

# AXIS SPI MASTER



May 21, 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>3</b>
2.1 Ports . . . . .	3
2.2 Waveforms . . . . .	3
<b>3 Building</b>	<b>5</b>
3.1 fusesoc . . . . .	5
3.2 Source Files . . . . .	5
3.2.1 fusesoc_info File List . . . . .	5
3.3 Targets . . . . .	6
3.3.1 fusesoc_info Targets . . . . .	6
3.4 Directory Guide . . . . .	6
<b>4 Simulation</b>	<b>7</b>
4.1 iverilog . . . . .	7
4.2 cocotb . . . . .	7
<b>5 Module Documentation</b>	<b>8</b>
5.1 axis_spi_master . . . . .	9
5.2 tb_cocotb python . . . . .	12
5.3 tb_cocotb verilog . . . . .	15
5.4 tb_spi verilog . . . . .	18

# 1 Usage

## 1.1 Introduction

The intent of this core is to provide a base AXIS to SPI Master interface. It is capable of back to back transfers with zero wait time. The data can be output at any rate up to half the input clock. The core SPI clock is generated for external use only and should NOT be routed into any logic. This device also does the chip selection based on the current activity of the core. CPOL/CPHA can be altered at anytime.

## 1.2 Dependencies

The following are the dependencies of the cores.

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

### 1.2.1 fusesoc\_info Depenecies

- dep
  - AFRL:clock:mod\_clock\_ena\_gen:1.1.1
  - AFRL:utility:helper:1.0.0
  - AFRL:simple:piso:1.0.0
  - AFRL:simple:sipo:1.0.0
- dep\_tb
  - AFRL:simulation:axis\_stimulator
  - AFRL:utility:sim\_helper
  - AFRL:simulation:clock\_stimulator

## 1.3 In a Project

This core connects a SPI to the AXIS bus. Meaning this is a streaming device only. Connect the MOSI/MISO to the SPI device in question and connect the AXIS to its intended endpoints.

## 2 Architecture

The core for this contains the following:

- **axis\_spi** Interface with SPI to AXIS interface.
- **mod\_clock\_ena\_gen** Generate an enable used to sample data for piso/sipo.
- **piso** Take parallel data and output it in serial.
- **sipo** Take serial data and output it in parallel.

The main core is made to interface a AXIS bus to the SPI bus. This is done using the SIPO and PISO cores to change from serial to parallel data streams. In addition mod clock enable gen cores create the negative and positive enables based upon the input clock and set rate to sample the data. This is then glued together in the core with some logic to output the appropriate SPI signals, including a generated clock. This generated clock is created by the mod clock gen enables only and is NOT used to clock any internal signals. Use only as a output clock. The core allows for any rate to be used up to half the input clock. The clock phase and polarity can be changed on the fly at anytime. All word transfers are the size of the AXIS bus. If multiple byte transfers of varing sizes are needed. It is recommened to set this to one byte width for the AXIS data bus and do back to back transfers for the number needed. Basically having data available to the core as soon as it can get it means there will be no gap in the spi output.

### 2.1 Ports

A port list is availabe, with specific signal information, in the 5. The generalized idea is a AXIS slave input for MOSI data. A AXIS master output for MISO data, and a master SPI interface with sclk, mosi, miso, and ss\_n signals. The dcount outputs give insight into the status of the in/out data bits of the core.

### 2.2 Waveforms

The idealized simulation waveforms are shown below. The values reflect the results of using the icarus backend with GTKwaveform view tools.

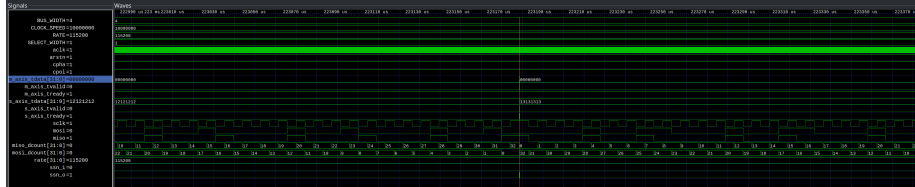
Back to back transfers will naturally occur if data is available in time for the next ready. This allows for zero intertransmission gaps.

[illegible]

### Figure 2: CPOL = 0 : CPHA = 0

[illegible][illegible]

Figure 5: CPOL = 1 : CPHA = 1



### 3 Building

The AXIS SPI is written in Verilog 2001. It 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. Linting is performed by verible using the lint target.

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

## 3.3 Targets

### 3.3.1 fusesoc\_info Targets

- default  
Info: Default for IP intergration.
- lint  
Info: Lint with Verible
- sim  
Info: Base simulation using icarus as default.
- sim\_rand\_data  
Info: Use random data as sim input.
- sim\_8bit\_count\_data  
Info: Use counter data as sim input.
- 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
  - **cocotb** testbench files

## 4 Simulation

There are a few different simulations that can be run for this core.

### 4.1 iverilog

iverilog is used for simple test benches for quick verification, visually, of the core.

- **sim** Standard simulation of SPI looped, input/output verification.

This uses a axis stimulator cores for master/slave. This will run all the data in the slave axis SPI interface, which will output the data over the SPI interface. This is then looped into the SPI input that then puts the valid data out on the master axis SPI interface.

### 4.2 cocotb

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

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

Each module has a cocotb based simulation. These use the cocotb extensions made by Alex. The two extensions used are cocotbext-axi and cocotbext-spi. These provide outside verification of the implimentation. These tests consist of the following fusesoc targets.

- **sim\_cocotb** Standard simulation of SPI data to and from cocotbexts this tests all CPOL/CPHA options.

Then you must use the cocotb sim target. The targets above can be run with the following:

```
$ fusesoc run --target sim_cocotb AFRL:device_converter:axis_spi:1.0.0
```



## 5 Module Documentation

- **axis\_spi** Interfaces AXIS to SPI.
- **tb\_spi** Verilog test bench.
- **tb\_cocotb verilog** Verilog test bench base for cocotb.
- **tb\_cocotb python** cocotb unit test functions.

# axis\_spi\_master.v

---

## AUTHORS

---

JAY CONVERTINO

---

## DATES

---

2025/04/22

---

## INFORMATION

---

### Brief

---

Stream SPI input/output data over AXIS bus in master mode.

### 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.

## axis\_spi\_master

---

```
module axis_spi_master #(
    parameter
    CLOCK_SPEED
    =
    20000000,
    parameter
    BUS_WIDTH
    =
    4,
    parameter
    SELECT_WIDTH
    =
    8
) ( input wire aclk, input wire arstn, input wire [BUS_WIDTH*8-1:0] s_axis_t
```

SPI core with axis input/output data. Read/Write is size of BUS\_WIDTH bytes. Write activates core for read.

### Parameters

<b>CLOCK_SPEED</b> parameter	This is the aclk frequency in Hz, this is the the frequency used for the bus and is divided by the rate.
<b>BUS_WIDTH</b> parameter	AXIS data width in bytes.
<b>SELECT_WIDTH</b> parameter	Bit width of the slave select.

### Ports

<b>aclk</b>	Clock for AXIS
<b>arstn</b>	Negative reset for AXIS
<b>s_axis_tdata</b>	Input data for SPI MOSI.
<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 from SPI MISO
<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>sclk</b>	spi clock, should only drive output pins to devices.
<b>mosi</b>	transmit for master output
<b>miso</b>	receive for master input
<b>ssn_i</b>	slave select input
<b>ssn_o</b>	slave select output
<b>rate</b>	output rate of spi core.
<b>cpol</b>	clock polarity of sclk
<b>cpha</b>	clock phase of sclk
<b>miso_dcount</b>	Current number of input bits available from parallel register.
<b>mosi_dcount</b>	current number of output bits available to serial shift output.

## STATE MACHINE

Constants that makeup the data\_state machine.

### ready

```
localparam ready = 3'd1
```

ready and waiting for data

### processing

```
localparam processing = 3'd3
```

data is being processed

## error

---

```
localparam error = 3'd0
```

someone made a whoops

## INSTANTIATED MODULES

---

### inst\_spi\_output\_clk

---

```
mod_clock_ena_gen #(
    .CLOCK_SPEED(CLOCK_SPEED)
) inst_spi_output_clk ( .clk(aclk), .rstn(arstn), .start0(1'b0), .clr(spi_en
```

Generates enable at rate for spi output data.

### inst\_spi\_input\_clk

---

```
mod_clock_ena_gen #(
    .CLOCK_SPEED(CLOCK_SPEED)
) inst_spi_input_clk ( .clk(aclk), .rstn(arstn), .start0(1'b1), .clr(spi_en
```

Generates enable at rate for spi input data.

### inst\_piso

---

```
piso #(
    .BUS_WIDTH(BUS_WIDTH)
) inst_piso ( .clk(aclk), .rstn(arstn), .ena(spi_ena_mosi), .rev(1'b0), .ld
```

take axis input parallel data at bus size, and output the word to the spi bus.

### inst\_sipo

---

```
sipo #(
    .BUS_WIDTH(BUS_WIDTH)
) inst_sipo ( .clk(aclk), .rstn(arstn), .ena(spi_ena_miso), .rev(1'b0), .ld
```

take serial input data, and output the world to the parallel data bus.

# 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

---

### 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.

## single\_word\_00

---

```
@cocotb.test()  
async def single_word_00(  
    dut  
)
```

Coroutine that is identified as a test routine. This routine tests for writing a single word, and then reading a single word for cpol == 0 and cpha == 0.

### Parameters

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

## single\_word\_10

---

```
@cocotb.test()  
async def single_word_10(  
    dut  
)
```

Coroutine that is identified as a test routine. This routine tests for writing a single word, and then reading a single word for cpol == 1 and cpha == 0.

### Parameters

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

## single\_word\_01

---

```
@cocotb.test()  
async def single_word_01(  
    dut  
)
```

Coroutine that is identified as a test routine. This routine tests for writing a single word, and then reading a

single word for cpol == 0 and cpha == 1.

#### Parameters

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

### single\_word\_11

---

```
@cocotb.test()
async def single_word_11(
    dut
)
```

Coroutine that is identified as a test routine. This routine tests for writing a single word, and then reading a single word for cpol == 1 and cpha == 1.

#### 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.

## tb\_cocotb.v

---

### AUTHORS

---

JAY CONVERTINO

---

### DATES

---

2025/04/24

---

### INFORMATION

---

#### Brief

---

Test bench wrapper for 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.

## tb\_cocotb

---

```
module tb_cocotb #(
  parameter
    CLOCK_SPEED
    =
    20000000,
  parameter
    BUS_WIDTH
    =
    4,
  parameter
    SELECT_WIDTH
    =
    1,
  parameter
```



```

    RATE
    =
    115200
) ( input aclk, input arstn, input [BUS_WIDTH*8-1:0] s_axis_tdata, input s_

```

SPI core with axis input/output data. Read/Write is size of BUS\_WIDTH bytes. Write activates core for read.

## Parameters

<b>CLOCK_SPEED</b> parameter	This is the aclk frequency in Hz, this is the the frequency used for the bus and is divided by the rate.
<b>BUS_WIDTH</b> parameter	AXIS data width in bytes.
<b>SELECT_WIDTH</b> parameter	Bit width of the slave select.
<b>RATE</b> parameter	Select the data rate of the spi core.

## Ports

<b>aclk</b>	Clock for AXIS
<b>arstn</b>	Negative reset for AXIS
<b>s_axis_tdata</b>	Input data for UART TX.
<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 from UART RX
<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>sclk</b>	spi clock, should only drive output pins to devices.
<b>mosi</b>	transmit for master output
<b>miso</b>	receive for master input
<b>ssn_i</b>	slave select input
<b>ssn_o</b>	slave select output
<b>rate</b> parameter	output rate of spi core.
<b>cpol</b>	clock polarity of spi_clk
<b>cpha</b>	clock phase of spi_clk
<b>miso_dcount</b>	Current number of input bits available from parallel register.
<b>mosi_dcount</b>	current number of output bits available to serial shift output.

## INSTANTIATED MODULES

---

### dut

```

axis_spi_master #(
    CLOCK_SPEED(CLOCK_SPEED),
    BUS_WIDTH(BUS_WIDTH),
    SELECT_WIDTH(SELECT_WIDTH)

```

```
| ) dut ( .aclk(aclk), .arstn(arstn), .s_axis_tdata(s_axis_tdata), .s_axis_t
```

---

Device under test, axis\_spi\_master

**tb\_spi.v**

---

**AUTHORS**

---

**JAY CONVERTINO**

---

**DATES**

---

**2025/04/22**

---

**INFORMATION**

---

**Brief**

---

Test bench for AXIS SPI

**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.