cocotbext FIFO

AIR FORCE RESEARCH LABORATORY

March 31, 2025

Jay Convertino

Contents

1 Usage

1.1 Introduction

Cocotb extension to test FIFO based 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.

```
source  = xilinxFIFOsource(dut, "wr", dut.wr_clk, dut.
    ↪ wr_rstn, dut.FWFT.value != 0)
sink = xilinxFIFOsink(dut, "rd", dut.rd_clk, dut.rd_rstn,
    ↪  dut.FWFT.value != 0)

await source.write(0, 0xAAAAAAAA)

rx_data = await sink.read(0)

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

2 Architecture

Please see 4 for more information.
xilinxFIFOsource write to Xilinx FIFOs.
xilinxFIFOsink read from Xilinx FIFOs.
xilinxFIFOmonitor tests to make sure signals are proper. N/A

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
   - **fifo.xilinx** Contains source files for the Xilinx FIFO.
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 tests folder. 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

2025/03/27

INFORMATION

Brief

xilinx fifo 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 APB3

License MIT

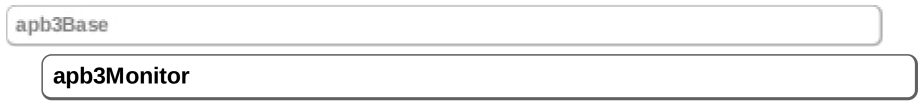
Copyright 2025 Jay Convertino

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.

apb3Monitor



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 apb3trans, this is only here to satisify the need to have it.

_run

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

_run thread that deals with checking signals, simple check for now.

driver.py

AUTHORS

JAY CONVERTINO

DATES

2025/03/27

INFORMATION

Brief

Bus Driver for Xilinx FIFO

License MIT

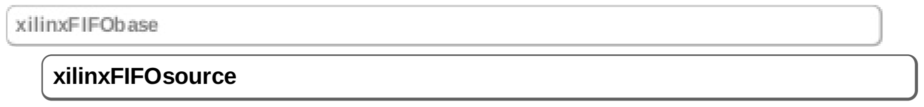
Copyright 2025 Jay Convertino

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.

xilinxFIFOsource



Drive xilinx FIFO write interfaces

VARIABLES

_signals

```
_signals
```

List of signals that are required

_optional_signals

```
_optional_signals
```

List of optional signals, these will never be required but will be used if found.

FUNCTIONS

__init__

```
def __init__(
self,
entity,
name,
clock,
resetn,
fwft
=
False,
ack
=
False,
*
args,
**
kwargs
)
```

Setup defaults and call base class constructor.

write

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

Write to a address some data

_check_type

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

Check and make sure we are only sending xilinxFIFOtrans

_run

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

_run thread that deals with read and write queues.

xilinxFIFOsink

xilinxFIFObase

xilinxFIFOsink

Drive xilinx FIFO read interfaces

VARIABLES

_signals

```
_signals
```

List of signals that are required

_optional_signals

```
_optional_signals
```

List of optional signals, these will never be required but will be used if found.

FUNCTIONS

__init__

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

Setup defaults and call base class constructor.

write

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

Write to a address some data

read

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

Read from a address and return data

_check_type

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

Check and make sure we are only sending xilinxFIFOtrans

_run

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

_run thread that deals with read and write queues.

absbus.py

AUTHORS

JAY CONVERTINO

DATES

2025/03/27

INFORMATION

Brief

abstraction of the xilinx fifo 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.

xilinxFIFOtrans

transaction
xilinxFIFOtrans

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

xilinxFIFOsourceState

xilinxFIFOsourceState

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

xilinxFIFOsinkState

enum.IntEnum

xilinxFIFOsinkState

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

xilinxFIFObase

busbase

xilinxFIFObase

xilinxFIFOsink

xilinxFIFOsource

abstract base class that defines Xilinx FIFO signals

FUNCTIONS

__init__

```
 def __init__(
 self,
 entity,
 name,
 clock,
 resetn,
 fwft
 =
 False,
 ack
 =
 False,
                                                                      *
 args,
                                                                     **
 kwargs
)
```

Setup defaults and call base class constructor.

busbase.py

AUTHORS

JAY CONVERTINO

DATES

2025/03/11

INFORMATION

Brief

classic bus define for packages

License MIT

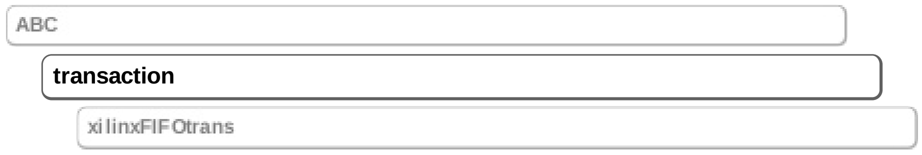
Copyright 2025 Jay Convertino

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.

transaction



Abstract class for transaction types

noSignal

noSignal

Class to use when a signal does not exist

busbase

busbase

xilinxFIFObase

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

_queue_read

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

Setup queue for read requests

_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 xilinx fifo 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
FIFO_DEPTH
 =
8,
parameter
BYTE_WIDTH
 =
4,
parameter
FWFT
 =
1
) ( input rd_clk, input rd_rstn, inout rd_en, inout rd_valid, inout [(BYTE_W
```

Test bench loop for xilinx fifo

Parameters

FIFO_DEPTH parameter	Depth of the fifo, must be a power of two number(divisable aka 256 = 2^8). Any non-power of two will be rounded up to the next closest.
BYTE_WIDTH parameter	How many bytes wide the data in/out will be.
FWFT parameter	1 for first word fall through mode. 0 for normal.

Ports

rd_clk	Clock for read data
rd_rstn	Negative edge reset for read.
rd_en	Active high enable of read interface.
rd_valid	Active high output that the data is valid.
rd_data	Output data
rd_empty	Active high output when read is empty.
wr_clk	Clock for write data
wr_rstn	Negative edge reset for write
wr_en	Active high enable of write interface.
wr_ack	Active high when enabled, that data write has been done.
wr_data	Input data
wr_full	Active high output that the FIFO is full.