

wishbone_standard_uart_lite.v

AUTHORS

JAY CONVERTINO

DATES

2024/02/29

INFORMATION

Brief

Wishbone Lite UART core

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.

wishbone_standard_uart_lite

Wishbone Standard based uart device.

Parameters

ADDRESS_WIDTH	Width of the address bus in bits.
BUS_WIDTH	Width of the data bus in bytes.
CLOCK_SPEED	This is the aclk frequency in Hz
BAUD_RATE	Serial Baud, this can be any value including non-standard.
PARITY_TYPE	Set the parity type, 0 = none, 1 = even, 2 = odd, 3 = mark, 4 = space.
STOP_BITS	Number of stop bits, 0 to crazy non-standard amounts.
DATA_BITS	Number of data bits, 1 to 8
RX_BAUD_DELAY	Delay in rx baud enable. This will delay when we sample a bit (default is midpoint

when rx delay is 0).

TX_BAUD_DELAY Delay in tx baud enable. This will delay the time the bit output starts.

Ports

clk	Clock for all devices in the core
rst	Positive reset
s_wb_cyc	Bus Cycle in process
s_wb_stb	Valid data transfer cycle
s_wb_we	Active High write, low read
s_wb_addr	Bus address
s_wb_data_i	Input data
s_wb_sel	Device Select
s_wb_ack	Bus transaction terminated
s_wb_data_o	Output data
s_wb_err	Active high when a bus error is present
irq	Interrupt when data is received
tx	transmit for UART (output to RX)
rx	receive for UART (input from TX)

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_uart_lite

```
up_uart_lite #(  
    ADDRESS_WIDTH(ADDRESS_WIDTH),  
    BUS_WIDTH(BUS_WIDTH),  
    CLOCK_SPEED(CLOCK_SPEED),
```

```

BAUD_RATE(BAUD_RATE),
PARITY_TYPE(PARITY_TYPE),
STOP_BITS(STOP_BITS),
DATA_BITS(DATA_BITS),
RX_BAUD_DELAY(RX_BAUD_DELAY),
TX_BAUD_DELAY(TX_BAUD_DELAY)
) inst_up_uart_lite ( .clk(clk), .rstn(~rst), .up_rreq(up_rreq), .up_rack(up_rack),

```

Module instance of up_uart creating a Logic wrapper for uart axis bus cores to interface with uP bus.