

# tb\_wishbone\_cocotb.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
    ADDRESS_WIDTH
    =
    32,
  parameter
    BUS_WIDTH
    =
    4,
  parameter
    DEPTH
```

```

    =
    512,
    parameter
    RAM_TYPE
    =
    "block",
    parameter
    HEX_FILE
    =
    ""
) ( 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_bte, input s_wb_cti, input s_wb_ack, input s_wb_data_o, input s_wb_err )

```

Test bench for wishbone.

## Parameters

<b>ADDRESS_WIDTH</b> parameter	Width of the axi address bus in bits.
<b>BUS_WIDTH</b> parameter	Bus width for data paths in bytes.
<b>DEPTH</b> parameter	Depth of the RAM in terms of data width words.
<b>RAM_TYPE</b> parameter	Used to set the ram_style attribute.
<b>HEX_FILE</b> parameter	Hex file to write to RAM.

## 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_bte</b>	Burst Type Extension
<b>s_wb_cti</b>	Cycle Type
<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

## INSTANTIATED MODULES

---

### dut

```

wishbone_classic_block_ram #(
    ADDRESS_WIDTH(ADDRESS_WIDTH),
    .
    .

```

```
BUS_WIDTH(BUS_WIDTH),  
DEPTH(DEPTH),  
RAM_TYPE(RAM_TYPE),  
HEX_FILE(HEX_FILE)  
) dut ( .clk(clk), .rst(rst), .s_wb_cyc(s_wb_cyc), .s_wb_stb(s_wb_stb), .s_v
```

Device under test, wishbone\_classic\_block\_ram