

# AXIS\_MOVING\_AVERAGE



April 25, 2025

Jay Convertino

# Contents

<b>1 Usage</b>	<b>2</b>
1.1 Introduction . . . . .	2
1.2 Dependencies . . . . .	2
1.2.1 fusesoc_info Depenecies . . . . .	2
1.3 In a Project . . . . .	2
<b>2 Architecture</b>	<b>2</b>
<b>3 Building</b>	<b>3</b>
3.1 fusesoc . . . . .	3
3.2 Source Files . . . . .	3
3.2.1 fusesoc_info File List . . . . .	3
3.3 Targets . . . . .	4
3.3.1 fusesoc_info Targets . . . . .	4
3.4 Directory Guide . . . . .	4
<b>4 Simulation</b>	<b>5</b>
4.1 iverilog . . . . .	5
4.2 cocotb . . . . .	5
<b>5 Code Documentation</b>	<b>6</b>
5.1 axis_moving_average . . . . .	7
5.2 tb_axis . . . . .	9
5.3 tb_cocotb verilog . . . . .	11
5.4 tb_cocotb python . . . . .	13

# 1 Usage

## 1.1 Introduction

This core provides the AXIS Moving Average function. This implements the moving average algorithm in an efficient way for FPGAs. Efficient since it uses powers of two to calculate its weights. This formula implemented is where  $n$  is constrained to a power of two and  $X$  is an unsigned number. This also works as a low pass filter or a smoothing algorithm.

## 1.2 Dependencies

The following are the dependencies of the cores.

- fusesoc 2.X
- iverilog (simulation)
- cocotb (simulation)

### 1.2.1 fusesoc\_info Dependencies

- dep
  - AFRL:utility:helper:1.0.0
- dep\_tb
  - AFRL:simulation:axis\_stimulator
  - AFRL:simulation:clock\_stimulator
  - AFRL:utility:sim\_helper

## 1.3 In a Project

Simply use this core between a sink and source AXIS devices. This has been tested with unsigned data types. Check the code to see if others will work correctly.

# 2 Architecture

The only module is the `axis_moving_average` module. This is a continuous output rather than a wait and release on range. It is listed below.

- **axis\_moving\_average** Implement moving average algorithm (see core for documentation).

Internally this uses the simple moving average algorithm for the output average. The weight(n) is rounded up to the nearest power of two.

$$SMA = \sum_{i=0}^{\log_2 n} \frac{D_0 + \dots + D_n}{n} \quad (1)$$

This core only uses a combinatorial method to divide the accumulator. Since all weights are powers of two this is done with a part select based on bit position.

The always block has the following steps.

1. If there is valid data, sum the new data into the accumulator and remove the top element in the buffer from the accumulator.
2. Insert the new element into the buffer.
3. Shift the buffer to so that old elements at the top of the buffer are shifted out.

Please see ?? for more information.

## 3 Building

The AXIS Moving Average 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 met the dependencies listed in the previous section.

### 3.1 fusesoc

Fusesoc is a system for building FPGA software without relying on the internal project management of the tool. Avoiding vendor lock in to Vivado or Quartus. These cores, when included in a project, can be easily integrated and targets created based upon the end developer needs. The core by itself is not a part of a system and should be integrated into a fusesoc based system. Simulations are setup to use fusesoc and are a part of its targets.

### 3.2 Source Files

#### 3.2.1 fusesoc\_info File List

- src
  - src/axis\_moving\_average.v
- tb\_cocotb

- 'tb/tb\_cocotb.py': 'file\_type': 'user', 'copyto': '.'
- 'tb/tb\_cocotb.v': 'file\_type': 'verilogSource'
- tb
  - 'tb/tb\_axis.v': 'file\_type': 'verilogSource'

### 3.3 Targets

#### 3.3.1 fusesoc\_info Targets

- default
 

Info: Default for IP intergration.
- sim
 

Info: Default for simulation using icarus.
- sim\_cocotb
 

Info: Cocotb unit tests

### 3.4 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 data width converter. 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:math:
  ↳ axis_moving_average:1.0.1 --BUS_WIDTH=8 --WEIGHT
  ↳ =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=$((RANDOM*2
  ↳ %32)); fusesoc run --target sim_cocotb AFRL:math:
  ↳ axis_moving_average:1.0.1 --BUS_WIDTH=$i --WEIGHT
  ↳ =$RY; echo "BUS_WIDTH:" $i "WEIGHT:" $RY; done
```

## 5 Code Documentation

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

- **axis\_moving\_average** AXIS moving average core.
- **tb\_axis** Verilog test bench.
- **tb\_cocotb verilog** Verilog test bench base for cocotb.
- **tb\_cocotb python** cocotb unit test functions.

# axis\_moving\_average.v

---

## AUTHORS

---

JAY CONVERTINO

---

## DATES

---

2023/02/01

---

## INFORMATION

---

### Brief

---

AXIS moving average for unsigned numbers.

### License MIT

---

Copyright 2023 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.

## axis\_moving\_average

---

```
module axis_moving_average #(
    parameter
    BUS_WIDTH
    =
    1,
    parameter
    WEIGHT
    =
    1
) ( input aclk, input arstn, output [8*BUS_WIDTH-1:0] m_axis_tdata, output r
```

AXIS moving average for unsigned numbers.



## Parameters

<b>BUS_WIDTH</b> parameter	Width of the BUS in bytes.
<b>WEIGHT</b> parameter	How many elements, rounded to a power of two, to accumulate.

## Ports

<b>ack</b>	Clock for AXIS
<b>arstn</b>	Negative reset for AXIS
<b>s_axis_tdata</b>	Input data
<b>s_axis_tvalid</b>	When set active high the input data is valid
<b>s_axis_tready</b>	When active high the device is ready for input data.
<b>m_axis_tdata</b>	Output data
<b>m_axis_tvalid</b>	When active high the output data is valid
<b>m_axis_tready</b>	When set active high the output device is ready for data.

## VARIABLES

---

### m\_axis\_tdata

---

```
assign m_axis_tdata = r_accumulator[(  
c_WEIGHT_POWER  
1  
):c_WEIGHT_POWER] 8*BUS_WIDTH+
```

Trim and shift data to get amount, this is the divide out.

### m\_axis\_tvalid

---

```
assign m_axis_tvalid = r_always_valid
```

Single clock edge valid

### s\_axis\_tready

---

```
assign s_axis_tready = m_axis_tready
```

We are ready if the destination is ready

## tb\_axis.v

---

### AUTHORS

---

JAY CONVERTINO

---

### DATES

---

2024/12/11

---

### INFORMATION

---

#### Brief

---

Test bench for axis\_moving\_average using axis stim and clock stim.

#### License MIT

---

Copyright 2024 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.

## tb\_axis

---

```
module tb_axis
```

Test bench for axis\_moving\_average. This will run a file through the system and write its output. These can then be compared to check for errors. If the files are identical, no errors. A FST file will be written.

### INSTANTIATED MODULES

---

#### clk\_stim

---

```

clk_stimulus #(
    CLOCKS(1),
    CLOCK_BASE(1000000),
    CLOCK_INC(1000),
    RESETS(1),
    RESET_BASE(2000),
    RESET_INC(100)
) clk_stim ( .clkv(tb_stim_clk), .rstnv(tb_stim_rstn), .rstv() )

```

Generate a 50/50 duty cycle set of clocks and reset.

## slave\_axis\_stim

---

```

slave_axis_stimulus #(
    BUS_WIDTH(BUS_WIDTH),
    USER_WIDTH(USER_WIDTH),
    DEST_WIDTH(DEST_WIDTH),
    FILE("random.bin")
) slave_axis_stim ( .m_axis_aclk(tb_stim_clk), .m_axis_arstn(tb_stim_rstn),

```

Device under test SLAVE stimulus module.

## dut

---

```

axis_moving_average #(
    BUS_WIDTH(BUS_WIDTH),
    WEIGHT(8)
) dut ( .aclk(tb_stim_clk), .arstn(tb_stim_rstn), .m_axis_tdata(tb_dut_data)

```

Device under test, axis\_moving\_average

## slave\_axis\_stim

---

Device under test SLAVE stimulus module.

# tb\_cocotb.v

---

## AUTHORS

---

JAY CONVERTINO

---

## DATES

---

2024/12/11

---

## INFORMATION

---

### Brief

---

Test bench wrapper for cocotb

### License MIT

---

Copyright 2024 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.

## tb\_cocotb

---

```
module tb_cocotb #(
  parameter
  BUS_WIDTH
  =
  1,
  parameter
  WEIGHT
  =
  1
) ( input aclk, input arstn, output [(BUS_WIDTH*8)-1:0] m_axis_tdata, output
```

Test bench for axis moving average. This will run a file through the system and write its output. These can then be compared to check for errors. If the files are identical, no errors. A FST file will be written.

## Parameters

<b>BUS_WIDTH</b> parameter	Width of the bus input/output
<b>WEIGHT</b> parameter	Divisor for moving average, rounded to the highest power of two.

## Ports

<b>aclk</b>	Clock for AXIS
<b>arstn</b>	Negative reset for AXIS
<b>m_axis_tdata</b>	Output data
<b>m_axis_tvalid</b>	When active high the output data is valid
<b>m_axis_tready</b>	When set active high the output device is ready for data.
<b>s_axis_tdata</b>	Input data
<b>s_axis_tvalid</b>	When set active high the input data is valid
<b>s_axis_tready</b>	When active high the device is ready for input data.

## INSTANTIATED MODULES

---

### dut

---

```
axis_moving_average #(
    BUS_WIDTH(BUS_WIDTH),
    WEIGHT(WEIGHT)
) dut ( .aclk(aclk), .arstn(arstn), .m_axis_tdata(m_axis_tdata), .m_axis_tvalid(m_axis_tvalid), .s_axis_tdata(s_axis_tdata), .s_axis_tvalid(s_axis_tvalid), .s_axis_tready(s_axis_tready) )
```

Device under test, axis\_moving\_average

# tb\_cocotb.py

---

## AUTHORS

---

JAY CONVERTINO

---

## DATES

---

2024/12/09

---

## INFORMATION

---

### Brief

---

Cocotb test bench

### License MIT

---

Copyright 2024 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.

## FUNCTIONS

---

### moving\_average

---

```
async def moving_average(  
    dut  
)
```

Emulate verilog moving average function for unsigned numbers only, this is a coroutine that runs at the same time as the main.

### Parameters

**dut**     device under test from cocotb test.

## random\_bool

---

```
def random_bool()
```

Return a infinite cycle of random bools

Returns: List

## start\_clock

---

```
def start_clock(  
    dut  
)
```

Start the simulation clock generator.

### Parameters

**dut** Device under test passed from cocotb test function

## reset\_dut

---

```
async def reset_dut(  
    dut  
)
```

Cocotb coroutine for resets, used with await to make sure system is reset.

### Parameters

**dut** Device under test passed from cocotb.

## conversion\_test

---

```
@cocotb.test()  
async def conversion_test(  
    dut  
)
```

Coroutine that is identified as a test routine. This routine tests for conversion based on current input to output size conversion.

### Parameters

**dut** Device under test passed from cocotb.

## conversion\_test\_rand\_ready

---

```
@cocotb.test()  
async def conversion_test_rand_ready(  
    dut  
)
```

---

Coroutine that is identified as a test routine. This routine tests for conversion based on current input to output size conversion.

#### Parameters

**dut**     Device under test passed from cocotb.

### in\_reset

---

```
@cocotb.test()
async def in_reset(
    dut
)
```

Coroutine that is identified as a test routine. This routine tests if device stays in unready state when in reset.

#### Parameters

**dut**     Device under test passed from cocotb.

### no\_clock

---

```
@cocotb.test()
async def no_clock(
    dut
)
```

Coroutine that is identified as a test routine. This routine tests if no ready when clock is lost and device is left in reset.

#### Parameters

**dut**     Device under test passed from cocotb.