AXIS_UART

April 24, 2025

Jay Convertino

Contents

1 Usage

1.1 Introduction

Simple UART core for TTL rs232 software mode data communications. No hardware handshake. This contains its own internal baud rate generator that creates an enable to allow data output or sampling. Baud clock and aclk can be the same clock.

RTS/CTS is not implemented, it simply asserts it as if its always ready, and ignores CTS.

1.2 Dependencies

The following are the dependencies of the cores.

- fusesoc 2.X
- iverilog (simulation)
- cocotb (simulation)

1.2.1 fusesoc_info Depenecies

- dep
  - AFRL:clock:mod_clock_ena_gen:1.1.1
  - AFRL:utility:helper:1.0.0
- dep_tb
  - AFRL:simulation:axis_stimulator
  - AFRL:utility:sim_helper

1.3 In a Project

This core connects a UART to the AXIS bus. Meaning this is a streaming device only. Connect the RX/TX to the UART in question and connect the AXIS to its intended endpoints.

2 Architecture

This core is made up of other cores that are documented in detail in there source. The cores this is made up of are the,

- **axis_uart_tx** Interface with UART TX and present the data over AXIS interface (see core for documentation).

- **axis_uart_rx** Interface with UART RX and present the data over AXIS interface (see core for documentation).

3 Building

The AXIS UART is written in Verilog 2001. It should synthesize in any modern FPGA software. The core comes as a fusesoc packaged core and can be included in any other core. Be sure to make sure you have meet the dependencies listed in the previous section.

3.1 fusesoc

Fusesoc is a system for building FPGA software without relying on the internal project management of the tool. Avoiding vendor lock in to Vivado or Quartus. These cores, when included in a project, can be easily integrated and targets created based upon the end developer needs. The core by itself is not a part of a system and should be integrated into a fusesoc based system. Simulations are setup to use fusesoc and are a part of its targets.

3.2 Source Files

3.2.1 fusesoc_info File List

- src
  - src/axis_uart.v
  - src/axis_uart_rx.v
  - src/axis_uart_tx.v
- tb
  - tb/tb_uart.v
  - tb/tb_uart_rx.v
  - tb/tb_uart_tx.v
- tb_cocotb_full
  - 'tb/tb_cocotb_full.py': 'file_type': 'user', 'copyto': '.'
  - 'tb/tb_cocotb_full.v': 'file_type': 'verilogSource'
- tb_cocotb_rx
  - 'tb/tb_cocotb_rx.py': 'file_type': 'user', 'copyto': '.'
  - 'tb/tb_cocotb_rx.v': 'file_type': 'verilogSource'

- tb_cocotb_tx
  - 'tb/tb_cocotb_tx.py': 'file_type': 'user', 'copyto': '.'
  - 'tb/tb_cocotb_tx.v': 'file_type': 'verilogSource'

3.3 Targets

3.3.1 fusesoc_info Targets

- default

  Info: Default for IP intergration.

- sim

  Info: Base simulation using icarus as default.

- sim_rand_data

  Info: Use random data as sim input.

- sim_rand_ready_rand_data

  Info: Use random data with a random ready as sim input.

- sim_8bit_count_data

  Info: Use counter data as sim input.

- sim_rand_ready_8bit_count_data

  Info: Use counter data with a random ready as sim input.

- sim_rx

  Info: Simulate only the rx block.

- sim_tx

  Info: Simulate only the tx block.

- sim_cocotb_full

  Info: Cocotb unit tests

- sim_cocotb_rx

  Info: Cocotb unit tests

- sim_cocotb_tx

  Info: Cocotb unit tests

3.4 Directory Guide

Below highlights important folders from the root of the directory.

1. **docs** Contains all documentation related to this project.
    - **manual** Contains user manual and github page that are generated from the latex sources.
2. **src** Contains source files for the core
3. **tb** Contains test bench files for iverilog and cocotb
    - **cocotb** testbench files

4 Simulation

There are a few different simulations that can be run for this core.

4.1 iverilog

iverilog is used for simple test benches for quick verification, visually, of the core.

- **sim** Standard simulation of TX/RX looped.
- **sim_rx** Simulation of receive only.
- **sim_tx** Simulation of transmit only.

4.2 cocotb

To use the cocotb tests you must install the following python libraries.

```
$ pip install cocotb
$ pip install cocotbext-axi
```

Each module has a cocotb based simulation. These use the cocotb extensions made by Alex. The two extensions used are cocotbext-axi and cocotbext-uart. These provide outside verification of the implimentation. These tests consist of 3 different fusesoc targets.

- **sim_cocotb_full** Standard simulation of TX/RX passing data to and from cocotbexts.
- **sim_cocotb_rx** Simulation of data receive using cocotbext.
- **sim_cocotb_tx** Simulation of data transmit using cocotbext.

Then you must use the cocotb sim target. The targets above can be run with various bus and fifo parameters.

```
$ fusesoc run --target AFRL:device_converter:axis_uart:1.0.0
```

5 Module Documentation

- **axis_uart_tx** Interfaces AXIS to the UART transmit line.
- **axis_uart_rx** Interfaces AXIS to the UART receive line.
- **axis_uart** Wrapper for all of the above modules to create a singular device to interface with.

axis_uart_rx.v

AUTHORS

JAY CONVERTINO

DATES

2021/06/24

INFORMATION

Brief

UART RX to AXIS bus.

License MIT

axis_uart_rx

```
module axis_uart_rx #(
parameter
PARITY_ENA
 =
0,
parameter
PARITY_TYPE
 =
0,
parameter
STOP_BITS
 =
1,
parameter
```

```
DATA_BITS
 =
8,
parameter
DELAY
 =
0
) ( input aclk, input arstn, output parity_err, output frame_err, output [DA
```

AXIS UART, simple UART with AXI Streaming interface.

Parameters

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.
DELAY parameter	Delay in rx data input.

Ports

aclk	Clock for AXIS
arstn	Negative reset for AXIS
parity_err	Indicates error with parity check (active high)
frame_err	Indicates error with frame (active high)
m_axis_tdata	Output data from UART RX
m_axis_tvalid	When active high the output data is valid
m_axis_tready	When set active high the output device is ready for data.
uart_clk	Clock used for BAUD rate generation
uart_rstn	Negative reset for UART, for anything clocked on uart_clk
uart_ena	Enable UART data processing from RX.
uart_hold	Output to hold back clock in reset state till uart is in receive state.
rxd	receive for UART (input from TX)

axis_uart_tx.v

AUTHORS

JAY CONVERTINO

DATES

2021/06/24

INFORMATION

Brief

UART TX from AXIS bus.

License MIT

axis_uart_tx

```
module axis_uart_tx #(
parameter
PARITY_ENA
 =
0,
parameter
PARITY_TYPE
 =
1,
parameter
STOP_BITS
 =
1,
parameter
```

```
 DATA_BITS
  =
 8,
 parameter
 DELAY
  =
 0
) ( input aclk, input arstn, input [DATA_BITS-1:0] s_axis_tdata, input s_axi
```

AXIS UART TX, simple UART TX from AXI Streaming interface.

Parameters

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.
DELAY parameter	Delay in tx data output. Delays the time to output of the data.

Ports

aclk	Clock for AXIS
arstn	Negative reset for AXIS
s_axis_tdata	Input data for UART TX.
s_axis_tvalid	When set active high the input data is valid
s_axis_tready	When active high the device is ready for input data.
uart_clk	Clock used for BAUD rate generation
uart_rstn	Negative reset for UART, for anything clocked on uart_clk
uart_ena	When active high enable UART transmit state.
txd	transmit for UART (output to RX)

axis_uart.v

AUTHORS

JAY CONVERTINO

DATES

2021/06/24

INFORMATION

Brief

Core for interfacing with simple UART communications. Output is always the size of DATA_BITS.

License MIT

axis_uart

```
module axis_uart #(
parameter
BAUD_CLOCK_SPEED
 =
2000000,
parameter
BAUD_RATE
 =
2000000,
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, output parity_err, output frame_err, input [DA
```

AXIS UART, simple UART with AXI Streaming interface.

Parameters

BAUD_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

aclk	Clock for AXIS
arstn	Negative reset for AXIS
parity_err	Indicates error with parity check (active high)

frame_err	Indicates error with frame (active high)
s_axis_tdata	Input data for UART TX.
s_axis_tvalid	When set active high the input data is valid
s_axis_tready	When active high the device is ready for input data.
m_axis_tdata	Output data from UART RX
m_axis_tvalid	When active high the output data is valid
m_axis_tready	When set active high the output device is ready for data.
uart_clk	Clock used for BAUD rate generation
uart_rstn	Negative reset for UART, for anything clocked on uart_clk
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

uart_baud_gen_tx

```
 mod_clock_ena_gen #(
                                                                                .
 CLOCK_SPEED(BAUD_CLOCK_SPEED),
                                                                                .
 DELAY(TX_BAUD_DELAY)
) uart_baud_gen_tx ( .clk(uart_clk), .rstn(uart_rstn), .start0(1'b1), .clr(1
```

Generates TX BAUD rate for UART modules using modulo divide method.

uart_baud_gen_rx

```
 mod_clock_ena_gen #(
                                                                                .
 CLOCK_SPEED(BAUD_CLOCK_SPEED),
                                                                                .
 DELAY(RX_BAUD_DELAY)
) uart_baud_gen_rx ( .clk(uart_clk), .rstn(uart_rstn), .start0(1'b0), .clr(u
```

Generates RX BAUD rate for UART modules using modulo divide method.

uart_tx

```
 axis_uart_tx #(
                                                                                .
 PARITY_ENA(PARITY_ENA),
                                                                                .
 PARITY_TYPE(PARITY_TYPE),
                                                                                .
 STOP_BITS(STOP_BITS),
                                                                                .
 DATA_BITS(DATA_BITS),
                                                                                .
```

```
 DELAY(TX_DELAY)
) uart_tx ( .aclk(aclk), .arstn(arstn), .s_axis_tdata(s_axis_tdata), .s_axis
```

Produces transmit data for tx UART from AXIS.

uart_rx

```
 axis_uart_rx #(
                                                                    .
 PARITY_ENA(PARITY_ENA),
                                                                    .
 PARITY_TYPE(PARITY_TYPE),
                                                                    .
 STOP_BITS(STOP_BITS),
                                                                    .
 DATA_BITS(DATA_BITS),
                                                                    .
 DELAY(RX_DELAY)
) uart_rx ( .aclk(aclk), .arstn(arstn), .parity_err(parity_err), .frame_err(
```

Consumes receive data for rx UART to AXIS.

tb_cocotb.py

AUTHORS

JAY CONVERTINO

DATES

2024/12/09

INFORMATION

Brief

Cocotb test bench

License MIT

FUNCTIONS

random_bool

```
def random_bool()
```

Return a infinte cycle of random bools

Returns: List

start_clock

```
def start_clock(
dut
)
```

Start the simulation clock generator.

Parameters

dut Device under test passed from cocotb test function

reset_dut

```
async def reset_dut(
dut
)
```

Cocotb coroutine for resets, used with await to make sure system is reset.

single_word

```
@cocotb.test()
async def single_word(
dut
)
```

Coroutine that is identified as a test routine. This routine tests for writing a single word, and then reading a single word.

Parameters

dut Device under test passed from cocotb.

in_reset

```
@cocotb.test()
async def in_reset(
dut
)
```

Coroutine that is identified as a test routine. This routine tests if device stays in unready state when in reset.

Parameters

dut Device under test passed from cocotb.

no_clock

```
@cocotb.test()
async def no_clock(
dut
)
```

Coroutine that is identified as a test routine. This routine tests if no ready when clock is lost and device is

left in reset.

Parameters

dut Device under test passed from cocotb.

tb_cocotb.v

AUTHORS

JAY CONVERTINO

DATES

2025/01/21

INFORMATION

Brief

Test bench wrapper for cocotb

License MIT

tb_cocotb

```
module tb_cocotb #(
parameter
BAUD_CLOCK_SPEED
 =
2000000,
parameter
BAUD_RATE
 =
2000000,
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, output parity_err, output frame_err, input [DA
```

Test bench for axis uart.

Parameters

BAUD_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

aclk	Clock for AXIS
arstn	Negative reset for AXIS
parity_err	Indicates error with parity check (active high)

frame_err	Indicates error with frame (active high)
s_axis_tdata	Input data for UART TX.
s_axis_tvalid	When set active high the input data is valid
s_axis_tready	When active high the device is ready for input data.
m_axis_tdata	Output data from UART RX
m_axis_tvalid	When active high the output data is valid
m_axis_tready	When set active high the output device is ready for data.
uart_clk	Clock used for BAUD rate generation
uart_rstn	Negative reset for UART, for anything clocked on uart_clk
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

```
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)
) dut ( .aclk(aclk), .arstn(arstn), .parity_err(parity_err), .frame_err(fra
```

Device under test, axis_uart

tb_cocotb.py

AUTHORS

JAY CONVERTINO

DATES

2024/12/09

INFORMATION

Brief

Cocotb test bench

License MIT

FUNCTIONS

random_bool

```
def random_bool()
```

Return a infinte cycle of random bools

Returns: List

start_clock

```
def start_clock(
dut
)
```

Start the simulation clock generator.

Parameters

dut Device under test passed from cocotb test function

reset_dut

```
async def reset_dut(
dut
)
```

Cocotb coroutine for resets, used with await to make sure system is reset.

single_word

```
@cocotb.test()
async def single_word(
dut
)
```

Coroutine that is identified as a test routine. This routine tests for writing a single word, and then reading a single word.

Parameters

dut Device under test passed from cocotb.

in_reset

```
@cocotb.test()
async def in_reset(
dut
)
```

Coroutine that is identified as a test routine. This routine tests if device stays in unready state when in reset.

Parameters

dut Device under test passed from cocotb.

no_clock

```
@cocotb.test()
async def no_clock(
dut
)
```

Coroutine that is identified as a test routine. This routine tests if no ready when clock is lost and device is

left in reset.

Parameters

dut Device under test passed from cocotb.

tb_coctb.v

AUTHORS

JAY CONVERTINO

DATES

2025/01/23

INFORMATION

Brief

Test bench wrapper for cocotb

License MIT

tb_cocotb

```
module tb_cocotb #(
parameter
PARITY_ENA
 =
0,
parameter
PARITY_TYPE
 =
0,
parameter
STOP_BITS
 =
1,
parameter
```

```
DATA_BITS
 =
8,
parameter
DELAY
 =
0
) ( input aclk, input arstn, output parity_err, output frame_err, output [DA
```

Test bench for axis_uart_rx.

Parameters

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.
DELAY parameter	Delay in rx data input.

Ports

aclk	Clock for AXIS
arstn	Negative reset for AXIS
parity_err	Indicates error with parity check (active high)
frame_err	Indicates error with frame (active high)
m_axis_tdata	Output data from UART RX
m_axis_tvalid	When active high the output data is valid
m_axis_tready	When set active high the output device is ready for data.
uart_clk	Clock used for BAUD rate generation
uart_rstn	Negative reset for UART, for anything clocked on uart_clk
uart_ena	Enable UART data processing from RX.
uart_hold	Output to hold clock till in receive state.
rxd	receive for UART (input from TX)

INSTANTIATED MODULES

dut

```
axis_uart_rx #(
                                                                    .
PARITY_ENA(PARITY_ENA),
                                                                    .
PARITY_TYPE(PARITY_TYPE),
                                                                    .
STOP_BITS(STOP_BITS),
                                                                    .
DATA_BITS(DATA_BITS),
                                                                    .
DELAY(DELAY)
```

```
) dut ( .aclk(aclk), .arstn(arstn), .parity_err(parity_err), .frame_err(fram
```

Device under test, axis_uart_rx

tb_cocotb.py

AUTHORS

JAY CONVERTINO

DATES

2024/12/09

INFORMATION

Brief

Cocotb test bench

License MIT

FUNCTIONS

random_bool

```
def random_bool()
```

Return a infinte cycle of random bools

Returns: List

start_clock

```
def start_clock(
dut
)
```

Start the simulation clock generator.

Parameters

dut Device under test passed from cocotb test function

reset_dut

```
async def reset_dut(
dut
)
```

Cocotb coroutine for resets, used with await to make sure system is reset.

single_word

```
@cocotb.test()
async def single_word(
dut
)
```

Coroutine that is identified as a test routine. This routine tests for writing a single word, and then reading a single word.

Parameters

dut Device under test passed from cocotb.

in_reset

```
@cocotb.test()
async def in_reset(
dut
)
```

Coroutine that is identified as a test routine. This routine tests if device stays in unready state when in reset.

Parameters

dut Device under test passed from cocotb.

no_clock

```
@cocotb.test()
async def no_clock(
dut
)
```

Coroutine that is identified as a test routine. This routine tests if no ready when clock is lost and device is

left in reset.

Parameters

dut Device under test passed from cocotb.

tb_coctb.v

AUTHORS

JAY CONVERTINO

DATES

2024/12/10

INFORMATION

Brief

Test bench wrapper for cocotb

License MIT

tb_cocotb

```
module tb_cocotb #(
parameter
PARITY_ENA
 =
0,
parameter
PARITY_TYPE
 =
1,
parameter
STOP_BITS
 =
1,
parameter
```

```
 DATA_BITS
  =
 8,
 parameter
 DELAY
  =
 0
) ( input aclk, input arstn, input [DATA_BITS-1:0] s_axis_tdata, input s_axi
```

Test bench for AXIS UART TX, simple UART TX from AXI Streaming interface.

Parameters

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.
DELAY parameter	Delay in tx data output. Delays the time to output of the data.

Ports

aclk	Clock for AXIS
arstn	Negative reset for AXIS
s_axis_tdata	Input data for UART TX.
s_axis_tvalid	When set active high the input data is valid
s_axis_tready	When active high the device is ready for input data.
uart_clk	Clock used for BAUD rate generation
uart_rstn	Negative reset for UART, for anything clocked on uart_clk
uart_ena	When active high enable UART transmit state.
txd	transmit for UART (output to RX)

INSTANTIATED MODULES

dut

```
 axis_uart_tx #(
                                                                    .
 PARITY_ENA(0),
                                                                    .
 PARITY_TYPE(1),
                                                                    .
 STOP_BITS(1),
                                                                    .
 DATA_BITS(8),
                                                                    .
 DELAY(0)
) dut ( .aclk(aclk), .arstn(arstn), .s_axis_tdata(s_axis_tdata), .s_axis_tva
```

Device under test, axis_uart_tx