

# 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
    =
    20000000,
  parameter
    BAUD_RATE
    =
    20000000,
  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,
parameter
BUS_WIDTH
=
1
) ( input aclk, input arstn, output parity_err, output frame_err, input [BUS
```

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.
<b>BUS_WIDTH</b> parameter	AXIS data bus width in bytes.

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),
    BUS_WIDTH(BUS_WIDTH)
) 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