

# axi\_lite\_uart.v

---

## AUTHORS

---

JAY CONVERTINO

---

## DATES

---

2024/02/29

---

## INFORMATION

---

### Brief

---

AXI Lite UART is a core for interfacing with UART devices.

### 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.

## axi\_lite\_uart

---

```
module axi_lite_uart #(
  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 aclk, input arstn, input s_axi_awvalid, input [ADDRESS_WIDTH-1:0]

```

AXI Lite based uart device.

## Parameters

<b>ADDRESS_WIDTH</b> parameter	Width of the axi address bus
<b>BUS_WIDTH</b> parameter	Number of bytes for the data bus
<b>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 all devices in the core
<b>arstn</b>	Negative reset
<b>s_axi_awvalid</b>	Axi Lite aw valid
<b>s_axi_awaddr</b>	Axi Lite aw addr
<b>s_axi_awprot</b>	Axi Lite aw prot
<b>s_axi_awready</b>	Axi Lite aw ready
<b>s_axi_wvalid</b>	Axi Lite w valid
<b>s_axi_wdata</b>	Axi Lite w data
<b>s_axi_wstrb</b>	Axi Lite w strb
<b>s_axi_wready</b>	Axi Lite w ready
<b>s_axi_bvalid</b>	Axi Lite b valid
<b>s_axi_bresp</b>	Axi Lite b resp
<b>s_axi_bready</b>	Axi Lite b ready
<b>s_axi_arvalid</b>	Axi Lite ar valid
<b>s_axi_araddr</b>	Axi Lite ar addr
<b>s_axi_arprot</b>	Axi Lite ar prot
<b>s_axi_arready</b>	Axi Lite ar ready
<b>s_axi_rvalid</b>	Axi Lite r valid
<b>s_axi_rdata</b>	Axi Lite r data
<b>s_axi_rresp</b>	Axi Lite r resp
<b>s_axi_rready</b>	Axi Lite r ready
<b>irq</b>	Interrupt when data is received
<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

## 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\_axi

---

```
up_axi #(  
  

```

```

        AXI_ADDRESS_WIDTH(ADDRESS_WIDTH)
    ) inst_up_axi ( .up_rstn (arstn), .up_clk (aclk), .up_axi_awvalid(s_axi_awv

```

Module instance of up\_axi for the AXI Lite bus to the uP bus.

## inst\_up\_uart

```

    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)
    ) inst_up_uart ( .clk(aclk), .rstn(arstn), .up_rreq(up_rreq), .up_rack(up_ra

```

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