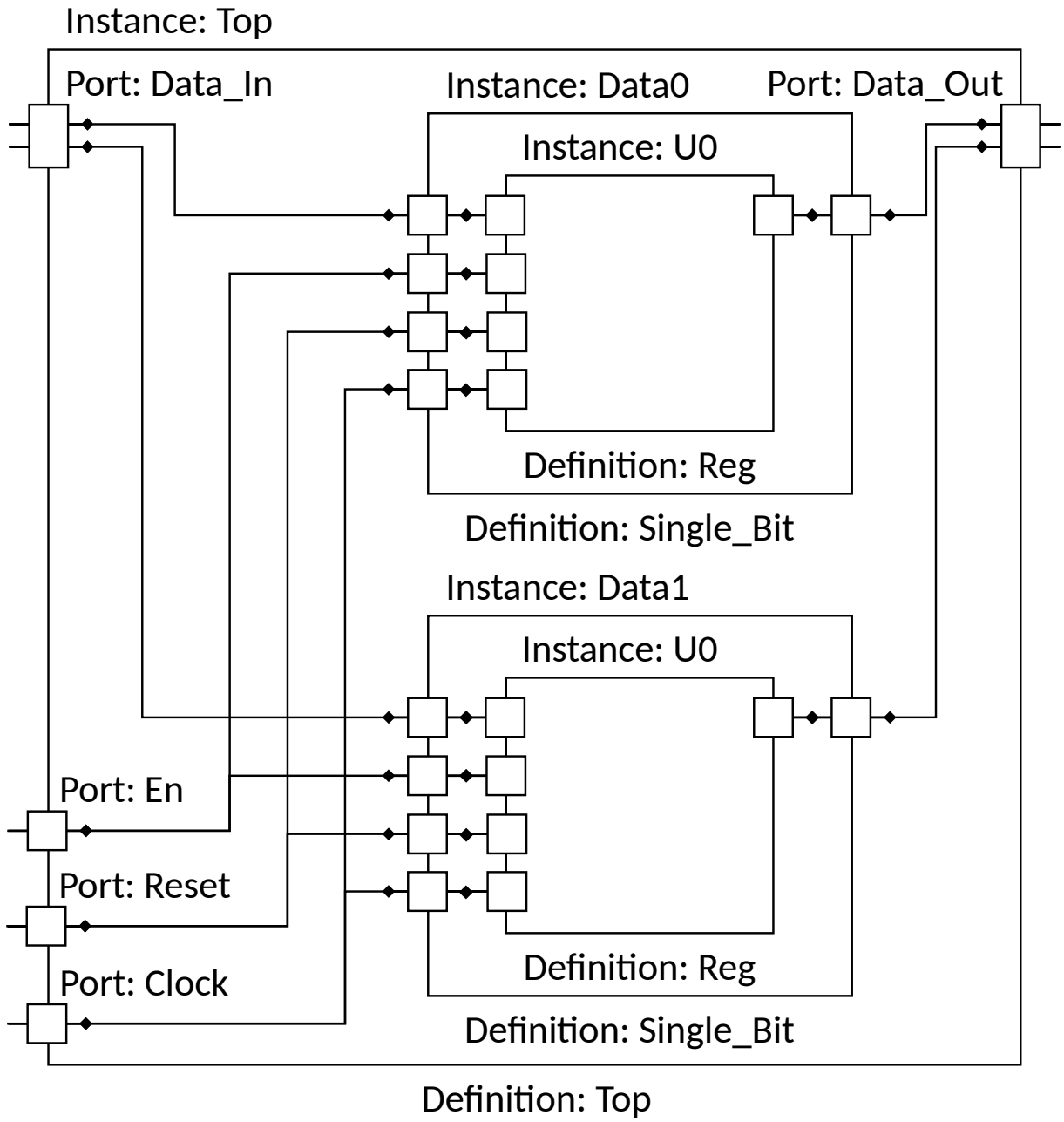

Fig. 1.1: Hierarchical representation of a Netlist

Figure Fig. 1.1 represents a netlist hierarchically. This representation is commonly found in many schematic views of electronic design automation (EDA) or computer aided design (CAD) tools. It presents a top level instance of a definition that in turn instances other definitions. Instances are connected accordingly and connections carry through hierarchical boundaries.

Most hardware description languages and netlist representations are capable of representing a structural netlist hierarchically.

The most basic element of a netlist is a definition. Verilog and System Verilog refer to a definition as a module. VHDL

refers to a definition as an entity. EDIF refers to a definition as a cell. While each language and framework has a different name, the functioning role of a definition is virtually the same. A definition defines the interface and contents of a component within a netlist.

An instance is an instantiation of a definition.

Hierarchy organizes larger netlists into a collection of higher-level and lower-level definitions of smaller netlists. Higher-level definitions instance lower-level definitions.

A.1 Installation

For installing instructions, please refer INSTALL.rst

A.2 Working Environment

SpyDrNet is coded in Python, and requires Python 3.5 or newer versions of Python. In order to import SpyDrNet to the project, use the following code:

```
>>> import spydrnet as sdn
>>>
```

In this tutorial, we will use 'sdn' as a shorcut for SpyDrNet to manipulate all the commands.

A.3 Parsing

SpyDrNet currently supports the parsing and composing for EDIF file and Verilog file

To parse a file, enter the following command for EDIF file

```
>>> netlist = sdn.parse('<netlist_filename>.edf')
```

Or the following for Verilog file

```
>>> netlist = sdn.parse('<netlist_filename>.v')
```

Nelist is an intermediate representation (IR). We are able to modify the netlist and add new elements. The following code returns the name of the top isntance of the netlist:

```
>>> netlist.top_instance.name
```

This creates a new library and the library is added to the nestlist. For the entire documentation of SpyDrNet, please refer *API Summary*

```
>>> netlist.create_library()
```

A.4 Composing

To compose a file, enter the following command

```
>>> sdn.compose(netlist, '<filename>.edf')
```

A new file named '<filename>.edf' should be generated in the working directory.

A.5 Examples

A.5.1 Creating from scratch

We can also create the hardware design from scratch

```
>>> import spydrnet as sdn
>>> nelist = sdn.Netlist('myNetlist')
>>> instance = sdn.Instance()
```

For full details regarding the initialization of a Netlist obejct, see *Netlist*

A.5.2 Renaming an objects

```
>>> instance.name = "my_instance"
```

A.5.3 Setting properties

```
>>> instance['NAME'] = "name"
```

For more of the functionality, features, and uses of SpyDrNet, please visit sec:examples

Below is a list of all the intermediate representations (IR) used by SpyDrNet. See *API Summary* for API specification.

A.6 Intermediate Representation

SpyDrNet's intermediate representation of netlists (IR) is what sets it apart for other EDA tools. The IR is structured to house netlists in a generic way while allowing for format specific constructs to be preserved.

`Element` Most IR classes inherit from this Python class. Objects of this class are referred to as a netlist elements. A netlist element contains a dictionary for storing data specific to itself. This is accomplished using Python get/set item functions, (see *Element Data*).

`Netlist` This class of Python objects is the netlist element with the highest level of organization (a whole netlist). It contains an ordered collection of libraries and any data associated with the netlist as a whole.

`Library` This netlist element contains an ordered collection of cell or module definitions associated with a library.

`Definition` A definition outlines the contents of each component that can be instantiated elsewhere in the design. It holds information that is pertinant to all instances of itself including subcomponents ports and connections

`Instance` This element holds pointers to the definition which it instances, and contains its own set of pins to be connected to within its parent definition.

Bundle The Bundle class is a parent class of Ports and Cables because each can be thought of as an array. This class defines the structure that helps us properly represent array objects in netlists including the width, direction (to or downto) and starting index. As a parent class this class is not directly instantiated in netlist.

Port The Port element inherits from Bundles and can be thought of as containing the information on how a Definition connects the outside world to the elements it contains.

Cable Cables are bundles of connectors between components within a definition. They connect ports to their destination pins

Pin The pin class is also a parent class, inherited from by the inner pin and outer pin objects. Unlike the Element and Bundle objects, Pins are useful because they can hide some of the implementation details of the underlying inner pins and outer pins.

InnerPin These pins are collected in Ports and are contained on the inside of the definitions. There is one set of inner pins per definition but they could refer to several sets of OuterPins

OuterPin These pins are collected on instances. They let us distinguish between connections to multiple instances of a single definition. These objects remove the need to carefuly track hierarcy while navegating a netlist.

Wire Wires are grouped inside cables and are elements that help hold connection information between single pins on instances within a definition and within it's ports.

More detail on the IR is provided in *API Summary*.

INDEX

Symbols

A

B

C

D

E

F

G

H

I

L

N

O

P

R

T

U

W