COCOTBEXT_mil-std-1553

March 7, 2025

Jay Convertino

Contents

1 Usage

1.1 Introduction

Standard AXIS FIFO with multiple options. The FIFO uses a AXIS interface for data in and out. It also emulates the Xilinx AXIS FIFO bugs and all. This is NOT dependent on Xilinx FPGA's and can be used on any FPGA supporting the Verilog block ram style primitive.

1.2 Dependencies

The following are the dependencies of the cores.

- iverilog (simulation)
- cocotb (simulation)

1.3 In a Project

Simply use this core between a sink and source devices. This buffer data from one bus to another. Check the code to see if others will work correctly.

2 Architecture

This AXIS FIFO is made for two modules. They are the FIFO, and AXIS FIFO control. The combination of these two provide the AXIS FIFO module. Having it made this way allows for future modules to be customized and brought in to change the FIFO's behavior. The current modules emulate the Xilinx AXIS FIFO IP core availble in Vivado 2018 and up.

AXIS FIFO control is the heart of the core when it comes to how it responds. The logic in the core is designed to emulate the Xilinx AXIS FIFO IP.

Please see 5 for more information.

3 Building

The AXIS FIFO core is written in Verilog 2001. They should synthesize in any modern FPGA software. The core comes as a fusesoc packaged core and can be included in any other core. Be sure to make sure you have meet the dependencies listed in the previous section.

3.1 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. **src** Contains source files for the core
3. **tb** Contains test bench files for iverilog and cocotb

4 Simulation

There are a few different simulations that can be run for this core. All currently use iVerilog (icarus) to run. The first is iverilog, which uses verilog only for the simulations. The other is cocotb. This does a unit test approach to the testing and gives a list of tests that pass or fail.

4.1 iverilog

All simulation targets that do NOT have cocotb in the name use a verilog test bench with verilog stimulus components. These all read in a file and then write a file that has been processed by the FIFO. Then the input and output file are compared with a MD5 sum to check that they match. If they do not match then the test has failed. All of these tests provide fst output files for viewing the waveform in the there target build folder.

4.2 cocotb

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

```
$ pip install cocotb
$ pip install cocotbext-axi
```

Then you must use the cocotb sim target. In this case it is sim_cocotb. This target can be run with various bus and fifo parameters.

```
$ fusesoc run --target sim_cocotb AFRL:buffer:axis_fifo
    ↪ :1.0.0 --BUS_WIDTH=8 --FIFO_DEPTH=32
```

The following is an example command to run through various parameters without typing them one by one.

```
$ for i in {1..32}; do sleep 5; export RY=$(($RANDOM
    ↪ %32+1)); fusesoc run --target sim_cocotb AFRL:
    ↪ buffer:axis_fifo:1.0.0 --BUS_WIDTH=$i --
    ↪ FIFO_DEPTH=$RY; echo "BUS␣WIDTH:" $i "FIFO␣DEPTH:
    ↪ " $RY; done
```

5 Code Documentation

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

- **AXIS_FIFO** will buffer data from input to output.
- **AXIS_FIFO_CONTROL** emulates the Xilinx FIFO IP interface and its behavior.
- **tb_axis** Verilog test bench.
- **tb_cocotb verilog** Verilog test bench base for cocotb.
- **tb_cocotb python** cocotb unit test functions.

__init__.py

AUTHORS

JAY CONVERTINO

DATES

2025/03/06

INFORMATION

Brief

MIL-STD-1553 define for packages license: 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.

"""

mil_std_1553.py

AUTHORS

JAY CONVERTINO

DATES

2025/03/06

INFORMATION

Brief

MIL-STD-1553 cocotb

License MIT

MILSTD1553Source

MILSTD1553Source

A mil-std-1553 transmit test routine.

FUNCTIONS

__init__

```
def __init__(
self,
data,
*
args,
**
kwargs
)
```

Initialize the object

VARIABLES

self._data

```
self._data
```

Set internal data connection to 1553 differential bus

self._base_delay

```
self._base_delay
```

1 MHz is 1000 nano seconds need half that due to manchester encoding method

self._idle

```
self._idle
```

Event trigger for cocotb

self._data

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_cmd

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

Write data to send that uses the command sync

write_data

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

Write data to send that uses the data sync

write_nowait_cmd

```
def write_nowait_cmd(
self,
data
)
```

Write data to send that uses command sync but do not wait after writting.

write_nowait_data

```
def write_nowait_data(
self,
data
)
```

Write data to send that uses data sync but do not wait after writting.

count

```
def count(
self
)
```

How many items in the queue

empty

```
def empty(
self
)
```

Is the quene empty?

idle

```
def idle(
self
)
```

Is the queue empty and the _run is not active processing data.

clear

```
def clear(
self
)
```

Remove all items from queue

_check_type

```
def _check_type(
self,
data
)
```

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

_cmd_sync

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

Generate a command sync on the diff output

_data_sync

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

Generate a data sync on the diff output

wait

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

Wait for the run thread to become idle.

_run

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

Thread that processing queue and outputs data in mil-std-1553 format.

MILSTD1553Sink

MILSTD1553Sink

A mil-std-1553 transmit test routine.

FUNCTIONS

__init__

```
def __init__(
self,
data,
*
args,
**
kwargs
)
```

Initialize the object

VARIABLES

self._data

```
self._data
```

Set internal data connection to 1553 differential bus

self._base_delay

```
self._base_delay
```

1 MHz is 1000 nano seconds need half that due to manchester decoding method

self._base_delay

```
self._base_delay_half
```

1 MHz is 1000 nano seconds need half of half that due to manchester decoding method

_cmd_sync

```
self._cmd_sync
```

command sync array value

_data_sync

```
self._data_sync
```

data sync array value

self._run_cr

```
self._run_cr
```

Thread instance of _run method

FUNCTIONS

_restart

```
def _restart(
self
)
```

Kill and restart run function

read_cmd

```
async def read_cmd(
self
)
```

Read any data that was identified with a command sync

read_nowait_cmd

```
def read_nowait_cmd(
self
)
```

Read any data that was identified with a command sync, and do not wait for data to become available.

read_data

```
async def read_data(
self
)
```

Read any data that was identified with a data sync.

read_nowait_data

```
def read_nowait_data(
self
)
```

Read any data that was identified with a data sync, and do not wait for data to become available.

count_cmd

```
def count_cmd(
self
)
```

How many elements are in the command queue?

count_data

```
def count_data(
self
)
```

How many elements are in the data queue?

empty_cmd

```
def empty_cmd(
self
)
```

Is the queue empty?

empty_data

```
def empty_data(
self
)
```

Is the queue empty?

idle

```
def idle(
self
)
```

Is _run waiting to process data?

clear_cmd

```
def clear_cmd(
self
)
```

Clear the command queue

clear_data

```
def clear_data(
self
)
```

Clear the data queue

wait_cmd

```
async def wait_cmd(
self,
timeout
 =
0,
timeout_unit
 =
'nsreg_data'
)
```

Wait for command data

wait_data

```
async def wait_data(
self,
timeout
 =
0,
timeout_unit
 =
'nsreg_data'
)
```

Wait for data data.

_run

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

Thread that takes input data in mil-std-1553 format and puts it in the proper command or data queue.

TB

TB

Create the device under test which is the source/sink.

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 2^16

random_payload

```
def random_payload()
```

Generate a list of random ints 2^16 in the range of 0 to 2^16

test_mil_std_1553

```
def test_mil_std_1553(
request
)
```

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

test_mil_std_1553.v

AUTHORS

JAY CONVERTINO

DATES

2025/03/06

INFORMATION

Brief

Test bench wrapper for mil-std-1553 cocotb

License MIT

test_mil_std_1553

```
module test_mil_std_1553 (
                                                                    inout [
1:0]
data
)
```

Simple loop of MIL-STD-1553 source/sink

Ports

data inout[1: 0]	Differential mil-std-1553 data	