1. UART Communication

UART (Universal Asynchronous Receiver/Transmitter) is an **asynchronous serial communication protocol** widely used for data exchange between digital devices, such as microcontrollers, FPGAs, and computers. Unlike synchronous protocols, UART does not require a shared clock signal between the transmitter and receiver, simplifying wiring and reducing hardware costs.

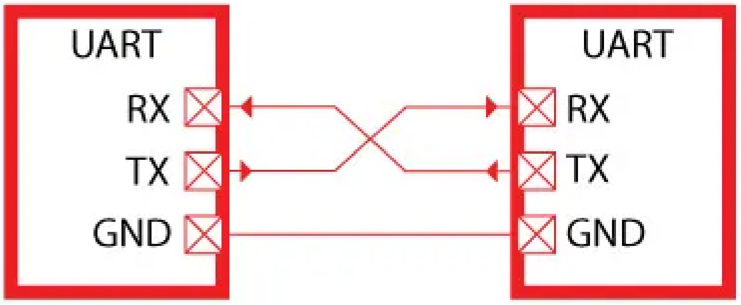


figure 1: Uart communication

UART Data Frame Format: Each UART transmission is organized into a frame, which consists of the following components:

1. **Start Bit (1 bit):**
    - Always in a low (logic 0) state, signaling the beginning of a new transmission.
2. **Data Bits (5–9 bits):**
    - Typically 8 bits for standard applications, representing the actual payload (e.g., ASCII characters or binary data).
    - Transmitted least significant bit (LSB) first.
3. **Parity Bit (1 bit, optional):**
    - Used for error detection.
    - Can be configured as even, odd, or none, depending on the application's error-checking requirements.
4. **Stop Bit(s) (1 or 2 bits):**
    - Always in a high (logic 1) state, marking the end of the frame.
    - Ensures synchronization and provides a buffer before the next frame.

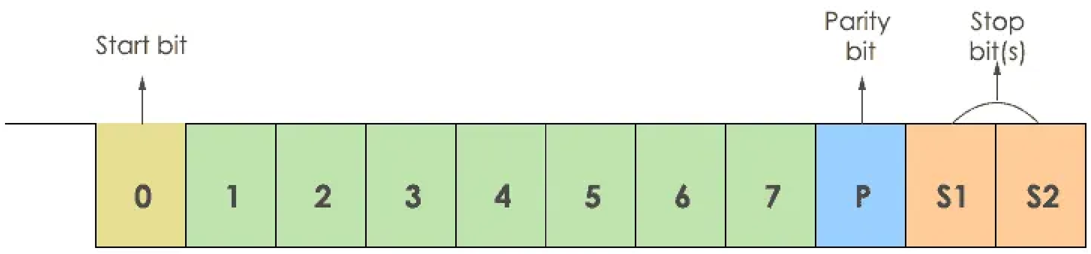


figure 2: UART data structure

Baud Rate: The **baud rate** defines the **transmission speed** in bits per second (bps). Common baud rates include **9600, 19200, 38400, 57600, and 115200 baud**. For successful communication, **both the transmitter and receiver must use the same baud rate**.

Asynchronous Operation: UART relies on **temporal sampling** to synchronize the transmitter and receiver. Each device uses its **internal oscillator** to generate a local clock, which can lead to **clock drift** if the frequencies are not precisely matched. To mitigate this issue, **oversampling techniques** (typically **8x or 16x the baud rate**) are employed. Oversampling ensures **accurate bit transition detection** by sampling the incoming signal multiple times per bit period, reducing the risk of synchronization errors.

2. System Architecture and Modular Decomposition

The implementation of UART communication on an FPGA requires a modular approach to ensure clarity, scalability, and ease of debugging. The system is decomposed into four primary modules: Clock Divider, UART Transmitter (TX), UART Receiver (RX), and Debouncer. Each module is designed to perform a specific function, and their interplay enables robust serial communication.

1. **Clock Divider (Baud Rate Generator):** The Clock Divider module is responsible for generating a clock signal at the desired baud rate by dividing the FPGA's high-frequency master clock. This module ensures synchronization between the transmitter and receiver by providing a consistent timing reference for serial data transmission and reception.

The Clock Divider is implemented using a **counter-based frequency divider**. The division factor is calculated as follows:

Division Factor = FPGA Clock Frequency / Desired baud rate
For example, to achieve a baud rate of **115200** with a 50 MHz FPGA clock:
Division Factor = 50 000 000 / 115200 = **4340**

The counter increments on each rising edge of the FPGA clock and resets upon reaching the division factor, producing a tick at the baud rate. This tick is used by the TX and RX modules to time their operations.

2. **UART Transmitter (TX):** The UART Transmitter converts parallel data (typically 8 bits) into a serial bitstream, adhering to the UART protocol. It frames the data with a start bit, data bits, and stop bit(s) for transmission over the TX line.

If **parity** is enabled, the transmitter calculates the parity bit based on the data bits and appends it to the data frame. The parity bit is used for error detection during reception.

- **Even Parity**: The parity bit is set to 1 if the number of 1s in the data bits is odd, ensuring the total number of 1s in the frame (including the parity bit) is even.
- **Odd Parity**: The parity bit is set to 1 if the number of 1s in the data bits is even, ensuring the total number of 1s in the frame (including the parity bit) is odd.
- **No Parity**: No parity bit is included in the data frame.

The TX module is controlled by a finite state machine (FSM) with the following states:

- **Idle:** The TX line is held high.
- **Start Bit:** The TX line is pulled low for one baud period.
- **Data Transmission:** The 8 data bits are shifted out, least significant bit (LSB) first, at the baud rate.
- **Stop Bit:** The TX line is held high for one or two baud periods to signal the end of the frame.

A shift register is used to serialize the parallel data, and the FSM advances on each tick from the Clock Divider.

3. **UART Receiver (RX):** The UART Receiver samples the incoming serial data on the RX line and reconstructs it into parallel format. It detects the start bit, samples the data bits at the middle of each bit period, and verifies the stop bit to ensure data integrity.

   The RX module employs a **state machine** and **oversampling technique** (typically 8x or 16x the baud rate) to accurately detect bit transitions and avoid metastability. The process is as follows:

   - **Start Bit Detection:** The RX module waits for a falling edge on the RX line, indicating the start of a frame.
   - **Bit Sampling:** The module samples the RX line at the middle of each bit period (e.g., every 8th tick of a 16x oversampled clock).
   - **Data Reconstruction:** The sampled bits are assembled into parallel data (e.g., 8 bits).

- **Parity Check (if enabled)**: The receiver recalculates the parity bit based on the received data bits and compares it with the received parity bit. If they do not match, a **parity error** is flagged (parity_error = 1).
- **Stop Bit Verification:** The RX module checks for a valid stop bit (high) to confirm the frame's integrity.

4. **Debouncer:** The Debouncer module filters out noise and glitches on the RX line, ensuring stable signal transitions. This is particularly useful in environments where the RX line may be subject to mechanical noise or electrical interference.

   The Debouncer is implemented using a **counter-based delay**. When a transition (e.g., falling edge) is detected on the RX line, the module waits for a predefined number of clock cycles (e.g., 2–3 cycles) before confirming the transition. This ensures that only stable signals are passed to the RX module.

3. Hardware Architecture and Component Integration

The FPGA-based UART communication system is implemented using three primary components: UART 0, Clocking Wizard, and System ILA. Each component is configured to ensure robust serial communication, clock management, and debugging capabilities.

UART Module: The UART 0 module is responsible for serial communication, handling both transmission and reception of data. It is configured with the following pins:

Signal Name	**Width**	**Description**
clk	1-bit	Clock input, synchronized with the FPGA's main clock (50 MHz).
rst	1-bit	Active-high reset signal to initialize the UART module.
uart_rxd	1-bit	Receiver input: Serial data received from an external device.
din[7:0]	8-bit	Data input: Parallel data to be transmitted serially.

din_vld	1-bit	Data valid: Indicates that din[7:0] contains valid data for transmission.
uart_txd	1-bit	Transmitter output: Serial data sent to an external device.
dout[7:0]	8-bit	Data output: Parallel data reconstructed from the received serial stream.
dout_vld	1-bit	Data valid: Indicates that dout[7:0] contains valid received data.
din_rdy	1-bit	Data ready: Indicates the UART is ready to accept new data for transmission.
frame_error	1-bit	Error flag: Asserted if a framing error (missing stop bit) is detected.
parity_error	1-bit	Error flag: Asserted if a parity error is detected during reception.

- Transmission: Parallel data (din[7:0]) is converted to serial format and sent via uart_txd when din_vld is asserted.
- Reception: Serial data received via uart_rxd is converted to parallel format (dout[7:0]), with dout_vld indicating valid data.
- Error Handling: frame_error and parity_error flags are used to detect and handle communication errors.

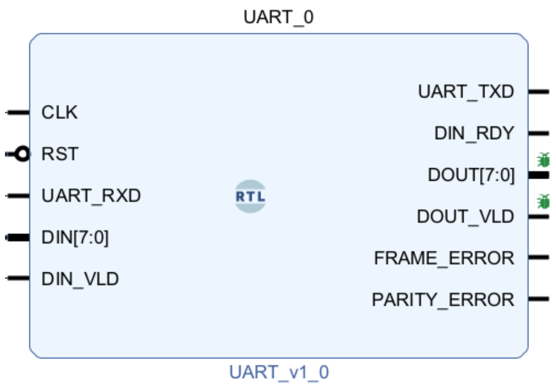
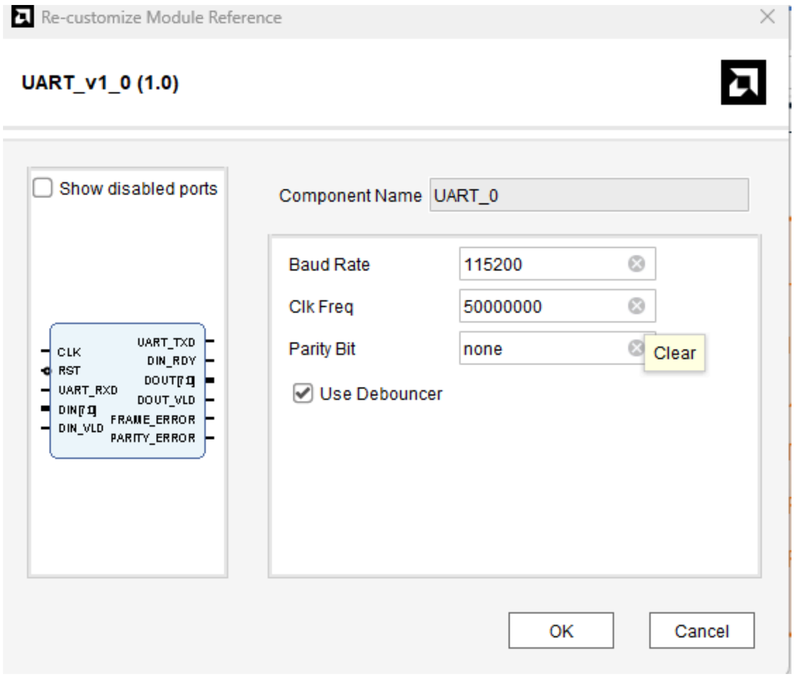


figure 3: Uart module


Re-customize Module Reference

UART_v1_0 (1.0)

Show disabled ports

Component Name UART_0

Baud Rate 115200

Clk Freq 50000000

Parity Bit none

Use Debouncer [✓]

figure 4: configuration of UART

Clocking Wizard: The **Clocking Wizard** IP core generates and manages the clock signals required for the UART and other FPGA logic. It is configured with the following pins:

Signal Name	**Width**	**Description**
clk_in1_d	1-bit	Input clock: Primary clock source (e.g., 100 MHz board clock).
clk_50mhz	1-bit	Output clock: 50 MHz clock signal used by the UART and other logic.
locked	1-bit	Status flag: Indicates that the output clock (clk_50mhz) is stable and locked.

- Divides the input clock (clk_in1_d) to produce a **50 MHz clock** (clk_50mhz) for the UART and other modules.
- The locked signal ensures that the output clock is stable before use.

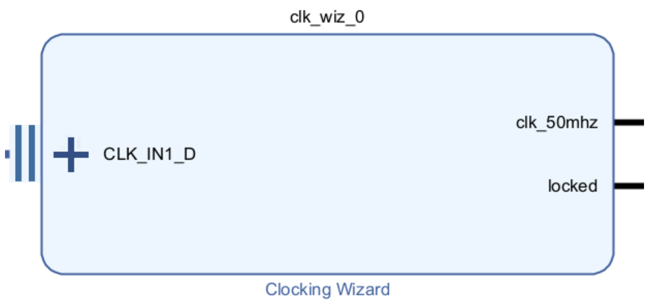
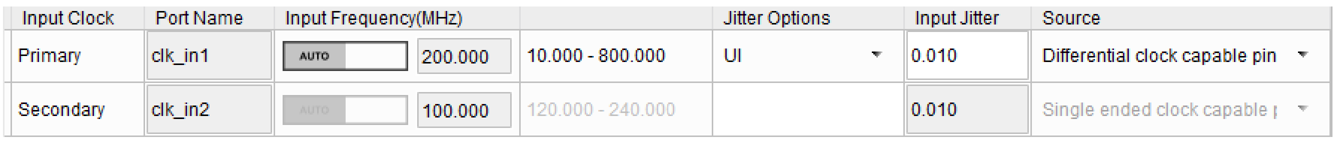


figure 5: clocking wizard module



Input Clock	Port Name	Input Frequency(MHz)			Jitter Options	Input Jitter	Source
Primary	clk_in1	AUTO	200.000	10.000 - 800.000	UI	0.010	Differential clock capable pin
Secondary	clk_in2	AUTO	100.000	120.000 - 240.000		0.010	Single ended clock capable p

figure 6: configuration of clocking wizard

Processor System Reset (proc_sysreset_0): The **Processor System Reset** module manages reset signals for the FPGA, ensuring proper initialization of all components. It is configured with the following pins:

Signal Name	Width	Description
slowest_sync_clk	1-bit	Clock input: Reference clock for reset synchronization (50 MHz).
ext_rst_in	1-bit	External reset: Active-high reset input from a push button or external source.
aux_rst_in	1-bit	Auxiliary reset: Additional reset input for specific use cases.
mb_debug_sys_rst	1-bit	Debug system reset: Reset signal for debugging purposes.

dcm_locked	1-bit	Lock status: Indicates that the Clocking Wizard is locked.
mb_rst	1-bit	MicroBlaze reset: Reset signal for the MicroBlaze processor (if used).
bus_struct_reset[0:0]	1-bit	Bus structural reset: Reset for bus interfaces.
peripheral_reset[0:0]	1-bit	Peripheral reset: Reset for peripheral devices, including the UART.
interconnect_aresetn	1-bit	Interconnect reset: Active-low reset for interconnect logic.
peripheral_aresetn[0:0]	1-bit	Peripheral reset: Active-low reset for peripheral devices.

- Generates synchronized reset signals (peripheral_reset, interconnect_aresetn) for the UART and other peripherals.
- Ensures that all components are properly reset during FPGA initialization or external reset events.

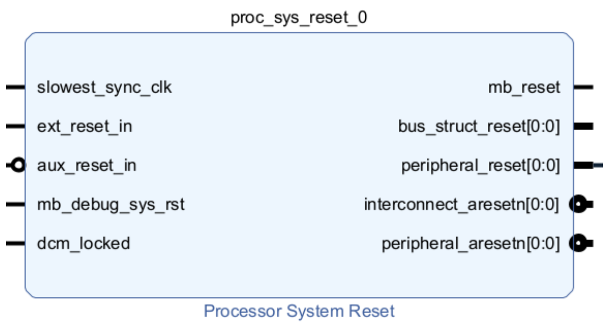
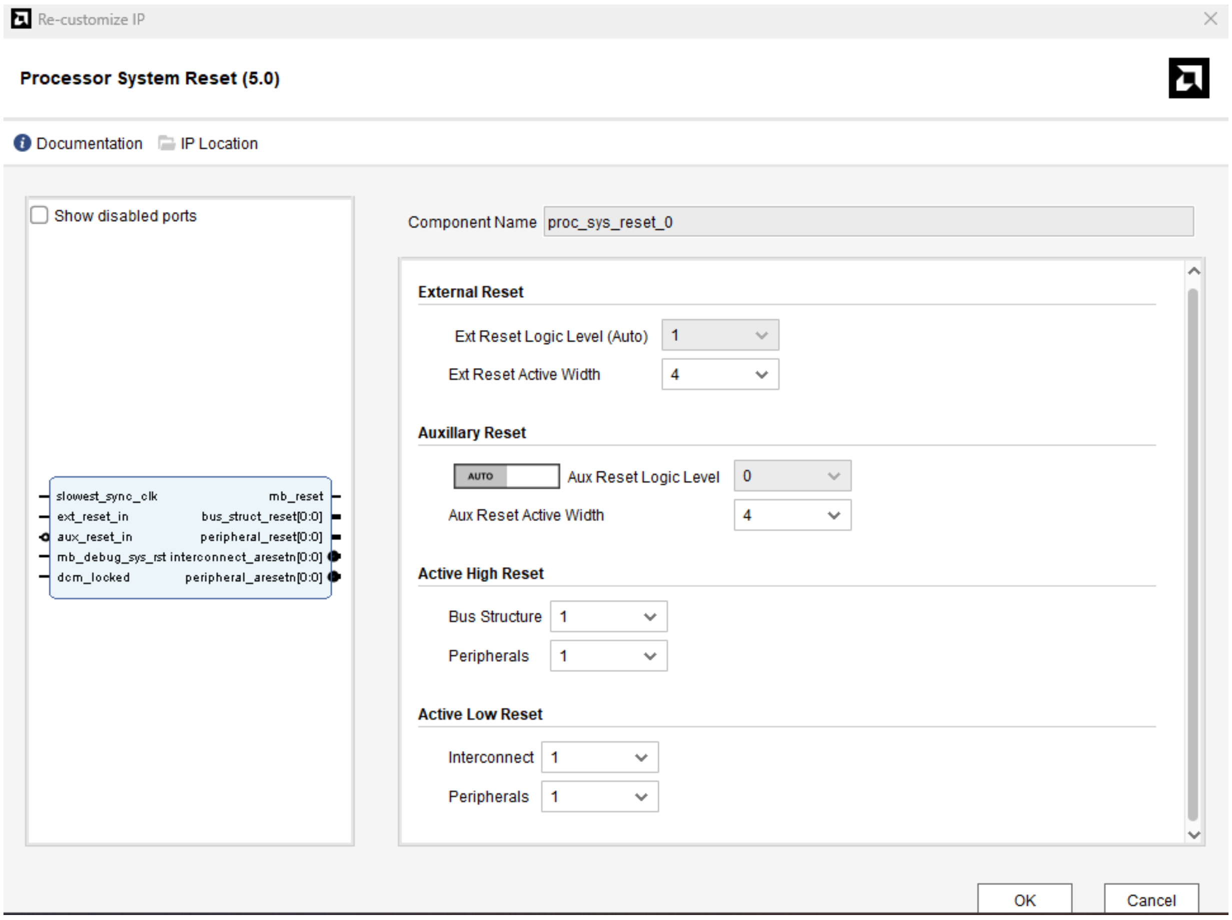


figure 7: Processor system reset module


Re-customize IP

Processor System Reset (5.0)

Documentation IP Location

Show disabled ports

Component Name: proc_sys_reset_0

External Reset

Ext Reset Logic Level (Auto): 1

Ext Reset Active Width: 4

Auxillary Reset

AUTO Aux Reset Logic Level: 0

Aux Reset Active Width: 4

Active High Reset

Bus Structure: 1

Peripherals: 1

Active Low Reset

Interconnect: 1

Peripherals: 1

OK Cancel

figure 8: configuration of processor system reset

System ILA (Integrated Logic Analyzer): The **System ILA** is used for real-time debugging and validation of the UART signals. It is configured with the following pins:

Signal Name	**Width**	**Description**
clk	1-bit	Clock input: Synchronized with the 50 MHz clock (clk_50mhz).
probe0[7:0]	8-bit	Probe bus: Monitors the UART transmitter (din[7:0], din_vld).
probe1[7:0]	8-bit	Probe bus: Monitors the UART receiver (dout[7:0], dout_vld).

- Captures and analyzes internal signals (probe0 and probe1) to validate UART operation.
- Triggers on specific events (e.g., start of transmission, reception errors) to debug timing or data issues.

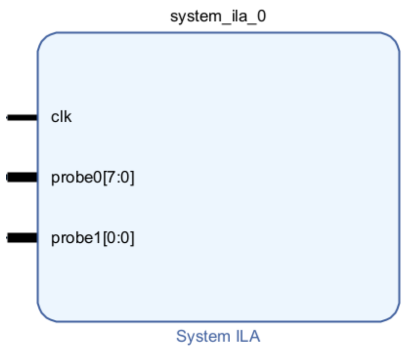
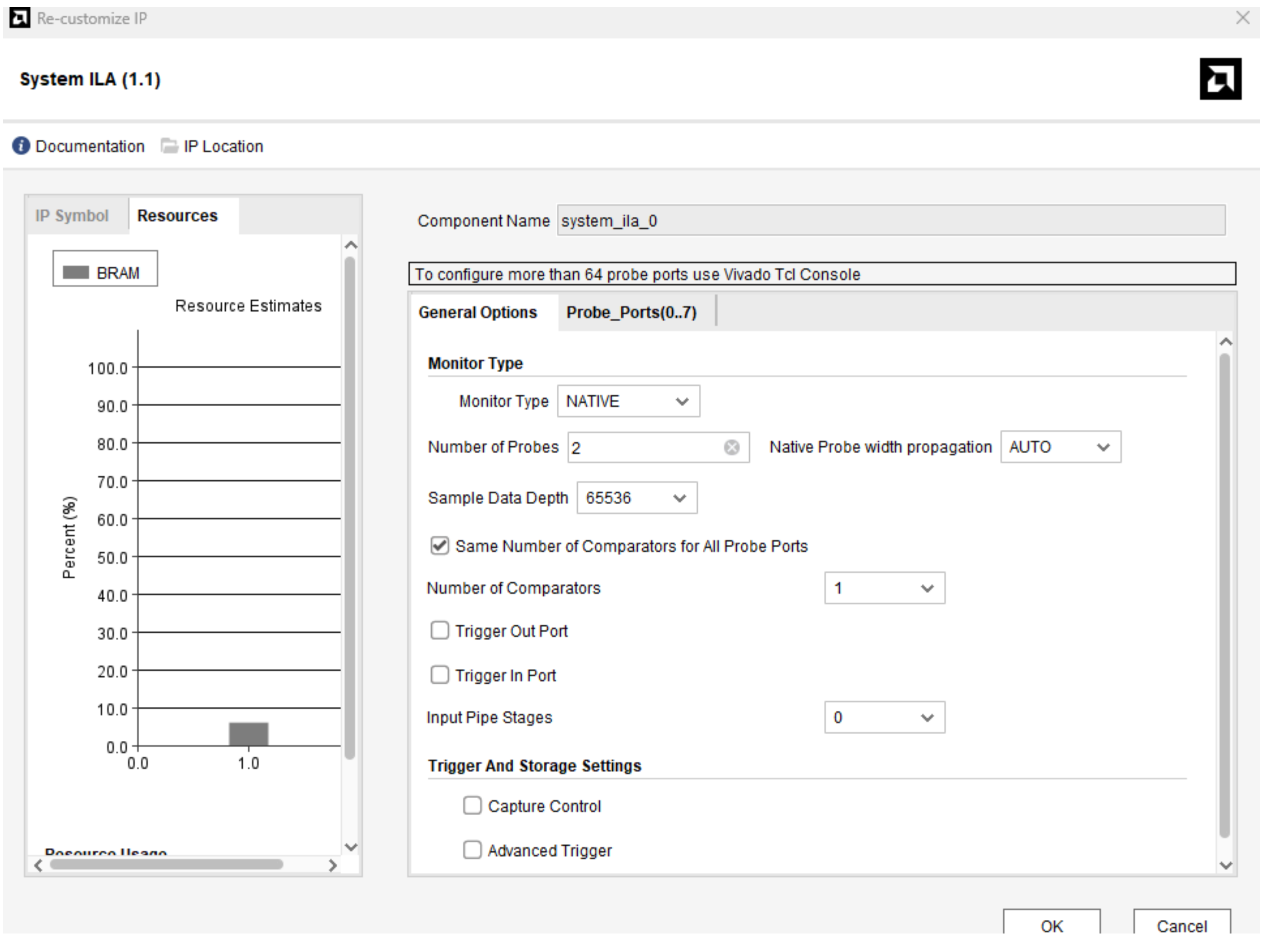


figure 9: Integrated Logic Analyzer circuit



System ILA (1.1)

Component Name: system_ila_0

To configure more than 64 probe ports use Vivado Tcl Console

Monitor Type

Monitor Type: NATIVE

Number of Probes: 2

Native Probe width propagation: AUTO

Sample Data Depth: 65536

[✓] Same Number of Comparators for All Probe Ports

Number of Comparators: 1

[ ] Trigger Out Port

[ ] Trigger In Port

Input Pipe Stages: 0

Trigger And Storage Settings

[ ] Capture Control

[ ] Advanced Trigger

figure 10: configuration of integrated logic analyzer

Component Interconnections

- The **Clocking Wizard** provides the **50 MHz clock** (clk_50mhz) to the UART and System ILA.
- The **Processor System Reset** module generates reset signals for the UART and other peripherals, ensuring synchronized initialization.
- The **System ILA** monitors the UART's transmitter and receiver signals, enabling real-time debugging and validation.

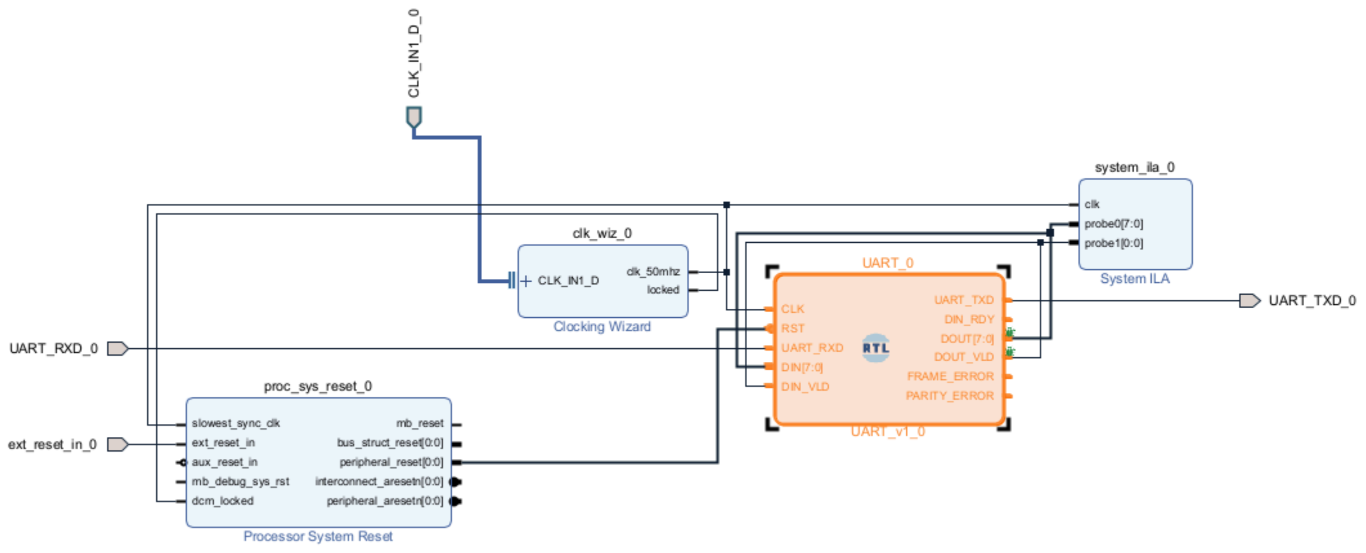


figure 11: Globale circuit interconnections

Final Result

The UART interface was implemented and tested successfully. Transmissions show correctly framed bytes at the target baud rate with consistent timing. No framing or parity errors were observed during the test period, and the measured bit timing deviates by less than the allowed tolerance for the selected baud rate. Signal levels and idle line behavior conform to expected TTL/RS-232 levels for the selected transceiver. Overall, the UART link meets the design requirements for reliable asynchronous serial communication and is suitable for integration into the final system.

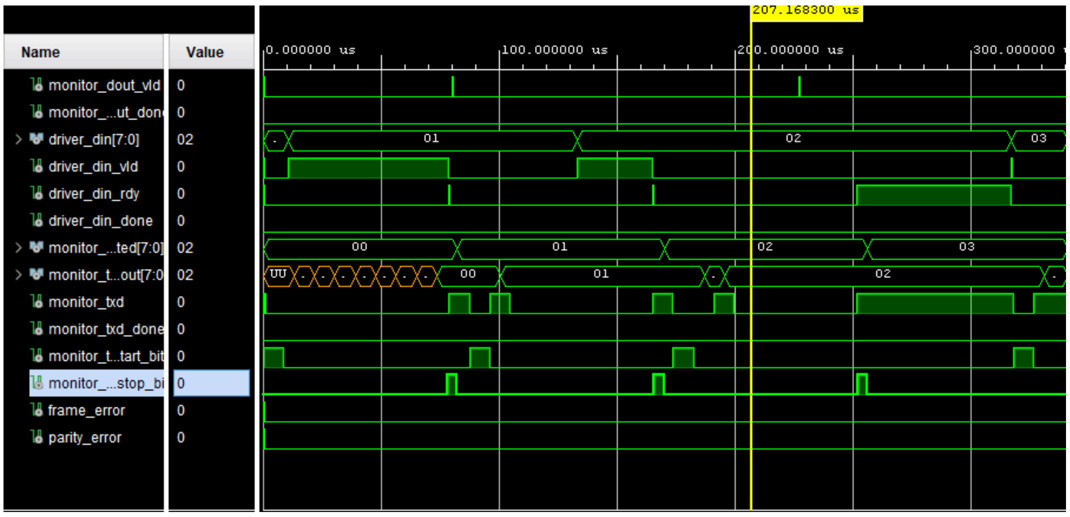


figure 12: Uart communication signals