IOB-UART, a RISC-V UART

User Guide, V0.1 , Build 397ec7b

IObundle

January 25, 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

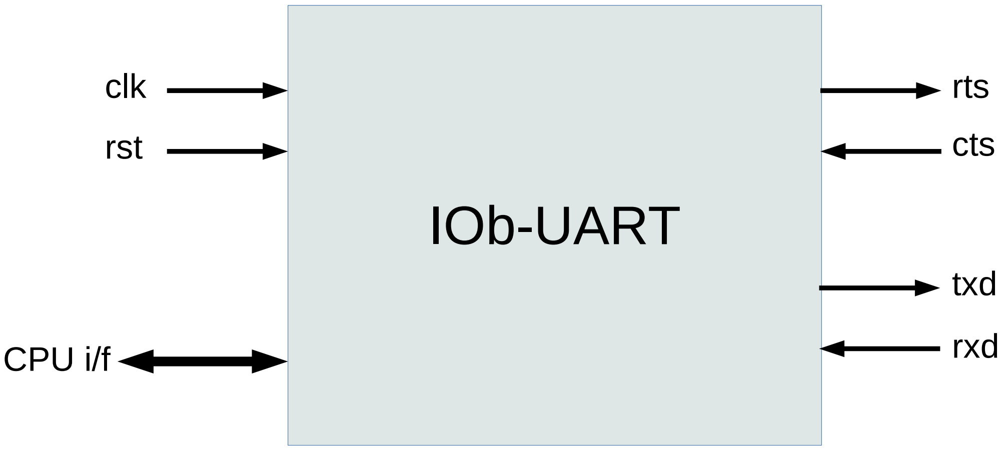

Figure 1: IP core symbol.

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 and a brief explanation of each block is given in Table 1.

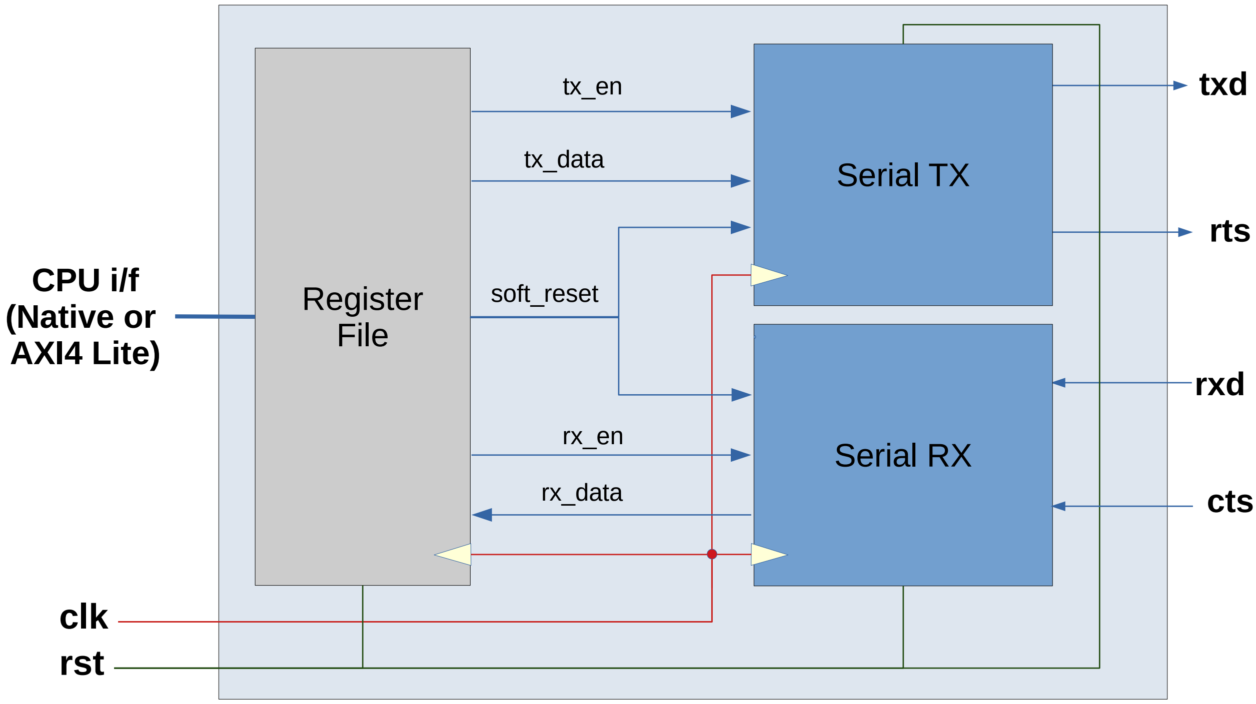

Figure 2: High-level block diagram.

Block	Description
Serial TX	After enabled, this block serializes the data previously written to the `tx_data` register by the CPU, and sends the data word over the single transmit line connected to output `txd`.
Serial RX	After enabled, this block deserializes the data in the incoming single transmit line connected to pin `txd`, and writes a data word to the `rx_data` register for the CPU to read.

Table 1: Block descriptions.

7 Interface Signals

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

Name	Direction	Width	Description
clk	input	1	System clock input
rst	input	1	System reset asynchronous and active high

Table 2: 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 3: 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	WDATA_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 4: CPU Native Slave Interface Signals

Name	Direction	Width	Description
s_axil_awid	input	'AXI_ID_W	Address write channel ID
s_axil_awaddr	input	AXIL_ADDR_W	Address write channel address
s_axil_awprot	input	'AXI_PROT_W	Address write channel protection type. Transactions set with Normal Secure and Data attributes (000).
s_axil_awqos	input	'AXI_QOS_W	Address write channel quality of service
s_axil_awvalid	input	1	Address write channel valid
s_axil_awready	output	1	Address write channel ready
s_axil_wid	input	'AXI_ID_W	Write channel ID
s_axil_wdata	input	AXIL_DATA_W	Write channel data
s_axil_wstrb	input	AXIL_DATA_W/8	Write channel write strobe
s_axil_wvalid	input	1	Write channel valid
s_axil_wready	output	1	Write channel ready
s_axil_bid	output	'AXI_ID_W	Write response channel ID
s_axil_bresp	output	'AXI_RESP_W	Write response channel response
s_axil_bvalid	output	1	Write response channel valid
s_axil_bready	input	1	Write response channel ready
s_axil_arid	input	'AXI_ID_W	Address read channel ID
s_axil_araddr	input	AXIL_ADDR_W	Address read channel address
s_axil_arprot	input	'AXI_PROT_W	Address read channel protection type. Transactions set with Normal Secure and Data attributes (000).
s_axil_arqos	input	'AXI_QOS_W	Address read channel quality of service
s_axil_arvalid	input	1	Address read channel valid
s_axil_arready	output	1	Address read channel ready
s_axil_rid	output	'AXI_ID_W	Read channel ID
s_axil_rdata	output	AXIL_DATA_W	Read channel data
s_axil_rresp	output	'AXI_RESP_W	Read channel response
s_axil_rvalid	output	1	Read channel valid
s_axil_rready	input	1	Read channel ready

Table 5: CPU AXI4 Lite 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 6: UART software accessible registers.

9 Implementation Results

Resource	Used
LUTs	100
Registers	112
DSPs	0
BRAM	0

Resource	Used
ALM	88
FF	124
DSP	0
BRAM blocks	0
BRAM bits	0

Table 7: FPGA results for Kintex Ultrascale (left) and Cyclone V GT (right).

