# IOB-UART, a RISC-V UART

## User Guide, V0.1 , Build e10e536


February 14, 2022

  

# Contents



# List of Tables



# List of Figures



 

  

# 1 Introduction

The IObundle UART is a RISC-V-based Peripheral written in Verilog, which users can download for free, modify, simulate and implement in FPGA or ASIC. It is written in Verilog and includes a C software driver. The IObundle UART is a very compact IP that works at high clock rates if needed. It supports full-duplex operation and a configurable baud rate. The IObundle UART has a fixed configuration for the Start and Stop bits. More flexible licensable commercial versions are available upon request.

# 2 Symbol

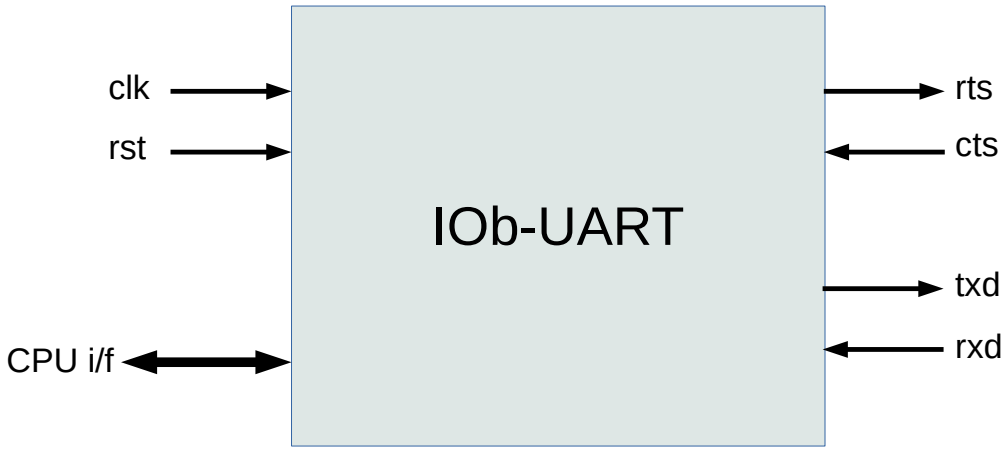


Figure 1: IP core symbol.

# 3 Features

- Supported in IObundle's RISC-V IOb-SoC open-source and free of charge template.
- IObundle's IOb-SoC native CPU interface.
- Verilog basic UART implementation.
- Soft reset and enable functions.
- Runtime configurable baud rate
- C software driver at the bare-metal level.
- Simple Verilog testbench for the IP's *nucleus*.
- System-level Verilog testbench available when simulating the IP embedded in IOb-SoC.
- Simulation Makefile for the open-source and free of charge Icarus Verilog simulator.
- FPGA synthesis and implementation scripts for two FPGA families from two FPGA vendors.
- Automated creation of FPGA netlists
- Automated production of documentation using the open-source and free Latex framework.

- IP data automatically extracted from FPGA tool logs to include in documents.
- Makefile tree for full automation of simulation, FPGA implementation and document production.
- AXI4 Lite CPU interface (premium option).
- Parity bits (premium option).

# 4 Benefits

- Compact and easy to integrate hardware and software implementation
- Can fit many instances in low cost FPGAs and ASICs
- Low power consumption

# 5 Deliverables

- ASIC or FPGA synthesized netlist or Verilog source code, and respective synthesis and implementation scripts
- ASIC or FPGA verification environment by simulation and emulation
- Bare-metal software driver and example user software
- User documentation for easy system integration
- Example integration in IOb-SoC (optional)

# 6 Block Diagram and Description

A high-level block diagram of the core is presented in Figure 2, followed by a brief description of each of the blocks.

  

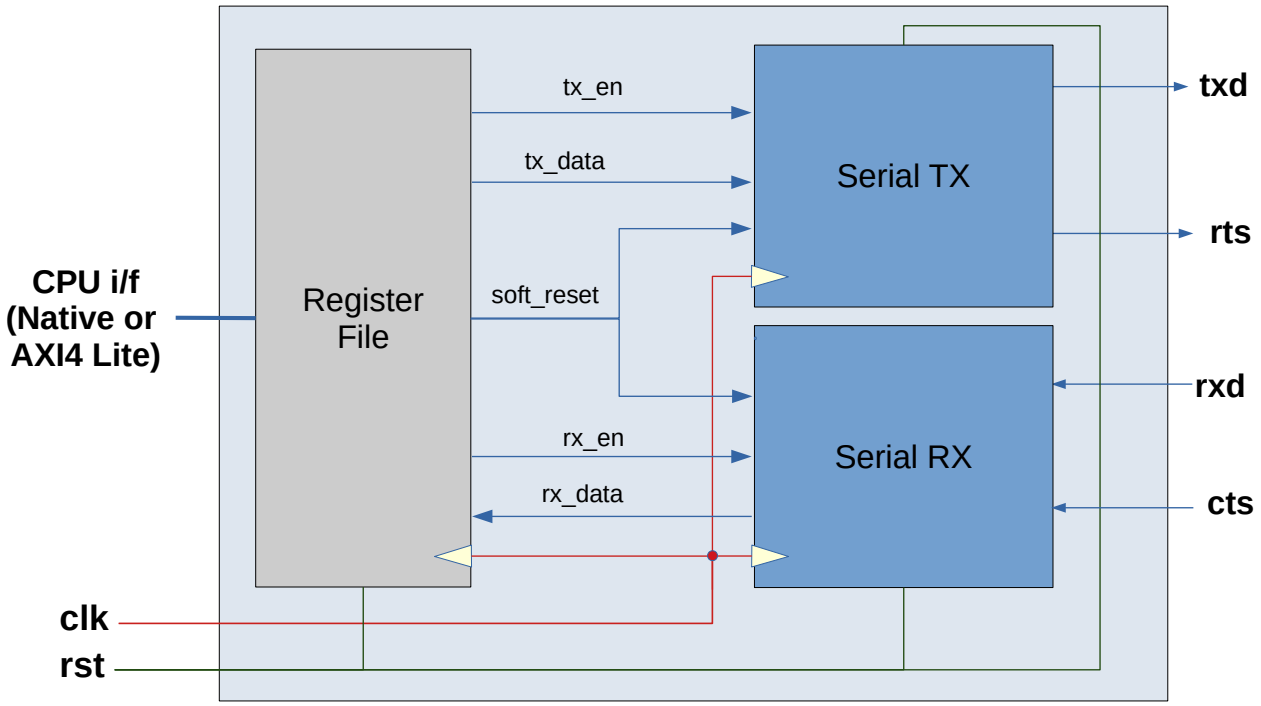


Figure 2: High-level block diagram.

**SERIAL TRANSMIT CONTROLLER :** This block serializes the data written to the UART_TXDATA by the CPU, and sends it to the `txd` ouput.

**SERIAL REICEIVE CONTROLLER :** This block deserializes the data received from the pin `rxd` input, and writes it to the UART_RXDATA register, so the CPU can read it.

# 7 Interface Signals

The interface signals of the core are described in the following tables.

| Name | Direction | Width | Description |
|---|---|---|---|
| | | | |

Table 1: General interface signals.

| Name | Direction | Width | Description |
|---|---|---|---|
| interrupt | OUTPUT | 1 | to be done |
| txd | OUTPUT | 1 | Serial transmit line |
| rxd | INPUT | 1 | Serial receive line |
| cts | INPUT | 1 | Clear to send; the destination is ready to receive a transmission sent by the UART |
| rts | OUTPUT | 1 | Ready to send; the UART is ready to receive a transmission from the sender. |

Table 2: RS232 Interface Signals

| Name | Direction | Width | Description |
|---|---|---|---|
| valid | INPUT | 1 | Native CPU interface valid signal |
| address | INPUT | ADDR_W | Native CPU interface address signal |
| wdata | INPUT | DATA_W | Native CPU interface data write signal |
| wstrb | INPUT | DATA_W/8 | Native CPU interface write strobe signal |
| rdata | OUTPUT | DATA_W | Native CPU interface read data signal |
| ready | OUTPUT | 1 | Native CPU interface ready signal |

Table 3: CPU Native Slave Interface Signals

# 8 Software Accessible Registers

The software accessible registers of the core are described in the following tables. The tables give information on the name, read/write capability, word aligned addresses, used word bits, and a textual description.

| Name | R/W | Addr | Bits | Initial Value | Description |
|---|---|---|---|---|---|
| UART_SOFTRESET | W | 0x00 | 0:0 | 0 | Bit duration in system clock cycles. |
| UART_DIV | W | 0x04 | 15:0 | 0 | Bit duration in system clock cycles. |
| UART_TXDATA | W | 0x08 | 7:0 | 0 | TX data |
| UART_TXEN | W | 0x0c | 0:0 | 0 | TX enable. |
| UART_TXREADY | R | 0x10 | 0:0 | 0 | TX ready to receive data |
| UART_RXDATA | R | 0x14 | 7:0 | 0 | RX data |
| UART_RXEN | W | 0x18 | 0:0 | 0 | RX enable. |
| UART_RXREADY | R | 0x1c | 0:0 | 0 | RX data is ready to be read. |

Table 4: UART software accessible registers.

# 9 Implementation Results

| Resource | Used |
|---|---|
| LUTs | 100 |
| Registers | 112 |
| DSPs | 0 |
| BRAM | 0 |

| Resource | Used |
|---|---|
| ALM | 87 |
| FF | 121 |
| DSP | 0 |
| BRAM blocks | 0 |
| BRAM bits | 0 |

Table 5: FPGA results for Kintex Ultrascale (left) and Cyclone V GT (right).