

# AXIS\_UART



March 6, 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 . . . . .	5
<b>4 Simulation</b>	<b>6</b>
4.1 iverilog . . . . .	6
4.2 cocotb . . . . .	6
<b>5 Module Documentation</b>	<b>7</b>
5.1 axis_uart_rx . . . . .	8
5.2 axis_uart_tx . . . . .	10
5.3 axis_uart . . . . .	12
5.4 tb_cocotb_full python . . . . .	16
5.5 tb_cocotb_full verilog . . . . .	19
5.6 tb_cocotb_rx python . . . . .	22
5.7 tb_cocotb_rx verilog . . . . .	25
5.8 tb_cocotb_tx python . . . . .	28
5.9 tb_cocotb_tx verilog . . . . .	31

# 1 Usage

## 1.1 Introduction

Simple UART core for TTL rs232 software mode data communications. No hardware handshake. This contains its own internal baud rate generator that creates an enable to allow data output or sampling. Baud clock and aclk can be the same clock.

RTS/CTS is not implemented, it simply asserts it as if its always ready, and ignores CTS.

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

## 1.3 In a Project

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

# 2 Architecture

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

- **axis\_uart\_tx** Interface with UART TX and present the data over AXIS interface (see core for documentation).

- **axis\_uart\_rx** Interface with UART RX and present the data over AXIS interface (see core for documentation).

## 3 Building

The AXIS UART 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 fusesoc\_info File List

- src
  - src/axis\_uart.v
  - src/axis\_uart\_rx.v
  - src/axis\_uart\_tx.v
- tb
  - tb/tb\_uart.v
  - tb/tb\_uart\_rx.v
  - tb/tb\_uart\_tx.v
- tb\_cocotb\_full
  - 'tb/tb\_cocotb\_full.py': 'file\_type': 'user', 'copyto': '.'
  - 'tb/tb\_cocotb\_full.v': 'file\_type': 'verilogSource'
- tb\_cocotb\_rx
  - 'tb/tb\_cocotb\_rx.py': 'file\_type': 'user', 'copyto': '.'
  - 'tb/tb\_cocotb\_rx.v': 'file\_type': 'verilogSource'

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

### 3.3 Targets

#### 3.3.1 fusesoc\_info Targets

- default
 

Info: Default for IP intergration.
- sim
 

Info: Base simulation using icarus as default.
- sim\_rand\_data
 

Info: Use random data as sim input.
- sim\_rand\_ready\_rand\_data
 

Info: Use random data with a random ready as sim input.
- sim\_8bit\_count\_data
 

Info: Use counter data as sim input.
- sim\_rand\_ready\_8bit\_count\_data
 

Info: Use counter data with a random ready as sim input.
- sim\_rx
 

Info: Simulate only the rx block.
- sim\_tx
 

Info: Simulate only the tx block.
- sim\_cocotb\_full
 

Info: Cocotb unit tests
- sim\_cocotb\_rx
 

Info: Cocotb unit tests
- sim\_cocotb\_tx
 

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 TX/RX looped.
- **sim\_rx** Simulation of receive only.
- **sim\_tx** Simulation of transmit only.

### 4.2 cocotb

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

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

Each module has a cocotb based simulation. These use the cocotb extensions made by Alex. The two extensions used are cocotbext-axi and cocotbext-uart. These provide outside verification of the implementation. These tests consist of 3 different fusesoc targets.

- **sim\_cocotb\_full** Standard simulation of TX/RX passing data to and from cocotbexts.
- **sim\_cocotb\_rx** Simulation of data receive using cocotbext.
- **sim\_cocotb\_tx** Simulation of data transmit using cocotbext.

Then you must use the cocotb sim target. The targets above can be run with various bus and fifo parameters.

```
$ fusesoc run --target AFRL:device_converter:axis_uart:1.0.0
```

## 5 Module Documentation

- **axis\_uart\_tx** Interfaces AXIS to the UART transmit line.
- **axis\_uart\_rx** Interfaces AXIS to the UART receive line.
- **axis\_uart** Wrapper for all of the above modules to create a singular device to interface with.



## axis\_uart\_rx.v

---

### AUTHORS

---

JAY CONVERTINO

---

### DATES

---

2021/06/24

---

### INFORMATION

---

#### Brief

---

UART RX to AXIS bus.

#### License MIT

---

Copyright 2021 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\_uart\_rx

---

```
module axis_uart_rx #(
    parameter
    PARITY_ENA
    =
    0,
    parameter
    PARITY_TYPE
    =
    0,
    parameter
    STOP_BITS
    =
    1,
    parameter
```

```

DATA_BITS
=
8,
parameter
DELAY
=
0
) ( input aclk, input arstn, output parity_err, output frame_err, output [DA

```

AXIS UART, simple UART with AXI Streaming interface.

## Parameters

<b>PARITY_ENA</b> parameter	Enable Parity for the data in and out.
<b>PARITY_TYPE</b> parameter	Set the parity type, 0 = even, 1 = odd, 2 = mark, 3 = space.
<b>STOP_BITS</b> parameter	Number of stop bits, 0 to crazy non-standard amounts.
<b>DATA_BITS</b> parameter	Number of data bits, 1 to crazy non-standard amounts.
<b>DELAY</b> parameter	Delay in rx data input.

## Ports

<b>aclk</b>	Clock for AXIS
<b>arstn</b>	Negative reset for AXIS
<b>parity_err</b>	Indicates error with parity check (active high)
<b>frame_err</b>	Indicates error with frame (active high)
<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>uart_clk</b>	Clock used for BAUD rate generation
<b>uart_rstn</b>	Negative reset for UART, for anything clocked on uart_clk
<b>uart_ena</b>	Enable UART data processing from RX.
<b>uart_hold</b>	Output to hold clock till in receive state.
<b>rx</b>	receive for UART (input from TX)

## axis\_uart\_tx.v

---

### AUTHORS

---

JAY CONVERTINO

---

### DATES

---

2021/06/24

---

### INFORMATION

---

#### Brief

---

UART TX from AXIS bus.

#### License MIT

---

Copyright 2021 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\_uart\_tx

---

```
module axis_uart_tx #(
    parameter
    PARITY_ENA
    =
    0,
    parameter
    PARITY_TYPE
    =
    1,
    parameter
    STOP_BITS
    =
    1,
    parameter
```

```

DATA_BITS
=
8,
parameter
DELAY
=
0
) ( input aclk, input arstn, input [DATA_BITS-1:0] s_axis_tdata, input s_axi

```

AXIS UART TX, simple UART TX from AXI Streaming interface.

## Parameters

<b>PARITY_ENA</b> parameter	Enable Parity for the data in and out.
<b>PARITY_TYPE</b> parameter	Set the parity type, 0 = even, 1 = odd, 2 = mark, 3 = space.
<b>STOP_BITS</b> parameter	Number of stop bits, 0 to crazy non-standard amounts.
<b>DATA_BITS</b> parameter	Number of data bits, 1 to crazy non-standard amounts.
<b>DELAY</b> parameter	Delay in tx data output. Delays the time to output of the data.

## 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>uart_clk</b>	Clock used for BAUD rate generation
<b>uart_rstn</b>	Negative reset for UART, for anything clocked on uart_clk
<b>uart_ena</b>	When active high enable UART transmit state.
<b>txd</b>	transmit for UART (output to RX)

## axis\_uart.v

---

### AUTHORS

---

JAY CONVERTINO

---

### DATES

---

2021/06/24

---

### INFORMATION

---

#### Brief

---

Core for interfacing with simple UART communications. Output is always the size of DATA\_BITS.

#### License MIT

---

Copyright 2021 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\_uart

---

```
module axis_uart #(
    parameter
    BAUD_CLOCK_SPEED
    =
    2000000,
    parameter
    BAUD_RATE
    =
    2000000,
    parameter
    PARITY_ENA
    =
    0,
    parameter
```

```

PARITY_TYPE
=
0,
parameter
STOP_BITS
=
1,
parameter
DATA_BITS
=
8,
parameter
RX_DELAY
=
0,
parameter
RX_BAUD_DELAY
=
0,
parameter
TX_DELAY
=
0,
parameter
TX_BAUD_DELAY
=
0
) ( input aclk, input arstn, output parity_err, output frame_err, input [DA

```

AXIS UART, simple UART with AXI Streaming interface.

## Parameters

<b>BAUD_CLOCK_SPEED</b> parameter	This is the aclk frequency in Hz
<b>BAUD_RATE</b> parameter	Serial Baud, this can be any value including non-standard.
<b>PARITY_ENA</b> parameter	Enable Parity for the data in and out.
<b>PARITY_TYPE</b> parameter	Set the parity type, 0 = even, 1 = odd, 2 = mark, 3 = space.
<b>STOP_BITS</b> parameter	Number of stop bits, 0 to crazy non-standard amounts.
<b>DATA_BITS</b> parameter	Number of data bits, 1 to crazy non-standard amounts.
<b>RX_DELAY</b> parameter	Delay in rx data input.
<b>RX_BAUD_DELAY</b> parameter	Delay in rx baud enable. This will delay when we sample a bit (default is midpoint when rx delay is 0).
<b>TX_DELAY</b> parameter	Delay in tx data output. Delays the time to output of the data.
<b>TX_BAUD_DELAY</b> parameter	Delay in tx baud enable. This will delay the time the bit output starts.

## Ports

<b>aclk</b>	Clock for AXIS
<b>arstn</b>	Negative reset for AXIS
<b>parity_err</b>	Indicates error with parity check (active high)

<b>frame_err</b>	Indicates error with frame (active high)
<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>uart_clk</b>	Clock used for BAUD rate generation
<b>uart_rstn</b>	Negative reset for UART, for anything clocked on uart_clk
<b>tx</b>	transmit for UART (output to RX)
<b>rx</b>	receive for UART (input from TX)
<b>rts</b>	request to send is a loop with CTS
<b>cts</b>	clear to send is a loop with RTS

## INSTANTIATED MODULES

---

### uart\_baud\_gen\_tx

---

```
mod_clock_ena_gen #(
    CLOCK_SPEED(BAUD_CLOCK_SPEED),
    START_AT_ZERO(1),
    DELAY(TX_BAUD_DELAY)
) uart_baud_gen_tx ( .clk(uart_clk), .rstn(uart_rstn), .hold(1'b0), .rate(BA
```

Generates TX BAUD rate for UART modules using modulo divide method.

### uart\_baud\_gen\_rx

---

```
mod_clock_ena_gen #(
    CLOCK_SPEED(BAUD_CLOCK_SPEED),
    START_AT_ZERO(0),
    DELAY(RX_BAUD_DELAY)
) uart_baud_gen_rx ( .clk(uart_clk), .rstn(uart_rstn), .hold(uart_hold_rx),
```

Generates RX BAUD rate for UART modules using modulo divide method.

### uart\_tx

---

```
axis_uart_tx #(
    PARITY_ENA(PARITY_ENA),
    PARITY_TYPE(PARITY_TYPE),
```

```

    STOP_BITS(STOP_BITS),
    DATA_BITS(DATA_BITS),
    DELAY(TX_DELAY)
) uart_tx ( .aclk(aclk), .arstn(arstn), .s_axis_tdata(s_axis_tdata), .s_axis

```

Produces transmit data for tx UART from AXIS.

## uart\_rx

```

axis_uart_rx #(
    PARITY_ENA(PARITY_ENA),
    PARITY_TYPE(PARITY_TYPE),
    STOP_BITS(STOP_BITS),
    DATA_BITS(DATA_BITS),
    DELAY(RX_DELAY)
) uart_rx ( .aclk(aclk), .arstn(arstn), .parity_err(parity_err), .frame_err(

```

Consumes receive data for rx UART to AXIS.



# 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

---

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

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

### 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

---

2025/01/21

---

### 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
    BAUD_CLOCK_SPEED
    =
    2000000,
  parameter
    BAUD_RATE
    =
    2000000,
  parameter
    PARITY_ENA
    =
    0,
  parameter
```

```

PARITY_TYPE
=
0,
parameter
STOP_BITS
=
1,
parameter
DATA_BITS
=
8,
parameter
RX_DELAY
=
0,
parameter
RX_BAUD_DELAY
=
0,
parameter
TX_DELAY
=
0,
parameter
TX_BAUD_DELAY
=
0
) ( input aclk, input arstn, output parity_err, output frame_err, input [DA

```

Test bench for axis uart.

## Parameters

<b>BAUD_CLOCK_SPEED</b> parameter	This is the aclk frequency in Hz
<b>BAUD_RATE</b> parameter	Serial Baud, this can be any value including non-standard.
<b>PARITY_ENA</b> parameter	Enable Parity for the data in and out.
<b>PARITY_TYPE</b> parameter	Set the parity type, 0 = even, 1 = odd, 2 = mark, 3 = space.
<b>STOP_BITS</b> parameter	Number of stop bits, 0 to crazy non-standard amounts.
<b>DATA_BITS</b> parameter	Number of data bits, 1 to crazy non-standard amounts.
<b>RX_DELAY</b> parameter	Delay in rx data input.
<b>RX_BAUD_DELAY</b> parameter	Delay in rx baud enable. This will delay when we sample a bit (default is midpoint when rx delay is 0).
<b>TX_DELAY</b> parameter	Delay in tx data output. Delays the time to output of the data.
<b>TX_BAUD_DELAY</b> parameter	Delay in tx baud enable. This will delay the time the bit output starts.

## Ports

<b>aclk</b>	Clock for AXIS
<b>arstn</b>	Negative reset for AXIS
<b>parity_err</b>	Indicates error with parity check (active high)

<b>frame_err</b>	Indicates error with frame (active high)
<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>uart_clk</b>	Clock used for BAUD rate generation
<b>uart_rstn</b>	Negative reset for UART, for anything clocked on uart_clk
<b>tx</b>	transmit for UART (output to RX)
<b>rx</b>	receive for UART (input from TX)
<b>rts</b>	request to send is a loop with CTS
<b>cts</b>	clear to send is a loop with RTS

## INSTANTIATED MODULES

---

### dut

---

```
axis_uart #(
    BAUD_CLOCK_SPEED(BAUD_CLOCK_SPEED),
    BAUD_RATE(BAUD_RATE),
    PARITY_ENA(PARITY_ENA),
    PARITY_TYPE(PARITY_TYPE),
    STOP_BITS(STOP_BITS),
    DATA_BITS(DATA_BITS),
    RX_DELAY(RX_DELAY),
    RX_BAUD_DELAY(RX_BAUD_DELAY),
    TX_DELAY(TX_DELAY),
    TX_BAUD_DELAY(TX_BAUD_DELAY)
) dut ( .aclk(aclk), .arstn(arstn), .parity_err(parity_err), .frame_err(frame_err),
    .tx(tx), .rx(rx), .rts(rts), .cts(cts) )
```

Device under test, axis\_uart

# 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

---

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

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

### 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\_coctb.v

---

## AUTHORS

---

JAY CONVERTINO

---

## DATES

---

2025/01/23

---

## 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
  PARITY_ENA
  =
  0,
  parameter
  PARITY_TYPE
  =
  0,
  parameter
  STOP_BITS
  =
  1,
  parameter
```

```

DATA_BITS
=
8,
parameter
DELAY
=
0
) ( input aclk, input arstn, output parity_err, output frame_err, output [DA

```

Test bench for axis\_uart\_rx.

## Parameters

<b>PARITY_ENA</b> parameter	Enable Parity for the data in and out.
<b>PARITY_TYPE</b> parameter	Set the parity type, 0 = even, 1 = odd, 2 = mark, 3 = space.
<b>STOP_BITS</b> parameter	Number of stop bits, 0 to crazy non-standard amounts.
<b>DATA_BITS</b> parameter	Number of data bits, 1 to crazy non-standard amounts.
<b>DELAY</b> parameter	Delay in rx data input.

## Ports

<b>aclk</b>	Clock for AXIS
<b>arstn</b>	Negative reset for AXIS
<b>parity_err</b>	Indicates error with parity check (active high)
<b>frame_err</b>	Indicates error with frame (active high)
<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>uart_clk</b>	Clock used for BAUD rate generation
<b>uart_rstn</b>	Negative reset for UART, for anything clocked on uart_clk
<b>uart_ena</b>	Enable UART data processing from RX.
<b>uart_hold</b>	Output to hold clock till in receive state.
<b>rx</b>	receive for UART (input from TX)

## INSTANTIATED MODULES

---

### dut

```

axis_uart_rx #(
    PARITY_ENA(PARITY_ENA),
    PARITY_TYPE(PARITY_TYPE),
    STOP_BITS(STOP_BITS),
    DATA_BITS(DATA_BITS),
    DELAY(DELAY)

```

```
|) dut ( .aclk(aclk), .arstn(arstn), .parity_err(parity_err), .frame_err(frame_err)
```

---

Device under test, axis\_uart\_rx

# 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

---

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

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

### 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\_coctb.v

---

### AUTHORS

---

JAY CONVERTINO

---

### DATES

---

2024/12/10

---

### 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
  PARITY_ENA
  =
  0,
  parameter
  PARITY_TYPE
  =
  1,
  parameter
  STOP_BITS
  =
  1,
  parameter
```



```

DATA_BITS
=
8,
parameter
DELAY
=
0
) ( input aclk, input arstn, input [DATA_BITS-1:0] s_axis_tdata, input s_axi

```

Test bench for AXIS UART TX, simple UART TX from AXI Streaming interface.

## Parameters

<b>PARITY_ENA</b> parameter	Enable Parity for the data in and out.
<b>PARITY_TYPE</b> parameter	Set the parity type, 0 = even, 1 = odd, 2 = mark, 3 = space.
<b>STOP_BITS</b> parameter	Number of stop bits, 0 to crazy non-standard amounts.
<b>DATA_BITS</b> parameter	Number of data bits, 1 to crazy non-standard amounts.
<b>DELAY</b> parameter	Delay in tx data output. Delays the time to output of the data.

## 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>uart_clk</b>	Clock used for BAUD rate generation
<b>uart_rstn</b>	Negative reset for UART, for anything clocked on uart_clk
<b>uart_ena</b>	When active high enable UART transmit state.
<b>txd</b>	transmit for UART (output to RX)

## INSTANTIATED MODULES

### dut

```

axis_uart_tx #(
    PARITY_ENA(0),
    PARITY_TYPE(1),
    STOP_BITS(1),
    DATA_BITS(8),
    DELAY(0)
) dut ( .aclk(aclk), .arstn(arstn), .s_axis_tdata(s_axis_tdata), .s_axis_tva

```

Device under test, axis\_uart\_tx

