



# **AXI4-Stream UART (Beta Release)**

Version 0.1

May 9, 2023

## Copyright

Copyright © 2021 Rapid Silicon. All rights reserved. This document may not, in whole or part, be reproduced, modified, distributed, or publicly displayed without prior written consent from Rapid Silicon ("Rapid Silicon").

## Trademarks

All Rapid Silicon trademarks are as listed at [www.rapidsilicon.com](http://www.rapidsilicon.com). Synopsys and Synplify Pro are trademarks of Synopsys, Inc. Aldec and Active-HDL are trademarks of Aldec, Inc. Modelsim and Questa are trademarks or registered trademarks of Siemens Industry Software Inc. or its subsidiaries in the United States or other countries. All other trademarks are the property of their respective owners.

## Disclaimers

NO WARRANTIES: THE INFORMATION PROVIDED IN THIS DOCUMENT IS "AS IS" WITHOUT ANY EXPRESS OR IMPLIED WARRANTY OF ANY KIND INCLUDING WARRANTIES OF ACCURACY, COMPLETENESS, MERCHANTABILITY, NONINFRINGEMENT OF INTELLECTUAL PROPERTY, OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL RAPID SILICON OR ITS SUPPLIERS BE LIABLE FOR ANY DAMAGES WHATSOEVER (WHETHER DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL, INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OF OR INABILITY TO USE THE INFORMATION PROVIDED IN THIS DOCUMENT, EVEN IF RAPID SILICON HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. BECAUSE SOME JURISDICTIONS PROHIBIT THE EXCLUSION OR LIMITATION OF CERTAIN LIABILITY, SOME OF THE ABOVE LIMITATIONS MAY NOT APPLY TO YOU.

Rapid Silicon may make changes to these materials, specifications, or information, or to the products described herein, at any time without notice. Rapid Silicon makes no commitment to update this documentation. Rapid Silicon reserves the right to discontinue any product or service without notice and assumes no obligation to correct any errors contained herein or to advise any user of this document of any correction if such be made. Rapid Silicon recommends its customers obtain the latest version of the relevant information to establish that the information being relied upon is current and before ordering any products.

# Contents

|   |           |
|---|-----------|
| <b>IP Summary</b>                         | <b>3</b>  |
| Introduction . . . . .                    | 3         |
| Features . . . . .                        | 3         |
| <b>Overview</b>                           | <b>4</b>  |
| AXIS UART . . . . .                       | 4         |
| <b>IP Specification</b>                   | <b>5</b>  |
| Standards . . . . .                       | 5         |
| IP Support Details . . . . .              | 6         |
| Port List . . . . .                       | 6         |
| Parameters . . . . .                      | 7         |
| Resource Utilization . . . . .            | 7         |
| <b>Design Flow</b>                        | <b>8</b>  |
| IP Customization and Generation . . . . . | 8         |
| Parameters Customization . . . . .        | 9         |
| <b>Example Design</b>                     | <b>10</b> |
| Overview . . . . .                        | 10        |
| Simulating the Example Design . . . . .   | 10        |
| Synthesis and PR . . . . .                | 10        |
| <b>Test Bench</b>                         | <b>11</b> |
| <b>Release</b>                            | <b>12</b> |
| Release History . . . . .                 | 12        |

# IP Summary

## Introduction

The AXIS UART is designed to be used with the AXIS bus, which provides a high-speed, low-latency data path between the UART and other components in the system. The AXIS UART is often used to interface with other peripherals such as memory or other processors, or to provide a serial communication link to external devices such as a PC or another microcontroller. UARTs typically include a transmitter and receiver and can be used for a wide range of applications. This is an AXI-Stream compliant UART IP that can be integrated in a number of systems.

## Features

- Configurable data width for the AXIL bus.
- Independent transmission and reception modules.
- Supports the AXI4-Stream interface specification.

# Overview

## AXIS UART

AXI Stream UART is a type of Universal Asynchronous Receiver-Transmitter (UART) interface that is designed to work with the Advanced Microcontroller Bus Architecture (AMBA) AXI Stream protocol. It is commonly used in FPGA (Field-Programmable Gate Array) designs to provide a simple and efficient way to communicate with other components in the system. The AXI Stream UART is a serial communication interface that enables the transfer of data between two devices in a streaming fashion, where data is sent continuously without any start or stop bits. This makes it ideal for applications that require a high throughput of data, such as video or audio streaming.

The AXI Stream UART interface is designed to work with the AXI Stream protocol, which is a high-speed, packet-based protocol used to transfer data between components in an FPGA. The AXI Stream protocol uses a simple, unidirectional interface that allows data to be transmitted in a continuous stream, without any handshaking signals.

One of the key benefits of using AXI Stream UART is that it simplifies the design process by eliminating the need for additional components, such as a serializer or deserializer. This can help reduce the overall complexity of the FPGA design and improve system performance. In summary, the AXI Stream UART is a high-speed, streaming interface that allows for efficient transfer of data between components in an FPGA design. It is commonly used in applications that require a high throughput of data, such as video or audio streaming. A block diagram for the UART IP is shown in Figure 1.

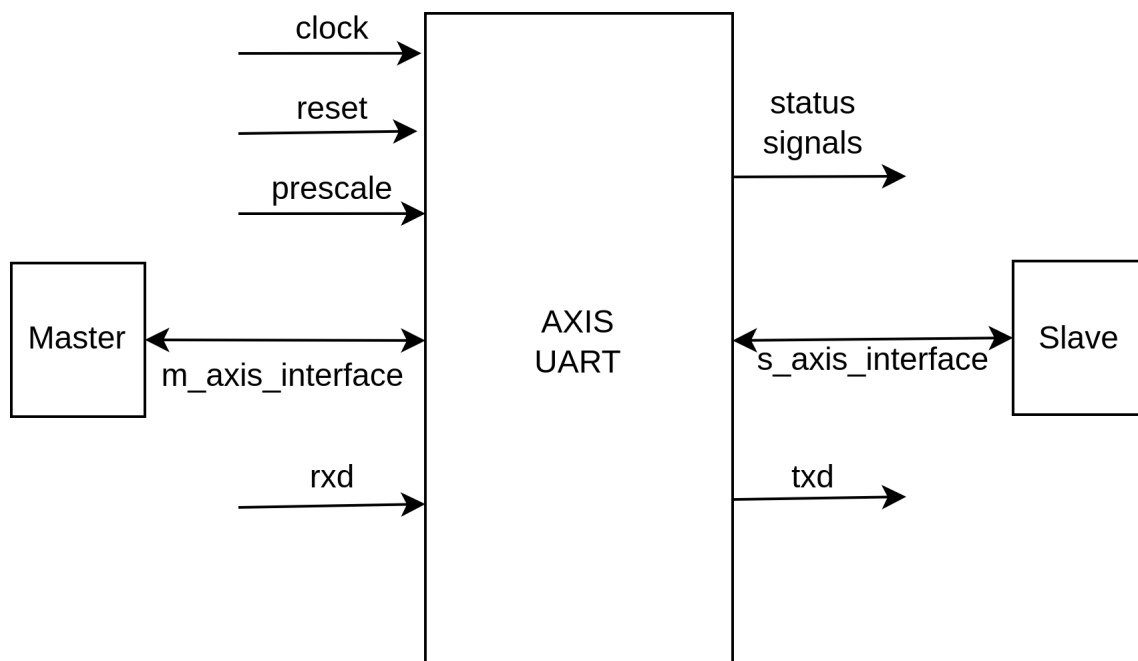


Figure 1: AXIS UART Block Diagram

# IP Specification

The UART transmitter and receiver both use a single transmit or receive pin. The modules take one parameter, DATA\_WIDTH, that specifies the width of both the data bus and the length of the actual data words communicated. The default value is 8 for an 8 bit interface. The prescale input determines the data rate - it should be set to  $F_{clk} / (\text{baud} * 8)$ . This is an input instead of a parameter so it can be changed at run time, though it is not buffered internally so care should be used to avoid corrupt data. The main interface to the user design is an AXI4-Stream interface that consists of the tdata, tvalid, and tready signals. tready flows in the opposite direction. tdata is considered valid when tvalid is high. The destination will accept data only when tready is high. Data is transferred from the source to the destination only when both tvalid and tready are high, otherwise the bus is stalled. Both interfaces also present a 'busy' signal that is high when an operation is taking place. The receiver also presents overrun error and frame error strobe outputs. If the data word currently in the tdata output register is not read before another word is received, then a single cycle pulse will be emitted from overrun\_error and the word is discarded. If the receiver does not get a stop bit of the right level, then a single pulse will be emitted from the frame\_error output and the received word will be discarded. The internal block diagram can be seen in Figure 2.

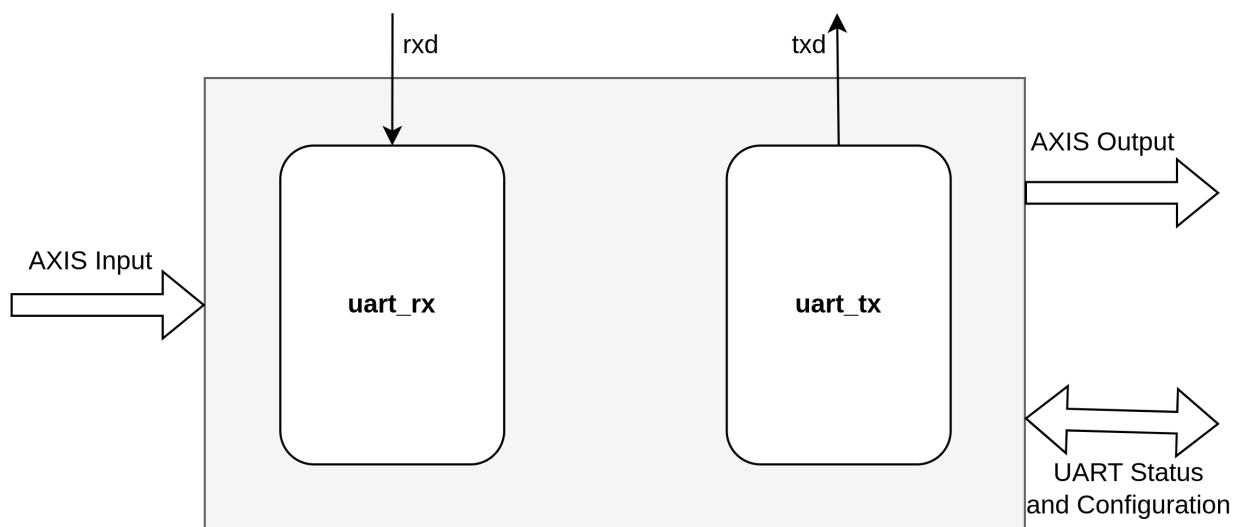


Figure 2: AXIS UART Internal Diagram

## Standards

The AXI4-Stream interface is compliant with the AMBA® AXI Protocol Specification.

## IP Support Details

The Table 1 gives the support details for AXIS UART.

| Compliance |             | IP Resources |                  |                  |                  | Tool Flow               |                 |           |
|------------|-------------|--------------|------------------|------------------|------------------|-------------------------|-----------------|-----------|
| Device     | Interface   | Source Files | Constraint Files | Testbench        | Simulation Model | Analyze and Elaboration | Simulation      | Synthesis |
| Gemini     | AXI4-Stream | Verilog      | -                | Python / Verilog | MyHDL            | Verific (Raptor)        | Icarus (Raptor) | Raptor    |

Table 1: IP Details

## Port List

Table 2 lists the top interface ports of the AXIS UART.

| Signal Name                 | I/O | Description                          |
|-----------------------------|-----|--------------------------------------|
| <b>AXI Clock and Reset</b>  |     |                                      |
| clk                         | I   | AXI4-Stream Clock                    |
| rst                         | I   | AXI4-Stream Active High Reset        |
| <b>AXI Slave Interface</b>  |     |                                      |
| s_axis_tdata                | I   | AXI4-Stream data                     |
| s_axis_tvalid               | I   | AXI4-Stream valid transfer           |
| s_axis_tready               | O   | AXI4-Stream transfer ready           |
| <b>AXI Master Interface</b> |     |                                      |
| m_axis_tdata                | O   | AXI4-Stream data                     |
| m_axis_tvalid               | O   | AXI4-Stream valid transfer           |
| m_axis_tready               | I   | AXI4-Stream transfer ready           |
| <b>UART Interface</b>       |     |                                      |
| rx_d                        | I   | UART reception                       |
| tx_d                        | O   | UART transmission                    |
| <b>Status Signals</b>       |     |                                      |
| tx_busy                     | O   | UART transmission busy               |
| rx_busy                     | O   | UART reception busy                  |
| rx_overrun_error            | O   | UART overrun of data reception       |
| rx_frame_error              | O   | UART reception of non-integral units |
| <b>Configuration</b>        |     |                                      |
| prescale                    | I   | UART prescaler configuration         |

Table 2: AXIS UART Interface

## Parameters

Table 3 lists the parameters of the AXIS UART.

| Parameter  | Values                             | Default Value | Description                  |
|------------|------------------------------------|---------------|------------------------------|
| DATA WIDTH | 8, 16, 32, 64, 128, 256, 512, 1024 | 8             | Data Width for each transfer |

Table 3: Parameters

## Resource Utilization

The parameters for computing the maximum and minimum resource utilization are given in Table 4, remaining parameters have been kept at their default values.

| Tool             | Raptor Design Suite |               |                   |          |
|------------------|---------------------|---------------|-------------------|----------|
| FPGA Device      | GEMINI              |               |                   |          |
| Configuration    |                     |               | Resource Utilized |          |
| Minimum Resource | Options             | Configuration | Resources         | Utilized |
|                  | DATA WIDTH          | 8             | LUTs              | 165      |
|                  |                     |               | Registers         | 79       |
| Maximum Resource | Options             | Configuration | Resources         | Utilized |
|                  | DATA WIDTH          | 1024          | LUTs              | 3193     |
|                  |                     |               | Registers         | 3121     |

Table 4: Resource Utilization



# Design Flow

## IP Customization and Generation

AXIS UART IP core is a part of the Raptor Design Suite Software. A customized AXIS UART can be generated from the Raptor's IP configurator window as shown in Figure 3.

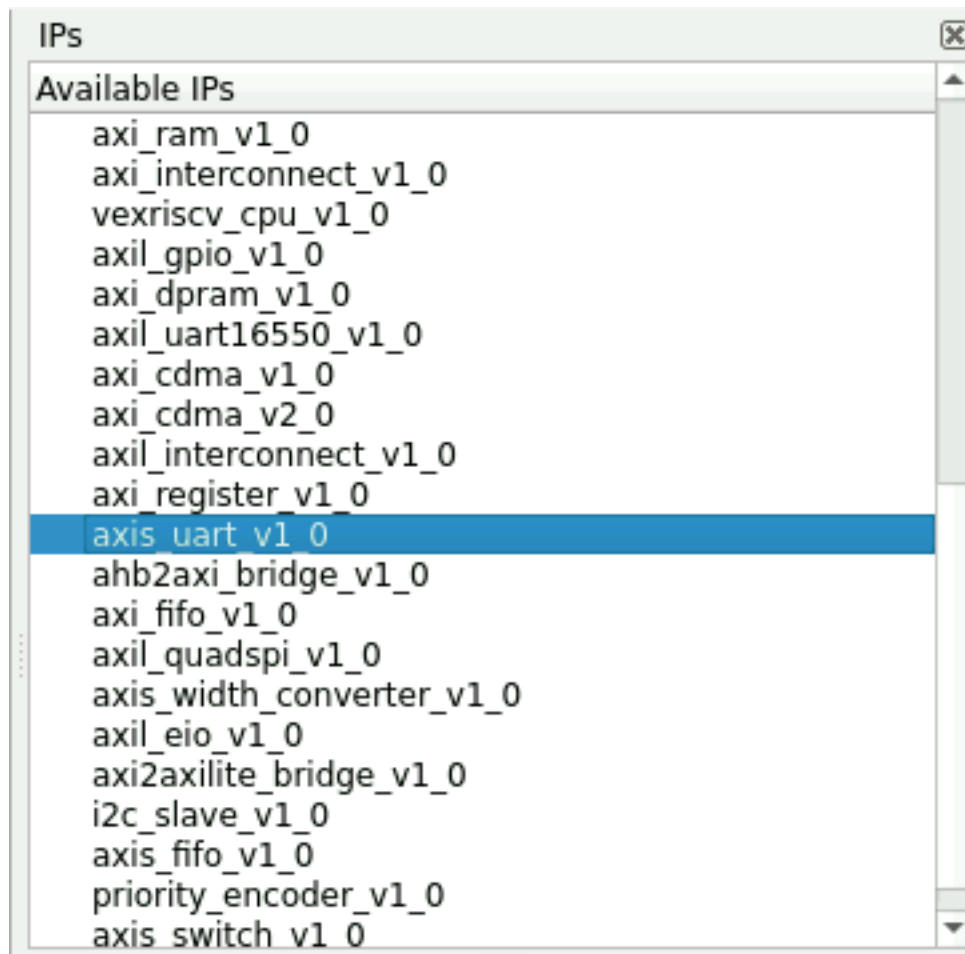


Figure 3: IP list

## Parameters Customization

From the IP configuration window, the parameters of the UART can be configured and UART features can be enabled for generating a customized UART IP core that suits the user application requirement as shown in Figure 4. After IP Customization, all the source files are made available to the user with a top wrapper that instantiates a parameterized instance of the AXIS UART.

The screenshot shows a 'Configure IP' dialog box with the following details:

- Title Bar:** Configure IP
- Text:** Configuring axis\_uart (v1.0) from rapidsilicon's ip library
- Parameters Section:**
  - DATA\_WIDTH:** A dropdown menu currently showing '8'.
- Output Section:**
  - Module Name:** A text field containing 'axis\_uart\_wrapper'.
  - Output Dir:** A text field containing '\_uart/v1\_0/axis\_uart\_wrapper'.
- Buttons:** A 'Generate IP' button is located at the bottom right of the dialog.

Figure 4: IP Configuration

# Example Design

## Overview

This AXIL UART IP can be utilized in a system that requires sequential transmission and reception of data from the outside world. UART is a crucial component in many electronic systems, enabling communication between the system and external devices through a serial interface. It can be embedded inside SoCs to enable two-way communication via the SoC in a streaming interface to maximize the throughput. One such example design of this AXIS UART can be visualized in Figure 5.

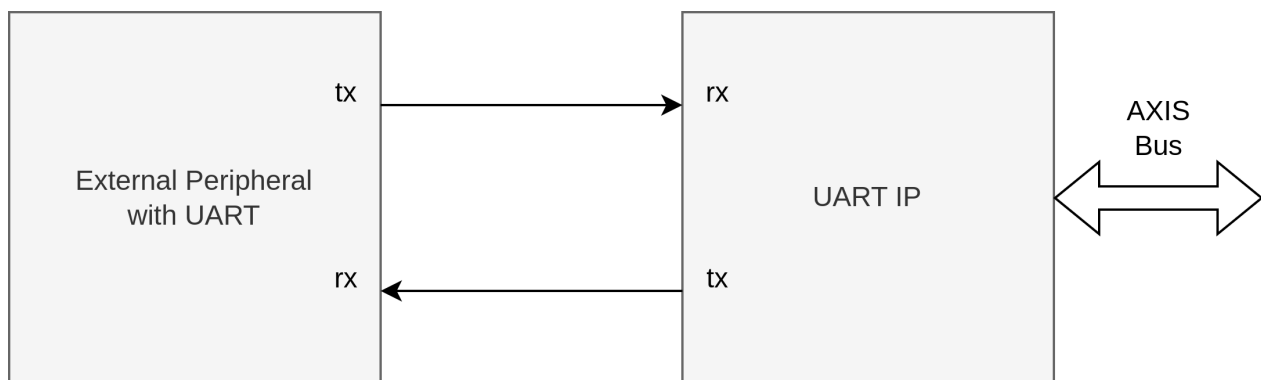


Figure 5: AXIS UART replicating connected with a Peripheral

## Simulating the Example Design

The IP being Verilog HDL, can be simulated via a bunch of industry standard stimulus. For instance, it could be simulated via writing a Verilog Test-bench, or incorporating a soft processor that can stimulate this UART IP. The bundled example design is stimulated via a MyHDL based environment that iteratively stimulates the soft IP with the help of a Verilog testbench by performing various transmission and reception operations on the UART interface while also stimulating the AXI-Stream interface.

## Synthesis and PR

Raptor Suite is armed with tools for Synthesis along with Post and Route capabilities and the generated postsynthesis and post-route and place netlists can be viewed and analyzed from within the Raptor. The generated bitstream can then be uploaded on an FPGA device to be utilized in hardware applications.

# Test Bench

The included testbench for the AXIS UART IP is a MyHDL based Python testbench that performs various transmission and reception operations on the IP core with the help of pre-defined Verilog testbenches. Python is used to simulate the Verilog testbenches under the influence of MyHDL for this purpose and a total of 2 tests are performed, one for the transmission and the other for reception of the data by the UART IP from the Verilog testbenches. The waveforms are also dumped in the format of .lxt for in-depth analysis of the whole operation. Being written in simple Verilog and Python, the testbenches are easily modifiable to provide maximum coverage of the UART IP.

# Release

## Release History

| Date        | Version | Revisions  |
|-------------|---------|--|
| May 9, 2023 | 0.1     | Initial version AXI4-Stream UART User Guide Document |