

tb_cocotb_up.v

AUTHORS

JAY CONVERTINO

DATES

2025/04/01

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
  CLOCK_SPEED
  =
  100000000,
  parameter
```

```

    BAUD_RATE
    =
    115200,
    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 clk, input rstn, input up_rreq, output up_rack, input [ADDRESS_WIDTH-1:0] up_addr, output [DATA_WIDTH-1:0] up_data )

```

uP UART testbench

Parameters

ADDRESS_WIDTH parameter	Width of the axi address bus
BUS_WIDTH parameter	Number of bytes for the data bus.
CLOCK_SPEED parameter	This is the aclk frequency in Hz
BAUD_RATE parameter	Serial Baud, this can be any value including non-standard.
PARITY_ENA parameter	Enable Parity for the data in and out.
PARITY_TYPE parameter	Set the parity type, 0 = even, 1 = odd, 2 = mark, 3 = space.
STOP_BITS parameter	Number of stop bits, 0 to crazy non-standard amounts.
DATA_BITS parameter	Number of data bits, 1 to crazy non-standard amounts.
RX_DELAY parameter	Delay in rx data input.
RX_BAUD_DELAY parameter	Delay in rx baud enable. This will delay when we sample a bit (default is midpoint when rx delay is 0).

TX_DELAY parameter	Delay in tx data output. Delays the time to output of the data.
TX_BAUD_DELAY parameter	Delay in tx baud enable. This will delay the time the bit output starts.

Ports

clk	Clock for all devices in the core
rstn	Negative reset
up_rreq	uP bus read request
up_rack	uP bus read ack
up_raddr	uP bus read address
up_rdata	uP bus read data
up_wreq	uP bus write request
up_wack	uP bus write ack
up_waddr	uP bus write address
up_wdata	uP bus write data
irq	Interrupt when data is received
tx	transmit for UART (output to RX)
rx	receive for UART (input from TX)
rts	request to send is a loop with CTS
cts	clear to send is a loop with RTS

INSTANTIATED MODULES

dut

```

up_uart #(
    ADDRESS_WIDTH(ADDRESS_WIDTH),
    BUS_WIDTH(BUS_WIDTH),
    CLOCK_SPEED(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 ( .clk(clk), .rstn(rstn), .up_rreq(up_rreq), .up_rack(up_rack), .up_raddr(up_raddr), .up_rdata(up_rdata), .up_wreq(up_wreq), .up_wack(up_wack), .up_waddr(up_waddr), .up_wdata(up_wdata), .irq(irq), .tx(tx), .rx(rx), .rts(rts), .cts(cts))
```

Device under test, up_uart