thesdk Documentation

Release 1.7_RC

Oct 25, 2021

CONTENTS:

Superclass class of TheSyDeKick - universal System Development Kit. Provides commmon methods and utility classes for other classes in TheSyDeKick.

Created by Marko Kosunen, marko.kosunen@aalto.fi, 2017.

Documentation instructions

Current docstring documentation style is Numpy https://numpydoc.readthedocs.io/en/latest/format.html

This text here is to remind you that documentation is important. However, you may find out that even the documentation of this entity may be outdated and incomplete. Regardless of that, every day and in every way we are getting better and better :).

class thesdk.**thesdk**

Following class attributes are set when this class imported

HOME

Directory ../../../ counting from location __init__.py file of thesdkclass. Used as a reference point for other locations

Type str

CONFIGFILE

HOME/TheSDK.config.

Type str

MODULEPATHS

List of directories under HOME/Entities that contain __init__.py file. Appended to sys.path to locate TheSyDeKick system modules

Type str

logfile

Default logfile: /tmp/TheSDK_randomstr_uname_YYYYMMDDHHMM.log Override with initlog if you want something else

Type str

global_parameters

List of global parameters to be read to GLOBALS dictionary from CONFIGFILE

Type list(str)

GLOBALS

Dictionary of global parameters, keys defined by global_parameters, values defined in CONFIGFILE

Type dict

classmethod initlog(**arg*)

Initializes logging. logfile passed as a parameter

abstract property _classfile

Abstract property of thesdk class. Defines the location of the classfile.

Define in every child class of thesdk as:

```
def _classfile(self):
    return os.path.dirname(os.path.realpath(__file__)) + "/"+__name__
```

property entitypath

Path to entity. Extracted from the location of __init__.py file.

property model
Simulation model to be used

'py' | 'sv' | 'vhdl' | 'eldo' | 'spectre' | 'hw'

property simpath
Simulation directory according to model.

Default: self.entitypath/Simulations/<simulator>sim For verilog and vhdl <simulator> is 'rtl'.

copy_propval(**arg*)
Method to copy attributes form parent.

Example:

```
a=some_thesdk_class(self)
```

Attributes listed in proplist attribute of 'some_thesdkclass' are copied from self to a. Impemented by including following code at the end of __init__ method of every entity:

```
if len(arg)>=1:
    parent=arg[0]
    self.copy_propval(parent,self.proplist)
    self.parent =parent;
```

print_log(***kwargs*)
Method to print messages to 'logfile'

Parameters ****kwargs** –

type: str 'I' = Information 'D' = Debug. Enabled by setting the Debug-attribute of an instance to true 'W' = Warning 'E' = Error 'F' = Fatal, quits the execution 'O' = Obsolete, used for obsolition warnings.

msg: str The messge to be printed

timer()
Timer decorator

Print execution time of member functions of classes inheriting thesdk to the logfile.

The timer is applied by decorating the function to be timed with @thesdk.timer. For example, calling a function calculate_something() belonging to an example class calculator(thesdk), would print the following:

```
class calculator(thesdk):

    @thesdk.timer
    def calculate_something(self):
        # Time-consuming calculations here
        print(result)
        return result

>> calc = calculator()
>> result = calc.calculate_something()
42
10:25:17 INFO at calculator: Finished 'calculate_something' in 0.758 s.
>> print(result)
42
```

silence(*show_error=True*, *debug=False*)
Context manager to redirect stdout (and optional errors) to /dev/null. Useful for cleaning up verbose function outputs. The silencing can be bypassed by setting debug=True. Errors are let through by default, but error messages can be silenced also by setting show_error=False. Silences only Python outputs (external commands such as spectre can still write to stdout).

To silence (prevent printing to stdout) of a section of code:

```
print('This is printed normally')
with self.silence():
    print('This will not be printed')
print('This is again printed normally')
```

property par
True | False (default)

Property defines whether parallel run is intended or not

property queue
Property holding the queue for parallel run result

run_parallel(***kwargs*)
Run instances in parallel and collect results

Usage: Takes in a set of instances, runs a given method for them, and saves result data to the original instances.

Results are returned as a dictionary. The dictionary can include IOS, which are saved to IOS of the original instance. Otherwise non-IO key-value pairs are saved as members of self.extracts.Members for the original instance. This is an example of returning both IOS and other data (place at the end of your simulation method, e.g. run()):

```
if self.par:
    ret_dict = {'NF' : 25}
    ret_dict.update(self.IOS.Members) #Adds IOS to return dictionary
    self.queue.put(ret_dict)
```

Some simulator modules can populate the extracts-bundle with generic extracted parameters. To pass this dictionary to the original instance, following example can be used:

```
if self.par:
    # Combine IOS and extracts into one dictionary
    ret_dict = {**self.IOS.Members,**self.extracts.Members}
    self.queue.put(ret_dict)
```

Parameters ****kwargs** –

duts: list List of instances you want to simulate

method: str Method called for each instance (default: run)

property IOS
Bundle of IO's

Property holding the IOS

Example

self.IOS.Members['input_A']=IO()

Type Type

property extracts
Bundle

Bundle for holding the returned results from simulations that are not attributes or IOs.

Example:

```
self.extracts.Members['sndr']=60
```

property statepath
String

Path where the entity state is stored and where existing states are loaded from.

property save_state
Boolean (default False)

Save the entity state after simulation (including output data). Any stored state can be loaded using the matching state name passed to the *load_state* property. The state is saved to *savestatepath* by default.

property load_state
String (default '')

Feature for loading results of previous simulation. When calling run() with this property set, the simulation is not re-executed, but the entity state and output data will be read from the saved state. The string value should be the *runname* of the desired simulation.

Loading the most recent result automatically:

```
self.load_state = 'last'
# or
self.load_state = 'latest'
```

Loading a specific past result using the *runname*:

```
self.load_state = '20201002103638_tmpdbw11nr4'
```

List available results by providing any non-existent *runname*:

```
self.load_state = 'this_does_not_exist'
```

_write_state()
Write the entity state to a binary file.

This should be called after the simulation has finished. This will probably need to be overloaded by the specific simulator module for more specific result storing.

_read_state()
Read the entity state from a binary file.

class thesdk.IO(kwargs)**
TheSyDeKick IO class. Child of thesdk to utilize logging method.

The IOs of an entity must be defined as:

```
self.IOS=Bundle()
self.IOS.Members['a']=IO()
```

and referred to as:

```
self.IOS.Members['a'].Data
```

Parameters ****kwargs** –

Data: numpy_array, None Sets the Data attribute during the initialization

property _classfile

Abstract property of thesdk class. Defines the location of the classfile.

Define in every child class of thesdk as:

```
def _classfile(self):
    return os.path.dirname(os.path.realpath(__file__)) + "/"+__name__
```

property Data

Data value of this IO

class thesdk.**Bundle**

Bundle class of named things.

Members

Type dict, dict([])

__getattr__(*name*)

Access the attribute <name> directly Not tested.

Returns self.Members['name']

Return type type of dict member

new(***kwargs*)

Parameters ****kwargs** – name: str, optional val: str, optional

CHAPTER

ONE

IOFILE PACKAGE

Provides Common file-io related attributes and methods for TheSyDeKinck

This package defines the CSF IO-file format and methods for all supported simulators. Intention is to keep as much of the file io as general adn common as possible. The simulator specific file-io functions should be written in simulator specific file-io packages.

Initially written for Verilog file-io by Marko Kosunen, marko.kosunen@aalto.fi, Yue Dai, 2018

Generalized for common file-io by Marko Kosunen, marko.kosunen@aalto.fi, 2019.

class thesdk.iofile.**iofile**(*parent=None*, ***kwargs*)

Class to provide file IO for external simulators.

When created, adds an iofile object to the parents iofile_bundle attribute. Accessible as iofile_bundle.Members['name'].

Example

Initiated in parent as: _=iofile(self,name='foobar')

Parameters

- **parent** (*object*, *None*) – The parent object initializing the iofile instance. Default None
- ****kwargs** –

  name: str Name of the file. Appended with random string during the simulation.

  param: str, -g g_file The string defining the testbench parameter to be be passed to the simulator at command line.

adopt(*parent=None*, ***kwargs*)

Redefine the parent entity of the file. Afrects where the file is written during the simulation

property datatype

Type of the data. Controls the reading and writing of the data 'complex' | 'int' | 'scomplex' | 'sint' | object

int | sint: Values of the data are handled as unsigned or signed integers

complex | scomplex: It is assumed that each column of the Data represents a complex number with integer real and imaginary parts. This is typical for RTL simulations. when the data is read or written, the columns are treated as Real Imag pairs. When writing, columns are split to two, when reading, adjacent colums are merged

object: Values are read and written as defined by the 'dtype' parameter of the read method. Default is object.

property dir
Direction of the iofile, in | out

property file
Name of the IO file to be read or written.

property ionames
Names of the DUT signals represented by the colums of the data. Two elements per colums should be given, if the type is complex or scomplex.

property iotype
sample | event

sample IO is synchronized with a sampling clock. Each of the rows represents the data valua at known sample insatnce

event IO is asynchronous. The first colums of the file should be the time of the event

Type Type of the IO

read(***kwargs*)
Method to read the file

Parameters ****kwargs** –

datatype: str, self.datatype Controls if the data is read in as complex or real

dtype: str, 'object' The datatype of the actual file. Default is object, i.e data is first read to internal variable as string. This is a help parameter to give more control over reading.

remove()
Remove the file

write(***kwargs*)
Method to write the file

Sets the 'dir' attribute to 'in', because only input files are written.

Parameters ****kwargs** –

data: numpy_array, self.Data Data to be written.

datatype: str, self.Datatype Datatype of the data.

iotype: str, self.iotype IO type of the IO file.

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

t

INDEX

Symbols

A

B

C

D

E

F

G

H

I

L

M

N

P

Q

R

S

T

W