cocotbext uP

AIR FORCE RESEARCH LABORATORY

March 26, 2025

Jay Convertino

Contents

1 Usage **2**
1.1 Introduction 2
1.2 Dependencies 2
1.3 In a Simulation 2

2 Architecture **2**
2.1 Directory Guide 3

3 Simulation **4**
3.1 cocotb 4

4 Code Documentation **5**
4.1 init 6
4.2 monitor 7
4.3 driver 9
4.4 absbus 13
4.5 busbase 16
4.6 test extension python 21
4.7 test extension verilog 22

1 Usage

1.1 Introduction

Cocotb extension to test uP bus master, and slave devices.

1.2 Dependencies

The following are the dependencies of the cores.

- iverilog (simulation)
- cocotb (simulation)
- cocotb-bus (simulation)

1.3 In a Simulation

Below is a simple example for reading and writing data from register zero in the cocotb extension.

```
master  = upMaster(dut, "apb", dut.clk, dut.rstn)
slave = upEchoSlave(dut, "apb", dut.clk, dut.rstn)

await master.write(0, 0xAAAAAAAA)

rx_data = await master.read(0)

assert 0xAAAAAAAA == rx_data, "RECEIVED␣DATA␣DOES␣NOT␣
    ↪ MATCH"
```

2 Architecture

Please see 4 for more information.

upMaster tests uP slave devices by executing read/write requests from the python test bench.

upEchoSlave provides a simple slave that will echo all register writes back over read when requested.

upMonitor tests to make sure signals are proper. Simple core at the moment, only checks for 0 at rest and if the wack/rack is correct per wreq/rreq.

2.1 Directory Guide

Below highlights important folders from the root of the directory.

1. **docs** Contains all documentation related to this project.
    - **manual** Contains user manual and github page that are generated from the latex sources.
2. **cocotbext** Contains source files for the extension
    - **up.ad** Contains source files for the Analog Devices uP version of the bus.
3. **tests** Contains test files for cocotb

3 Simulation

A simulation for testing the cores can be run to verify operation.

3.1 cocotb

To use the cocotb tests you must install the following python libraries.

```
$ pip install cocotb
$ pip install −e .
```

Then you must enter the tests folder and enter the folder of the type you wish to test. From there you may execute the following command which will kick off the test.

```
$ make
```

4 Code Documentation

Natural docs is used to generate documentation for this project. The next lists the following sections.

- **init** Python init code.
- **monitor** Contains bus monitor code.
- **driver** Contains bus driver code.
- **absbus** Contains bus abstraction for monitor, and driver code.
- **busbase** Contains bus base for threads and read/write methods.
- **cocotb test** Python TestFactory code.
- **cocotb verilog test wrapper** Verilog wrapper module.

__init__.py

AUTHORS

JAY CONVERTINO

DATES

2025/03/26

INFORMATION

Brief

uP define for packages

License MIT

Copyright 2025 Jay Convertino

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

""""

Copyright (c) 2020 Alex Forencich

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

""""

monitor.py

AUTHORS

JAY CONVERTINO

DATES

2025/03/11

INFORMATION

Brief

Monitor for uP

License MIT

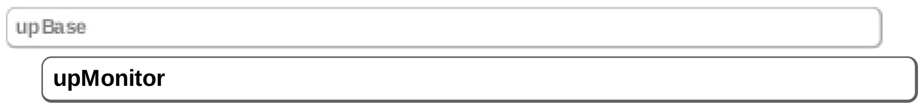
Copyright 2025 Jay Convertino

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

upMonitor



Check signals to make sure they are applied properly.

FUNCTIONS

__init__

```
def __init__(
self,
entity,
name,
clock,
resetn,
                                                                          *
args,
                                                                         **
kwargs
)
```

Setup defaults and call base class constructor.

_check_type

```
def _check_type(
self,
trans
)
```

Check and make sure we are only sending uptrans, this is only here to satisify the need to have it.

_run_write

```
async def _run_write(
self
)
```

Coroutine for writing uP bus

_run_read

```
async def _run_read(
self
)
```

Coroutine for reading uP bus

driver.py

AUTHORS

JAY CONVERTINO

DATES

2025/03/11

INFORMATION

Brief

Bus Driver for Analog Devices uP

License MIT

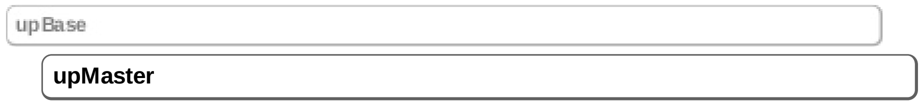
Copyright 2025 Jay Convertino

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

upMaster



Drive slave devices over the uP bus

FUNCTIONS

__init__

```
def __init__(
self,
entity,
name,
clock,
resetn,
*
args,
**
kwargs
)
```

Setup defaults and call base class constructor.

read

```
async def read(
self,
address
)
```

Read from a address and return data

write

```
async def write(
self,
address,
data
)
```

Write to a address some data

_check_type

```
def _check_type(
self,
trans
)
```

Check and make sure we are only sending 2 bytes at a time and that it is a bytes/bytearray

_run_write

```
async def _run_write(
self
)
```

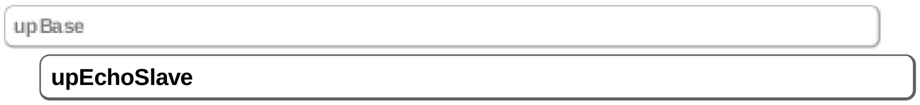
method for write thread

_run_read

```
async def _run_read(
self
)
```

method for read thread

upEchoSlave



Respond to master reads and write by returning data, simple echo core.

FUNCTIONS

__init__

```
def __init__(
self,
entity,
name,
clock,
resetn,
numreg
=
256,
                                                                  *
args,
                                                                  **
kwargs
)
```

Setup defaults and call base class constructor.

_check_type

```
def _check_type(
self,
trans
)
```

Check and make sure we are only sending a type of uptrans.

_run_write

```
async def _run_write(
self
)
```

method for write thread

_run_read

```
async def _run_read(
self
)
```

method for read thread

absbus.py

AUTHORS

JAY CONVERTINO

DATES

2025/03/26

INFORMATION

Brief

abstraction of the Analog Devices uP bus

License MIT

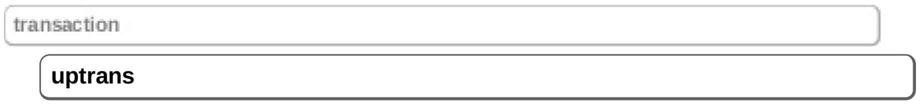
Copyright 2025 Jay Convertino

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

uptrans



create an object that associates a data member and address for operation.

upState

upState

An enum class that provides the current state and will change states per spec.

upBase

- busbase
  - **upBase**
    - upEchoSlave
    - upMaster
    - upMonitor

abstract base class that defines uP signals

VARIABLES

_signals

```
_signals
```

List of signals that are required

FUNCTIONS

__init__

```
def __init__(
self,
entity,
name,
clock,
resetn,
                                                                    *
args,
                                                                   **
kwargs
)
```

Setup defaults and call base class constructor.

_restart_rw

```
def _restart_rw(
self
)
```

kill and restart _run thread.

_run

```
async def _run(
self
)
```

_run thread that deals with read and write.

_run_read

```
async def _run_read(
self
)
```

Abstract method for read thread

_run_write

```
async def _run_write(
self
)
```

Abstract method for write thread

busbase.py

AUTHORS

JAY CONVERTINO

DATES

2025/03/11

INFORMATION

Brief

classic bus define for packages

License MIT

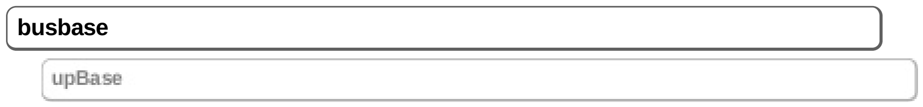
Copyright 2025 Jay Convertino

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

busbase



A busbase to transmit test routine.

FUNCTIONS

__init__

```
def __init__(
self,
entity
:
SimHandleBase,
name
:
Optional[str],
clock
:
SimHandleBase,
                                                                    *
args
:
Any,
                                                                   **
kwargs
:
Any
)
```

Initialize the object

VARIABLES

wqueue

```
self.wqueue
```

Queue to store write requests

qqueue

```
self.qqueue
```

Queue to store read requests

rqueue

```
self.rqueue
```

Queue to store result of read requests

self.**_idle**

```
self._idle
```

Event trigger for cocotb

self.**_run_cr**

```
self._run_cr
```

Thread instance of _run method

FUNCTIONS

_restart

```
def _restart(
self
)
```

kill and restart _run thread.

write_count

```
def write_count(
self
)
```

How many items in the write queue

read_count

```
def read_count(
self
)
```

How many items in the read queue

write_empty

```
def write_empty(
self
)
```

Is the quene empty?

read_empty

```
def read_empty(
self
)
```

Is the quene empty?self.bus.penable.value

write_clear

```
def write_clear(
self
)
```

Remove all write items from queue

read_clear

```
def read_clear(
self
)
```

Remove all read items from queue

wait

```
async def wait(
self
)
```

Wait for the run thread to become idle.

idle

```
def idle(
self
)
```

Are all the queues empty and the _run is not active processing data.

write_trans

```
async def write_trans(
self,
trans
:
transaction
)
```

Write transaction to send to write queue

read_trans

```
async def read_trans(
self,
trans
:
transaction
)
```

Read bus and output and tranaction.

_write

```
async def _write(
self,
trans
:
transaction
)
```

Write data one element at a time

_read

```
async def _read(
self,
trans
:
transaction
)
```

Read dat one element at a time

_check_type

```
def _check_type(
self,
trans
)
```

Check and make sure we are only sending the correct transaction type

_run

```
async def _run(
self
)
```

Virtual method for _run thread that deals with read and write queues.

TB

TB

Create the device under test which is the master/slave.

FUNCTIONS

run_test

```
async def run_test(
dut,
payload_data
 =
None
)
```

Tests the source/sink for valid transmission of data.

incrementing_payload

```
def incrementing_payload()
```

Generate a list of ints that increment from 0 to 2^8

test

```
def test(
request
)
```

Main cocotb function that specifies how to put the test together.

test.v

AUTHORS

JAY CONVERTINO

DATES

2025/03/17

INFORMATION

Brief

Test bench for analog devices uP using cocotb

License MIT

Copyright 2025 Jay Convertino

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

test

```
 module test #(
 parameter
 ADDRESS_WIDTH
  =
 32,
 parameter
 BUS_WIDTH
  =
 4
) ( input clk, input rstn, inout up_rreq, inout up_rack, inout [ADDRESS_WID
```

Test bench loop for up

Parameters

ADDRESS_WIDTH parameter	Width of the uP address port, max 32 bit.
BUS_WIDTH parameter	Width of the uP bus data port.

Ports

clk	Clock for all devices in the core
rstn	Negative reset
up_rreq	uP bus read request
up_rack	uP bus read ack
up_raddr	uP bus read address
up_rdata	uP bus read data
up_wreq	uP bus write request
up_wack	uP bus write ack
up_waddr	uP bus write address
up_wdata	uP bus write data