

# BUS SPI MASTER



May 1, 2025

Jay Convertino

# Contents

<b>1 Usage</b>	<b>2</b>
1.1 Introduction . . . . .	2
1.2 Dependencies . . . . .	2
1.2.1 axi_lite_spi_master Depenecies . . . . .	2
1.2.2 wishbone_standard_spi_master Depenecies . . . . .	2
1.2.3 up_spi_master Depenecies . . . . .	3
1.3 In a Project . . . . .	3
<b>2 Architecture</b>	<b>3</b>
2.1 Waveforms . . . . .	4
<b>3 Building</b>	<b>6</b>
3.1 fusesoc . . . . .	6
3.2 Source Files . . . . .	6
3.2.1 axi_lite_spi_master File List . . . . .	6
3.2.2 wishbone_standard_spi_master File List . . . . .	7
3.2.3 up_spi_master File List . . . . .	7
3.3 Targets . . . . .	7
3.3.1 axi_lite_spi_master Targets . . . . .	7
3.3.2 wishbone_standard_spi_master Targets . . . . .	7
3.3.3 up_spi_master Targets . . . . .	8
3.4 Directory Guide . . . . .	8
<b>4 Simulation</b>	<b>9</b>
4.1 cocotb . . . . .	9
<b>5 Module Documentation</b>	<b>10</b>
5.1 axi_lite_spi_master . . . . .	11
5.2 wishbone_standard_spi_master . . . . .	16
5.3 up_spi_master . . . . .	20
5.3.1 Registers . . . . .	21
5.4 tb_cocotb_wishbone_standard python . . . . .	26
5.5 tb_cocotb_wishbone_standard verilog . . . . .	29
5.6 tb_cocotb_axi_lite python . . . . .	32
5.7 tb_cocotb_axi_lite verilog . . . . .	35
5.8 tb_cocotb_up python . . . . .	38
5.9 tb_cocotb_up verilog . . . . .	42

# 1 Usage

## 1.1 Introduction

BUS SPI Master is a core that emulates and extends the Altera SPI IP core. This is a SPI master device only. It is currently available as a AXI Lite, Wishbone Standard, and uP bus IP core. The Altera core this is based on has Linux and uboot drivers, so by mimicking it this core has instant access to drivers. There is a extension register for control that allows for changes to the SPI Master the Altera IP does not have. The following is information on how to use the device in an FPGA, software, and in simulation.

## 1.2 Dependencies

The following are the dependencies of the cores.

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

### 1.2.1 axi\_lite\_spi\_master Depenecies

- dep
  - AFRL:utility:helper:1.0.0
  - AFRL:device:up\_spi\_master:1.0.0
  - AD:common:up\_axi:1.0.0
- dep\_tb
  - AFRL:simulation:axis\_stimulator
  - AFRL:utility:sim\_helper

### 1.2.2 wishbone\_standard\_spi\_master Depenecies

- dep
  - AFRL:utility:helper:1.0.0
  - AFRL:device:up\_spi\_master:1.0.0
  - AFRL:bus:up\_wishbone\_standard:1.0.0

### 1.2.3 up\_spi\_master Dependencies

- dep
  - AFRL:utility:helper:1.0.0
  - AFRL:device\_converter:axis\_spi\_master:1.0.0

## 1.3 In a Project

First, pick a core that matches the target bus in question. Then connect the BUS SPI MASTER core to that bus. Once this is complete the SPI pins will need to be routed to the slave SPI devices

## 2 Architecture

This core is made up of other cores that are documented in detail in their source. The cores this is made up of are the,

- **axis\_spi\_master** Interface with SPI master and present the data over AXIS interface (see core for documentation).
- **up\_axi** An AXI Lite to uP converter core (see core for documentation).
- **up\_wishbone\_standard** A wishbone standard to uP converter core (see core for documentation).
- **up\_spi\_master** Takes uP bus and converts it to interface to the AXIS SPI (see module documentation for information 5).

For register documentation please see up\_spi\_master in 5

Interrupts for this core are enabled in the control register. First the general IE bit, interrupt enable, is set to 1. Then bits are chosen that set what conditions the interrupt should be activated for. This interrupt goes active high (1) when a condition becomes true. Starting with the interrupt end of packet bit (IEOP), setting this active will enable the interrupt to go off when the status EOP bit is true. This will stay that way till the tx or rx register is cleared of the EOP word. Interrupt read ready (IRRDY) when set active will trigger an interrupt when the status bit RRDY is active. This will stay tripped till a word is read. Interrupt transmit ready (ITRDY) when set active will trigger an interrupt when the status bit TRDY is active. This will stay tripped till a word is written. Interrupt transmit overrun (ITOE) when set active will trigger an interrupt when the status bit TOE is active. This will stay tripped till the status register is written. Interrupt receive underrun when set active will trigger an interrupt when the status bit ROE is

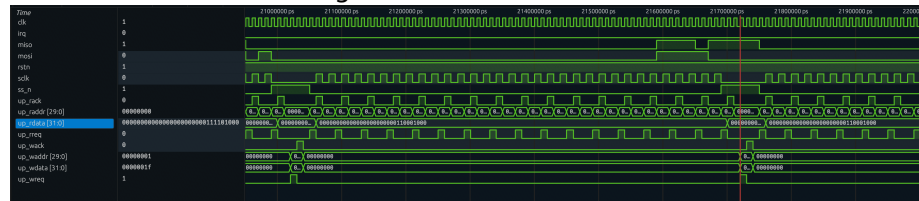
active. ITOE, TOE, IROE, ROE, and E are only cleared by writing to the status register. The status register does NOT write any actual data to the status register, so the status bits are not directly affected. It simply resets ROE/TOE to 0. See 5 up\_spi\_master for more detail the location and function of register bits.

## 2.1 Waveforms

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

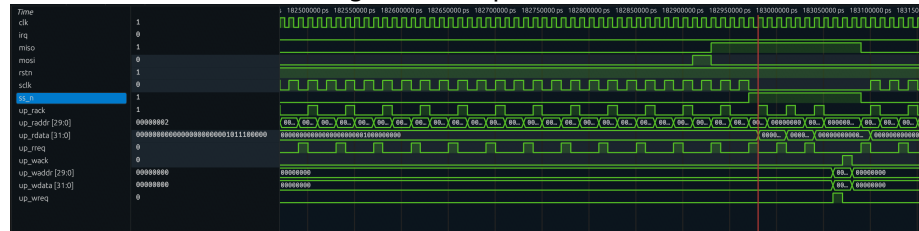
Write slave is the first uP test that looks at writing to a slave device using the SPI master without reading the received data.

Figure 1: write slave uP



Loop test checks for reads and writes on the uP bus.

Figure 2: loop test uP



Next are various interrupt enable tests to see if they respond as they should.

Figure 3: read ready interrupt enabled

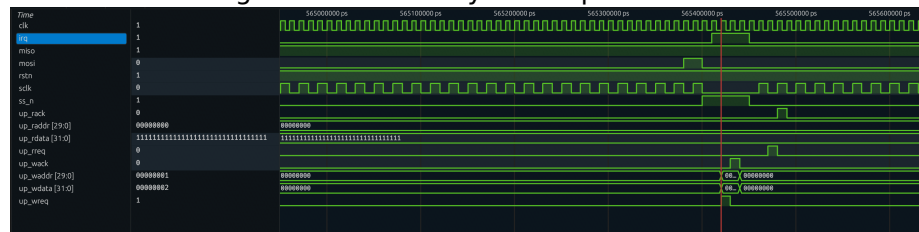
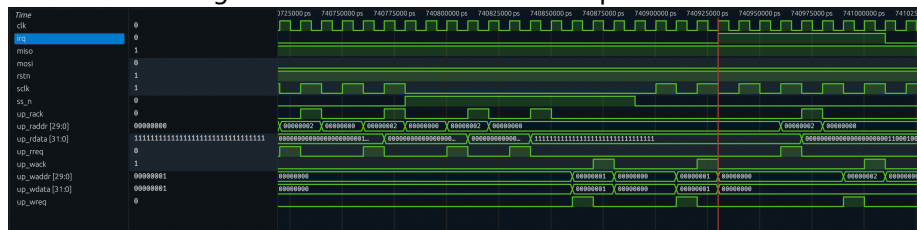
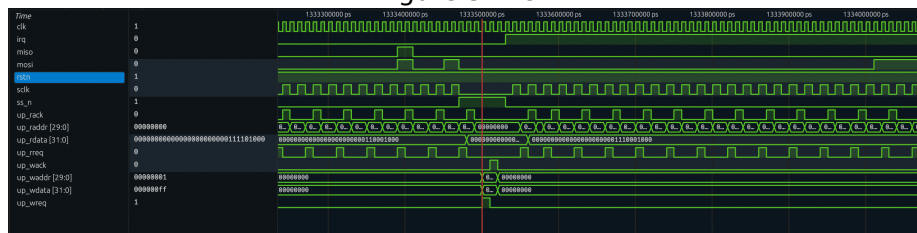


Figure 4: transmit error interrupt enabled



End of packet is a feature added to the Altera IP core in 2019. It is not present in the linux drivers. Its functionality is questionable when it comes to being useful. Essentially it allows the device to signal bits or a irq if a EOP word matches the transmit or receive register contents. It is cleared when those contents change and are not the EOP.

Figure 5: EOP



The last two waveforms show the uP to bus adapters in a loopback test. Since it takes two writes to get the SPI slave to echo a value written two write previous, the read value will be two writes behind.

Figure 6: loop test AXI Lite

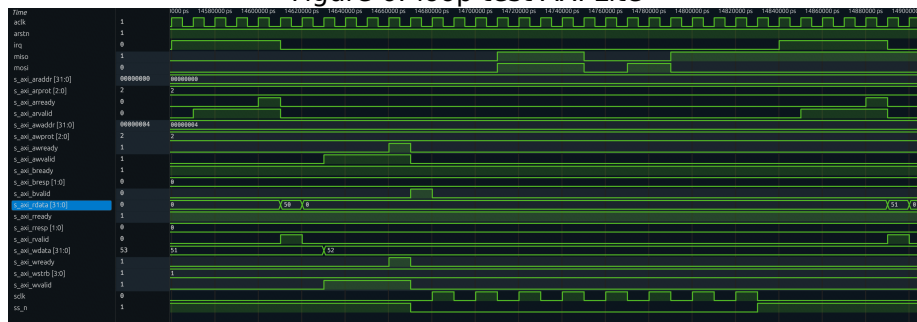
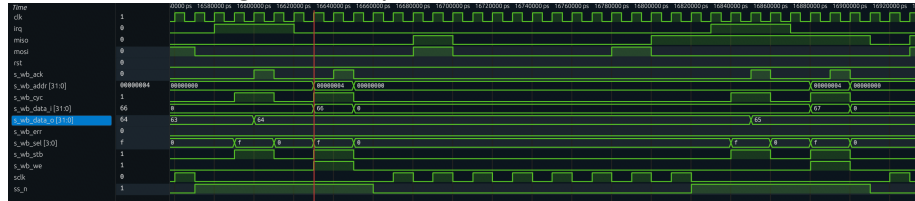


Figure 7: loop test wishbone standard



### 3 Building

The BUS SPI MASTER 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.

### 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 axi\_lite\_spi\_master File List

- src
  - src/axi\_lite\_spi\_master.v
- tb\_cocotb
  - 'tb/tb\_cocotb\_axi\_lite.py': 'file\_type': 'user', 'copyto': '.'
  - 'tb/tb\_cocotb\_axi\_lite.v': 'file\_type': 'verilogSource'
- tb
  - tb/tb\_uart.v

### **3.2.2 wishbone\_standard\_spi\_master File List**

- src
  - src/wishbone\_standard\_spi\_master.v
- tb\_cocotb
  - 'tb/tb\_cocotb\_wishbone\_standard.py': 'file\_type': 'user', 'copyto':  
'.'
  - 'tb/tb\_cocotb\_wishbone\_standard.v': 'file\_type': 'verilogSource'

### **3.2.3 up\_spi\_master File List**

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

## **3.3 Targets**

### **3.3.1 axi\_lite\_spi\_master Targets**

- default
  - Info: Default for IP intergration.
- sim\_cocotb
  - Info: Cocotb unit tests

### **3.3.2 wishbone\_standard\_spi\_master Targets**

- default
  - Info: Default for IP intergration.
- sim\_cocotb
  - Info: Cocotb unit tests



### 3.3.3 up\_spi\_master Targets

- default  
Info: Default for IP intergration.
- 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 cocotb

Cocotb is the only method for simulating the various iterations of the bus\_spi\_master core. At the moment there is a axi\_lite, wishbone\_standard, and uP based versions. This is currently set to use icarus as the sim tool for cocotb. The uP testbench is the one that will test all the various register bits and interrupt options. Others will only do loop back tests.

To run the wishbone sim use the command below.

```
fusesoc run --target sim_cocotb AFRL:device:wishbone_standard_spi_master:1.0.0
```

To run the axi\_lite sim use the command below.

```
fusesoc run --target sim_cocotb AFRL:device:axi_lite_spi_master:1.0.0
```

To run the uP sim use the command below.

```
fusesoc run --target sim_cocotb AFRL:device:up_spi_master:1.0.0
```

## 5 Module Documentation

`up_spi_master` is the module that integrates the AXIS SPI MASTER core. This uses inputs/outputs for data tied directly to registers mapped in the uP bus. The uP bus is the microprocessor bus based on Analog Devices design. It resembles a APB bus in design, and is the bridge to other buses BUS SPI MASTER can use. This makes changing for AXI Lite, to Wishbone to whatever quick and painless.

`axi_lite_spi` module adds a AXI Lite to uP (microprocessor) bus converter. The converter is from Analog Devices.

`wishbone_standard_spi` module adds a Wishbone Standard to uP (microprocessor) bus converter. This converter was designed for Wishbone Standard only, NOT pipelined or Registered (cti).

The next sections document these modules. `up_spi_master` contains the register map explained, and what the various bits do.

# axi\_lite\_spi\_master.v

---

## AUTHORS

---

JAY CONVERTINO

---

## DATES

---

2025/04/30

---

## INFORMATION

---

### Brief

---

AXI Lite SPI Master is a core for interfacing with SPI Slave devices.

### 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.

## axi\_lite\_spi\_master

---

```
module axi_lite_spi_master #(
  parameter
  ADDRESS_WIDTH
  =
  32,
  parameter
  BUS_WIDTH
  =
  4,
  parameter
  WORD_WIDTH
  =
  4,
  parameter
```

```

CLOCK_SPEED
=
100000000,
parameter
SELECT_WIDTH
=
16,
parameter
DEFAULT_RATE_DIV
=
0,
parameter
DEFAULT_CPOL
=
0,
parameter
DEFAULT_CPHA
=
0
) ( input aclk, input arstn, input s_axi_awvalid, input [ADDRESS_WIDTH-1:0]

```

AXI Lite based SPI Master device. BUS\_WIDTH is 4 bytes.

## Parameters

<b>ADDRESS_WIDTH</b> parameter	Width of the uP address port, max 32 bit.
<b>BUS_WIDTH</b> parameter	Width of the uP bus data port, only valid values are 2 or 4.
<b>WORD_WIDTH</b> parameter	Width of each SPI Master word. This will also set the bits used in the TX/RX data registers. Must be less than or equal to BUS_WIDTH. VALID: 1 to 4.
<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>SELECT_WIDTH</b> parameter	Bit width of the slave select, defaults to 16 to match altera spi ip.
<b>DEFAULT_RATE_DIV</b> parameter	Default divider value of the main clock to use for the spi data output clock rate. 0 is 2 (2^(X+1) X is the DEFAULT_RATE_DIV)
<b>DEFAULT_CPOL</b> parameter	Default clock polarity for the core (0 or 1).
<b>DEFAULT_CPHA</b> parameter	Default clock phase for the core (0 or 1).

## Ports

<b>aclk</b>	Clock for all devices in the core
<b>arstn</b>	Negative reset
<b>s_axi_awvalid</b>	Axi Lite aw valid
<b>s_axi_awaddr</b>	Axi Lite aw addr
<b>s_axi_awprot</b>	Axi Lite aw prot
<b>s_axi_awready</b>	Axi Lite aw ready
<b>s_axi_wvalid</b>	Axi Lite w valid
<b>s_axi_wdata</b>	Axi Lite w data
<b>s_axi_wstrb</b>	Axi Lite w strb
<b>s_axi_wready</b>	Axi Lite w ready
<b>s_axi_bvalid</b>	Axi Lite b valid
<b>s_axi_bresp</b>	Axi Lite b resp

<b>s_axi_bready</b>	Axi Lite b ready
<b>s_axi_arvalid</b>	Axi Lite ar valid
<b>s_axi_araddr</b>	Axi Lite ar addr
<b>s_axi_arprot</b>	Axi Lite ar prot
<b>s_axi_arready</b>	Axi Lite ar ready
<b>s_axi_rvalid</b>	Axi Lite r valid
<b>s_axi_rdata</b>	Axi Lite r data
<b>s_axi_rresp</b>	Axi Lite r resp
<b>s_axi_rready</b>	Axi Lite r ready
<b>irq</b>	Interrupt when data is received
<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>ss_n</b>	slave select output

## up\_rreq

---

```
wire up_rreq
```

uP read bus request

## up\_rack

---

```
wire up_rack
```

uP read bus acknowledge

## up\_raddr

---

```
wire [ADDRESS_WIDTH-(
  BUS_WIDTH
  2
)-1:0] up_raddr
```

uP read bus address

## up\_rdata

---

```
wire [31:0] up_rdata
```

uP read bus request

## up\_wreq

---

```
wire up_wreq
```

uP write bus request

## up\_wack

---

```
wire up_wack
```

uP write bus acknowledge

## up\_waddr

---

```
wire [ADDRESS_WIDTH-(  
BUS_WIDTH  
2  
)-1:0] up_waddr
```

uP write bus address

## up\_wdata

---

```
wire [31:0] up_wdata
```

uP write bus data

## INSTANTIATED MODULES

---

### inst\_up\_axi

---

```
up_axi #(  
AXI_ADDRESS_WIDTH(ADDRESS_WIDTH)  
) inst_up_axi ( .up_rstn (arstn), .up_clk (aclk), .up_axi_awvalid(s_axi_awv
```

Module instance of up\_axi for the AXI Lite bus to the uP bus.

### inst\_up\_spi\_master

---

```
up_spi_master #(  
ADDRESS_WIDTH(ADDRESS_WIDTH),  
BUS_WIDTH(BUS_WIDTH),  
WORD_WIDTH(WORD_WIDTH),  
CLOCK_SPEED(CLOCK_SPEED),  
SELECT_WIDTH(SELECT_WIDTH),  
DEFAULT_RATE_DIV(DEFAULT_RATE_DIV),
```

```

        DEFAULT_CPOL(DEFAULT_CPOL),
        DEFAULT_CPHA(DEFAULT_CPHA)
    ) inst_up_spi_master ( .clk(aclk), .rstn(arstn), .up_rreq(up_rreq), .up_rack

```

Module instance of up\_spi\_master creating a Logic wrapper for spi master axis bus cores to interface with uP bus.



# wishbone\_standard\_spi\_master.v

---

## AUTHORS

---

JAY CONVERTINO

---

## DATES

---

2025/04/30

---

## INFORMATION

---

### Brief

---

Wishbone Standard SPI Master core.

### 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.

## wishbone\_standard\_spi\_master

---

```
module wishbone_standard_spi_master #(
  parameter
    ADDRESS_WIDTH
    =
    32,
  parameter
    BUS_WIDTH
    =
    4,
  parameter
    WORD_WIDTH
    =
    4,
  parameter
```

```

CLOCK_SPEED
=
100000000,
parameter
SELECT_WIDTH
=
16,
parameter
DEFAULT_RATE_DIV
=
0,
parameter
DEFAULT_CPOL
=
0,
parameter
DEFAULT_CPHA
=
0
) ( input clk, input rst, input s_wb_cyc, input s_wb_stb, input s_wb_we, input s_wb_addr, input s_wb_data_i, input s_wb_sel, input s_wb_ack, input s_wb_data_o, input s_wb_err, input irq )

```

Wishbone Standard based SPI Master device.

## Parameters

<b>ADDRESS_WIDTH</b> parameter	Width of the uP address port, max 32 bit.
<b>BUS_WIDTH</b> parameter	Width of the uP bus data port, only valid values are 2 or 4.
<b>WORD_WIDTH</b> parameter	Width of each SPI Master word. This will also set the bits used in the TX/RX data registers. Must be less than or equal to BUS_WIDTH. VALID: 1 to 4.
<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>SELECT_WIDTH</b> parameter	Bit width of the slave select, defaults to 16 to match altera spi ip.
<b>DEFAULT_RATE_DIV</b> parameter	Default divider value of the main clock to use for the spi data output clock rate. 0 is 2 (2^(X+1) X is the DEFAULT_RATE_DIV)
<b>DEFAULT_CPOL</b> parameter	Default clock polarity for the core (0 or 1).
<b>DEFAULT_CPHA</b> parameter	Default clock phase for the core (0 or 1).

## Ports

<b>clk</b>	Clock for all devices in the core
<b>rst</b>	Positive reset
<b>s_wb_cyc</b>	Bus Cycle in process
<b>s_wb_stb</b>	Valid data transfer cycle
<b>s_wb_we</b>	Active High write, low read
<b>s_wb_addr</b>	Bus address
<b>s_wb_data_i</b>	Input data
<b>s_wb_sel</b>	Device Select
<b>s_wb_ack</b>	Bus transaction terminated
<b>s_wb_data_o</b>	Output data
<b>s_wb_err</b>	Active high when a bus error is present
<b>irq</b>	Interrupt when data is received

<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>ss_n</b>	slave select output

## up\_rreq

---

```
wire up_rreq
```

uP read bus request

## up\_rack

---

```
wire up_rack
```

uP read bus acknowledge

## up\_raddr

---

```
wire [ADDRESS_WIDTH-(  
BUS_WIDTH  
2  
)-1:0] up_raddr
```

uP read bus address

## up\_rdata

---

```
wire [31:0] up_rdata
```

uP read bus request

## up\_wreq

---

```
wire up_wreq
```

uP write bus request

## up\_wack

---

```
wire up_wack
```

uP write bus acknowledge

## up\_waddr

---

```
wire [ADDRESS_WIDTH-(
BUS_WIDTH
2
)-1:0] up_waddr
```

uP write bus address

## up\_wdata

```
wire [31:0] up_wdata
```

uP write bus data

## INSTANTIATED MODULES

### inst\_up\_wishbone\_standard

```
up_wishbone_standard #(
ADDRESS_WIDTH(ADDRESS_WIDTH),
BUS_WIDTH(BUS_WIDTH)
) inst_up_wishbone_standard ( .clk(clk), .rst(rst), .s_wb_cyc(s_wb_cyc), .s_
```

Module instance of up\_wishbone\_standard for the Wishbone Classic Standard bus to the uP bus.

### inst\_up\_spi\_master

```
up_spi_master #(
ADDRESS_WIDTH(ADDRESS_WIDTH),
BUS_WIDTH(BUS_WIDTH),
WORD_WIDTH(WORD_WIDTH),
CLOCK_SPEED(CLOCK_SPEED),
SELECT_WIDTH(SELECT_WIDTH),
DEFAULT_RATE_DIV(DEFAULT_RATE_DIV),
DEFAULT_CPOL(DEFAULT_CPOL),
DEFAULT_CPHA(DEFAULT_CPHA)
) inst_up_spi_master ( .clk(clk), .rstn(~rst), .up_rreq(up_rreq), .up_rack(u
```

Module instance of up\_spi\_master creating a Logic wrapper for spi master axis bus cores to interface with uP bus.

# up\_spi\_master.v

---

## AUTHORS

---

JAY CONVERTINO

---

## DATES

---

2024/04/29

---

## INFORMATION

---

### Brief

---

uP Core for interfacing with axis spi that emulates the ALTERA SPI IP.

### 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.

## up\_spi\_master

---

```
module up_spi_master #(
  parameter
  ADDRESS_WIDTH
  =
  32,
  parameter
  BUS_WIDTH
  =
  4,
  parameter
  WORD_WIDTH
  =
  4,
  parameter
```

```

CLOCK_SPEED
=
100000000,
parameter
SELECT_WIDTH
=
16,
parameter
DEFAULT_RATE_DIV
=
0,
parameter
DEFAULT_CPOL
=
0,
parameter
DEFAULT_CPHA
=
0
) ( input clk, input rstn, input up_rreq, output up_rack, input [ADDRESS_WID

```

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

## Parameters

<b>ADDRESS_WIDTH</b> parameter	Width of the uP address port, max 32 bit.
<b>BUS_WIDTH</b> parameter	Width of the uP bus data port, only valid values are 2 or 4.
<b>WORD_WIDTH</b> parameter	Width of each SPI Master word. This will also set the bits used in the TX/RX data registers. Must be less than or equal to BUS_WIDTH, VALID: 1 to 4.
<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>SELECT_WIDTH</b> parameter	Bit width of the slave select, defaults to 16 to match altera spi ip.
<b>DEFAULT_RATE_DIV</b> parameter	Default divider value of the main clock to use for the spi data output clock rate. 0 is 2 (2^(X+1) X is the DEFAULT_RATE_DIV)
<b>DEFAULT_CPOL</b> parameter	Default clock polarity for the core (0 or 1).
<b>DEFAULT_CPHA</b> parameter	Default clock phase for the core (0 or 1).

## Ports

<b>clk</b>	Clock for all devices in the core
<b>rstn</b>	Negative reset
<b>up_rreq</b>	uP bus read request
<b>up_rack</b>	uP bus read ack
<b>up_raddr</b>	uP bus read address
<b>up_rdata</b>	uP bus read data
<b>up_wreq</b>	uP bus write request
<b>up_wack</b>	uP bus write ack
<b>up_waddr</b>	uP bus write address
<b>up_wdata</b>	uP bus write data
<b>irq</b>	Interrupt when data is received
<b>sclk</b>	spi clock, should only drive output pins to devices.

**mosi**            transmit for master output  
**miso**            receive for master input  
**ss\_n**            slave select output

## DIVISOR

```
localparam DIVISOR = BUS_WIDTH/2
```

Divide the address register default location for 1 byte access to multi byte access. (register offsets are byte offsets).

## REG\_SIZE

```
localparam REG_SIZE = 8
```

Number of bits for the register address

## REGISTER INFORMATION

Core has 7 registers at the offsets that follow when at a full 32 bit bus width, Internal address is OFFSET >> BUS\_WIDTH/2 (32bit would be h4 >> 2 = 1 for internal address).

<b>RX_DATA_REG</b>	h00
<b>TX_DATA_REG</b>	h04
<b>STATUS_REG</b>	h08
<b>CONTROL_REG</b>	h0C
<b>RESERVED</b>	h10
<b>SLAVE_SELECT_REG</b>	h14
<b>EOP_VALUE_REG</b>	h18
<b>CONTROL_EXT_REG</b>	h1C

## RX\_DATA\_REG

```
localparam RX_DATA_REG = 8'h0 >> DIVISOR
```

Defines the address offset for RX DATA OUTPUT

RX DATA REGISTER	
31:N	N:0
UNUSED	RECEIVED DATA

Valid bits are from WORD\_WIDTH\*8-1:0, which are data.

## TX\_DATA\_REG

```
localparam TX_DATA_REG = 8'h4 >> DIVISOR
```

Defines the address offset to write the TX DATA INPUT.

TX DATA REGISTER	
31:N	N:0
UNUSED	TRANSMIT DATA

Valid bits are from WORD\_WIDTH\*8-1:0, which are data.

## STATUS\_REG

```
localparam STATUS_REG = 8'h8 >> DIVISOR
```

Defines the address offset to read the status bits.

STATUS REGISTER								
31:10	9	8	7	6	5	4	3	2:0
UNUSED	EOP	E	RRDY	TRDY	TMT	TOE	ROE	UNUSED

### Status Register, 1 is considered active.

<b>EOP</b>	9, This bit is active(1) when the EOP_VALUE_REG is equal to RX_DATA_REG or TX_DATA_REG.
<b>E</b>	8, Logical or of TOE and ROE (Clear by writing status).
<b>RRDY</b>	7, Receive is ready (full) when the bit is 1, empty when the bit is 0.
<b>TRDY</b>	6, Transmit is ready (empty) when the bit is 1, full when the bit is 0.
<b>TMT</b>	5, Transmit shift register empty is set to 1 when all bits have been output.
<b>TOE</b>	4, Transmit overrun is set to 1 when a TX_DATA_REG write happens whne TRDY is 1 (Clear by writing status reg).
<b>ROE</b>	3, Receive overrun is set to 1 when RRDY is 1 and a new received word is going to be written to RX_DATA_REG (Clear by writing status reg)

## CONTROL\_REG

```
localparam CONTROL_REG = 8'hC >> DIVISOR
```

Defines the address offset to set the control bits.

CONTROL REGISTER									
31:11	10	9	8	7	6	5	4	3	2:0
UNUSED	SSO	IEOP	IE	IRRDY	ITRDY	UNUSED	ITOE	IROE	UNUSED



Control Register, 1 is considered active. **All zeros on reset.**

<b>SSO</b>	10, Setting this to 1 will force all ss_n lines to 0 (selected).
<b>IEOP</b>	9, Generate a interrupt on EOP status bit going active if set to 1.
<b>IE</b>	8, Enable (1) or disable(0) all interrupts that are active.
<b>IRRDY</b>	7, Generate a interrupt on RRDY status bit going active if set to 1.
<b>ITRDY</b>	6, Generate a interrupt on TRDY status bit going active if set to 1.
<b>ITOE</b>	4, Generate a interrupt on TOE status bit going active if set to 1.
<b>IROE</b>	3, Generate a interrupt on ROE status bit going active if set to 1.

## RESERVED

```
localparam RESERVED = 8'h10 >> DIVISOR
```

Defines the address offset that is not used.

## SLAVE\_SELECT\_REG

```
localparam SLAVE_SELECT_REG = 8'h14 >> DIVISOR
```

Defines the address offset to set the slave select value

SLAVE SELECT REGISTER	
31:N	N:0
UNUSED	SLAVE SELECT

Valid bits are from SELECT\_WIDTH-1:0, which are the slave select output lines to drive low during data transmission.

## EOP\_VALUE\_REG

```
localparam EOP_VALUE_REG = 8'h18 >> DIVISOR
```

Defines the address offset to set the end of packet match value

EOP REGISTER	
31:N	N:0
UNUSED	EOP

Valid bits are from BUS\_WIDTH\*8:0, which are used to check for a word match between rx and/or tx and

update status.

## CONTROL\_EXT\_REG

```
localparam CONTROL_EXT_REG = 8'h1C >> DIVISOR
```

Defines the address offset for control register extensions

CONTROL REGISTER EXTENDED			
31:6	5	4	3:0
UNUSED	CPHA	CPOL	RATE DIV

## Control Extension to add capabilities to Altera IP core.

<b>CPHA</b>	5, Clock Phase Bit, 0 or 1 per SPI specs (default value set by IP parameter).
<b>CPOL</b>	4, Clock Polarity bit, 0 or 1 per SPI specs (default value set by IP parameter).
<b>RATE_TOP</b>	3, Top bit for rate control. Divider values are 0 to 15 ( $2^X+1$ where X is the divider value).
<b>RATE_BOT</b>	0, Bottom bit for rate control.

## INSTANTIATED MODULES

### inst\_axis\_spi

```
axis_spi_master #(
    CLOCK_SPEED(CLOCK_SPEED),
    BUS_WIDTH(WORD_WIDTH),
    SELECT_WIDTH(SELECT_WIDTH)
) inst_axis_spi_master ( .aclk(clk), .arstn(rstn), .s_axis_tdata(r_tx_wdata
```

SPI Master instance with AXIS interface

# tb\_cocotb\_wishbone\_standard.py

---

## AUTHORS

---

JAY CONVERTINO

---

## DATES

---

2025/04/30

---

## INFORMATION

---

### Brief

---

Cocotb test bench

### 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.

## 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.

## loop\_data

---

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

Coroutine that is identified as a test routine. Use echo slave to loop data, check write wishbone equals spi slave contents, bus writes equal bus reads.

### 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\_wishbone\_standard.v

---

## AUTHORS

---

JAY CONVERTINO

---

## DATES

---

2025/04/30

---

## 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.BUS\_WIDTH

## tb\_cocotb

---

```
module tb_cocotb #(
  parameter
    ADDRESS_WIDTH
    =
    32,
  parameter
    BUS_WIDTH
    =
    4,
  parameter
    WORD_WIDTH
    =
    4,
  parameter
```

```

CLOCK_SPEED
=
100000000,
parameter
SELECT_WIDTH
=
16,
parameter
DEFAULT_RATE_DIV
=
0,
parameter
DEFAULT_CPOL
=
0,
parameter
DEFAULT_CPHA
=
0
) ( input clk, input rst, input s_wb_cyc, input s_wb_stb, input s_wb_we, in

```

Wishbone Standard based SPI Master device.

## Parameters

<b>ADDRESS_WIDTH</b> parameter	Width of the uP address port, max 32 bit.
<b>BUS_WIDTH</b> parameter	Width of the uP bus data port(can not be less than 2 bytes, max tested is 4).
<b>WORD_WIDTH</b> parameter	Width of each SPI Master word. This will also set the bits used in the TX/RX data registers. Must be less than or equal to BUS_WIDTH 1 to 4.
<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>SELECT_WIDTH</b> parameter	Bit width of the slave select, defaults to 16 to match altera spi ip.
<b>DEFAULT_RATE_DIV</b> parameter	Default divider value of the main clock to use for the spi data output clock rate. 0 is 2 (2^(X+1) X is the DEFAULT_RATE_DIV)
<b>DEFAULT_CPOL</b> parameter	Default clock polarity for the core (0 or 1).
<b>DEFAULT_CPHA</b> parameter	Default clock phase for the core (0 or 1).

## Ports

<b>clk</b>	Clock for all devices in the core
<b>rst</b>	Positive reset
<b>s_wb_cyc</b>	Bus Cycle in process
<b>s_wb_stb</b>	Valid data transfer cycle
<b>s_wb_we</b>	Active High write, low read
<b>s_wb_addr</b>	Bus address
<b>s_wb_data_i</b>	Input data
<b>s_wb_sel</b>	Device Select
<b>s_wb_ack</b>	Bus transaction terminated
<b>s_wb_data_o</b>	Output data
<b>s_wb_err</b>	Active high when a bus error is present
<b>irq</b>	Interrupt when data is received





# tb\_cocotb\_axi\_lite.py

---

## AUTHORS

---

JAY CONVERTINO

---

## DATES

---

2025/03/04

---

## INFORMATION

---

### Brief

---

Cocotb test bench

### 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.

## 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.

## loop\_data

---

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

Coroutine that is identified as a test routine. Use echo slave to loop data, check write axi equals spi slave contents, axi writes equal axi reads.

### 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\_axi\_lite.v

---

## AUTHORS

---

JAY CONVERTINO

---

## DATES

---

2025/04/30

---

## 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. BUS\_WIDTH

## tb\_cocotb

---

```
module tb_cocotb #(
  parameter
  ADDRESS_WIDTH
  =
  32,
  parameter
  BUS_WIDTH
  =
  4,
  parameter
  WORD_WIDTH
  =
  4,
  parameter
```

```

CLOCK_SPEED
=
100000000,
parameter
SELECT_WIDTH
=
16,
parameter
DEFAULT_RATE_DIV
=
0,
parameter
DEFAULT_CPOL
=
0,
parameter
DEFAULT_CPHA
=
0
) ( input aclk, input arstn, input s_axi_awvalid, input [ADDRESS_WIDTH-1:0]

```

AXI Lite based SPI Master device.

## Parameters

<b>ADDRESS_WIDTH</b> parameter	Width of the uP address port, max 32 bit.
<b>BUS_WIDTH</b> parameter	Width of the uP bus data port, only valid values are 2 or 4.
<b>WORD_WIDTH</b> parameter	Width of each SPI Master word. This will also set the bits used in the TX/RX data registers. Must be less than or equal to BUS_WIDTH 1 to 4.
<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>SELECT_WIDTH</b> parameter	Bit width of the slave select, defaults to 16 to match altera spi ip.
<b>DEFAULT_RATE_DIV</b> parameter	Default divider value of the main clock to use for the spi data output clock rate. 0 is 2 (2^(X+1) X is the DEFAULT_RATE_DIV)
<b>DEFAULT_CPOL</b> parameter	Default clock polarity for the core (0 or 1).
<b>DEFAULT_CPHA</b> parameter	Default clock phase for the core (0 or 1).

## Ports

<b>aclk</b>	Clock for all devices in the core
<b>arstn</b>	Negative reset
<b>s_axi_awvalid</b>	Axi Lite aw valid
<b>s_axi_awaddr</b>	Axi Lite aw addr
<b>s_axi_awprot</b>	Axi Lite aw prot
<b>s_axi_awready</b>	Axi Lite aw ready
<b>s_axi_wvalid</b>	Axi Lite w valid
<b>s_axi_wdata</b>	Axi Lite w data
<b>s_axi_wstrb</b>	Axi Lite w strb
<b>s_axi_wready</b>	Axi Lite w ready
<b>s_axi_bvalid</b>	Axi Lite b valid
<b>s_axi_bresp</b>	Axi Lite b resp

<b>s_axi_bready</b>	Axi Lite b ready
<b>s_axi_arvalid</b>	Axi Lite ar valid
<b>s_axi_araddr</b>	Axi Lite ar addr
<b>s_axi_arprot</b>	Axi Lite ar prot
<b>s_axi_arready</b>	Axi Lite ar ready
<b>s_axi_rvalid</b>	Axi Lite r valid
<b>s_axi_rdata</b>	Axi Lite r data
<b>s_axi_rresp</b>	Axi Lite r resp
<b>s_axi_rready</b>	Axi Lite r ready
<b>irq</b>	Interrupt when data is received
<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>ss_n</b>	slave select output

## INSTANTIATED MODULES

---

### dut

---

```

axi_lite_spi_master #(
    ADDRESS_WIDTH(ADDRESS_WIDTH),
    BUS_WIDTH(BUS_WIDTH),
    WORD_WIDTH(WORD_WIDTH),
    CLOCK_SPEED(CLOCK_SPEED),
    SELECT_WIDTH(SELECT_WIDTH),
    DEFAULT_RATE_DIV(DEFAULT_RATE_DIV),
    DEFAULT_CPOL(DEFAULT_CPOL),
    DEFAULT_CPHA(DEFAULT_CPHA)
) dut ( .aclk(aclk), .arstn(arstn), .s_axi_awvalid(s_axi_awvalid), .s_axi_av

```

Device under test, axi\_lite\_spi\_master

# tb\_cocotb\_up.py

---

## AUTHORS

---

JAY CONVERTINO

---

## DATES

---

2025/04/29

---

## INFORMATION

---

### Brief

---

Cocotb test bench

### 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.

## 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.

## write\_slave\_test

---

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

Coroutine that is identified as a test routine. Simply write data over uP bus to SPI mosi

### Parameters

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

## loop\_test

---

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

Coroutine that is identified as a test routine. Loop test SPI

### Parameters

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

## IRRDY\_test

---

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

Coroutine that is identified as a test routine. Receive Ready interrupt test



### Parameters

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

## ITRDY\_test

---

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

Coroutine that is identified as a test routine. Transmit Ready interrupt test

### Parameters

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

## ITOE\_test

---

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

Coroutine that is identified as a test routine. Transmit Written when not ready interrupt test

### Parameters

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

## IROE\_test

---

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

Coroutine that is identified as a test routine. Receive was never read, we missed data.

### Parameters

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

## SSO\_assert\_test

---

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

Coroutine that is identified as a test routine. Write control SS bit to assert all enable lines.

### Parameters

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

## end\_of\_packet\_test

---

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

Coroutine that is identified as a test routine. check if the packet 0xAA has been added every 10th word. No check on EOP receive at the moment.

### 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\_up.v

---

## AUTHORS

---

JAY CONVERTINO

---

## DATES

---

2025/04/29

---

## 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. BUS\_WIDTH

## tb\_cocotb

---

```
module tb_cocotb #(
  parameter
  ADDRESS_WIDTH
  =
  32,
  parameter
  BUS_WIDTH
  =
  4,
  parameter
  WORD_WIDTH
  =
  4,
  parameter
```

```

CLOCK_SPEED
=
100000000,
parameter
SELECT_WIDTH
=
16,
parameter
DEFAULT_RATE_DIV
=
0,
parameter
DEFAULT_CPOL
=
0,
parameter
DEFAULT_CPHA
=
0
) ( input clk, input rstn, input up_rreq, output up_rack, input [ADDRESS_WID

```

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

## Parameters

<b>ADDRESS_WIDTH</b> parameter	Width of the uP address port, max 32 bit.
<b>BUS_WIDTH</b> parameter	Width of the uP bus data port(can not be less than 2 bytes, max tested is 4).
<b>WORD_WIDTH</b> parameter	Width of each SPI Master word. This will also set the bits used in the TX/RX data registers. Must be less than or equal to BUS_WIDTH 1 to 4.
<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>SELECT_WIDTH</b> parameter	Bit width of the slave select, defaults to 16 to match altera spi ip.
<b>DEFAULT_RATE_DIV</b> parameter	Default divider value of the main clock to use for the spi data output clock rate. 0 is 2 (2^(X+1) X is the DEFAULT_RATE_DIV)
<b>DEFAULT_CPOL</b> parameter	Default clock polarity for the core (0 or 1).
<b>DEFAULT_CPHA</b> parameter	Default clock phase for the core (0 or 1).

## Ports

<b>clk</b>	Clock for all devices in the core
<b>rstn</b>	Negative reset
<b>up_rreq</b>	uP bus read request
<b>up_rack</b>	uP bus read ack
<b>up_raddr</b>	uP bus read address
<b>up_rdata</b>	uP bus read data
<b>up_wreq</b>	uP bus write request
<b>up_wack</b>	uP bus write ack
<b>up_waddr</b>	uP bus write address
<b>up_wdata</b>	uP bus write data
<b>irq</b>	Interrupt when data is received
<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>ss_n</b>	slave select output

## INSTANTIATED MODULES

---

### dut

---

```

up_spi_master #(
    ADDRESS_WIDTH(ADDRESS_WIDTH),
    BUS_WIDTH(BUS_WIDTH),
    WORD_WIDTH(WORD_WIDTH),
    CLOCK_SPEED(CLOCK_SPEED),
    SELECT_WIDTH(SELECT_WIDTH),
    DEFAULT_RATE_DIV(DEFAULT_RATE_DIV),
    DEFAULT_CPOL(DEFAULT_CPOL),
    DEFAULT_CPHA(DEFAULT_CPHA)
) dut ( .clk(clk), .rstn(rstn), .up_rreq(up_rreq), .up_rack(up_rack), .up_r

```

Device under test, up\_spi\_master