

UART Protocol

- [Understanding the UART Protocol](#)
- [Creating Your Project in ChipInventor](#)
- [Blocks Used in the Project](#)
- [Connecting the Blocks](#)
- [Project Simulation](#)
- [FPGA Synthesis and Programming](#)
- [Hardware Validation](#)
- [Wrapping Up](#)

Understanding the UART Protocol

UART (Universal Asynchronous Receiver-Transmitter) is a widely used asynchronous serial communication protocol for exchanging data between devices, such as FPGAs and computers.

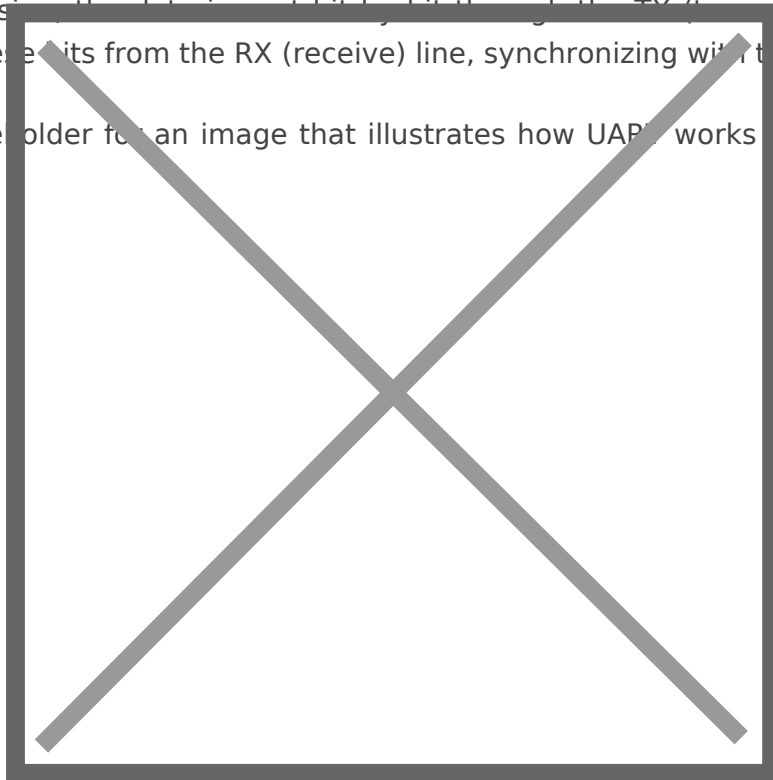
Unlike synchronous protocols, UART does not use a shared clock signal. Instead, data is sent at a predefined transmission rate (baud rate) agreed upon by both devices. Each transmitted byte includes:

- **1 start bit**
- **8 data bits**
- **1 stop bit**

During transmission, the transmitting device sends data through the TX (transmit) line, while the receiving device reads these bits from the RX (receive) line, synchronizing with the established timing.

Below is a placeholder for an image that illustrates how UART works (start bit, data bits, stop bit,

idle state, etc.).



Creating Your Project in ChipInventor

In this tutorial, you will create a serial communication system using UART with three main blocks: `uart_rx`, `uart_logic_const`, and `uart_tx`. When a character is received via UART, the system compares it with a predefined value and toggles an LED if it matches.

Step-by-Step:

1. Open **ChipInventor**.
2. Click on "**New Project**".
3. Fill in the fields as follows:
 - **Name:** UART Communication FPGA
 - **Description:** UART system with RX, logic comparison, and TX
 - **Type:** FPGA
4. Click "**Create**".

Blocks Used in the Project

The project uses the following blocks based on the provided Verilog modules:

- **uart_rx:** Responsible for receiving data from the UART port and indicating when a byte has been completely received.
- **uart_logic_const:** Compares the received byte with a predefined character and toggles an LED state.
- **uart_tx:** Sends data via the UART serial port.
- **Input/Output Pins:**
 - clk: Main system clock
 - uart_rx: UART data input
 - uart_tx: UART data output
 - b0: Reset button
 - led0: Indicator LED



Connecting the Blocks

Assemble the project with the following configuration:

uart_rx Block

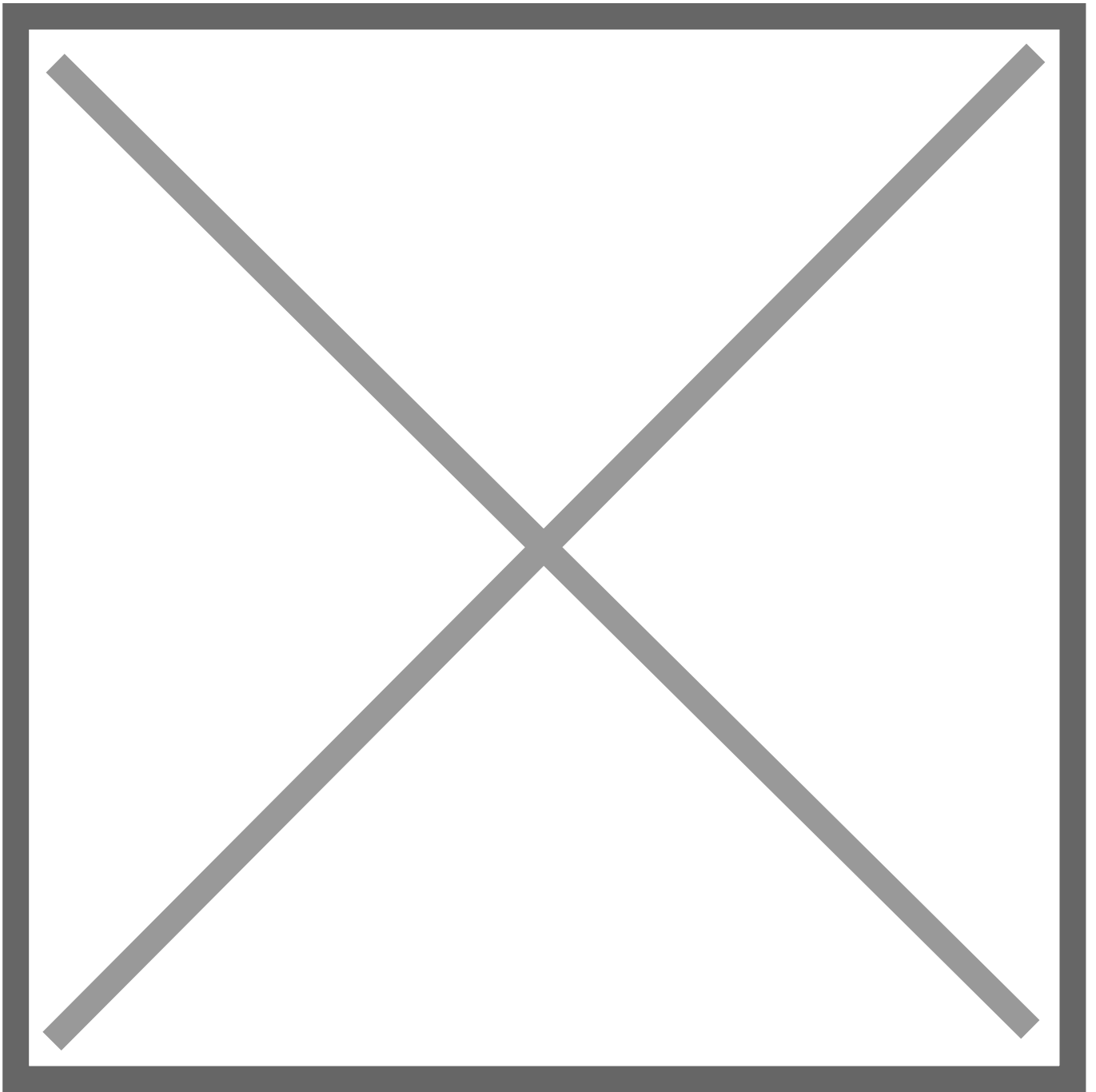
- **Inputs:**
 - clk → system clock
 - uartRx → UART input pin
- **Outputs:**
 - rxByte → connects to the logic block
 - byteReady → connects to both the logic block and uart_tx

uart_logic_const Block

- **Inputs:**
 - clk → system clock
 - rxByte → from rxByte output of uart_rx
 - byteReady → from byteReady output of uart_rx
 - compareChar → constant value (e.g., 8'h61 = 'a')
- **Output:**
 - signal → connects to LED (led0)

uart_tx Block

- **Inputs:**
 - clk → clock
 - reset → button (b0)
 - tx_data → receives rxByte from uart_rx
 - tx_data_valid → receives byteReady from uart_rx
- **Output:**
 - tx_pin → connects to the UART TX output pin
 - tx_data_ready → not used in this simple project



Final Connections:

Block/Pin	Connection
clk	All blocks
uart_rx	UART RX input
uart_tx	UART TX output
b0	Reset signal for uart_tx
led0	Output from signal of uart_logic_const

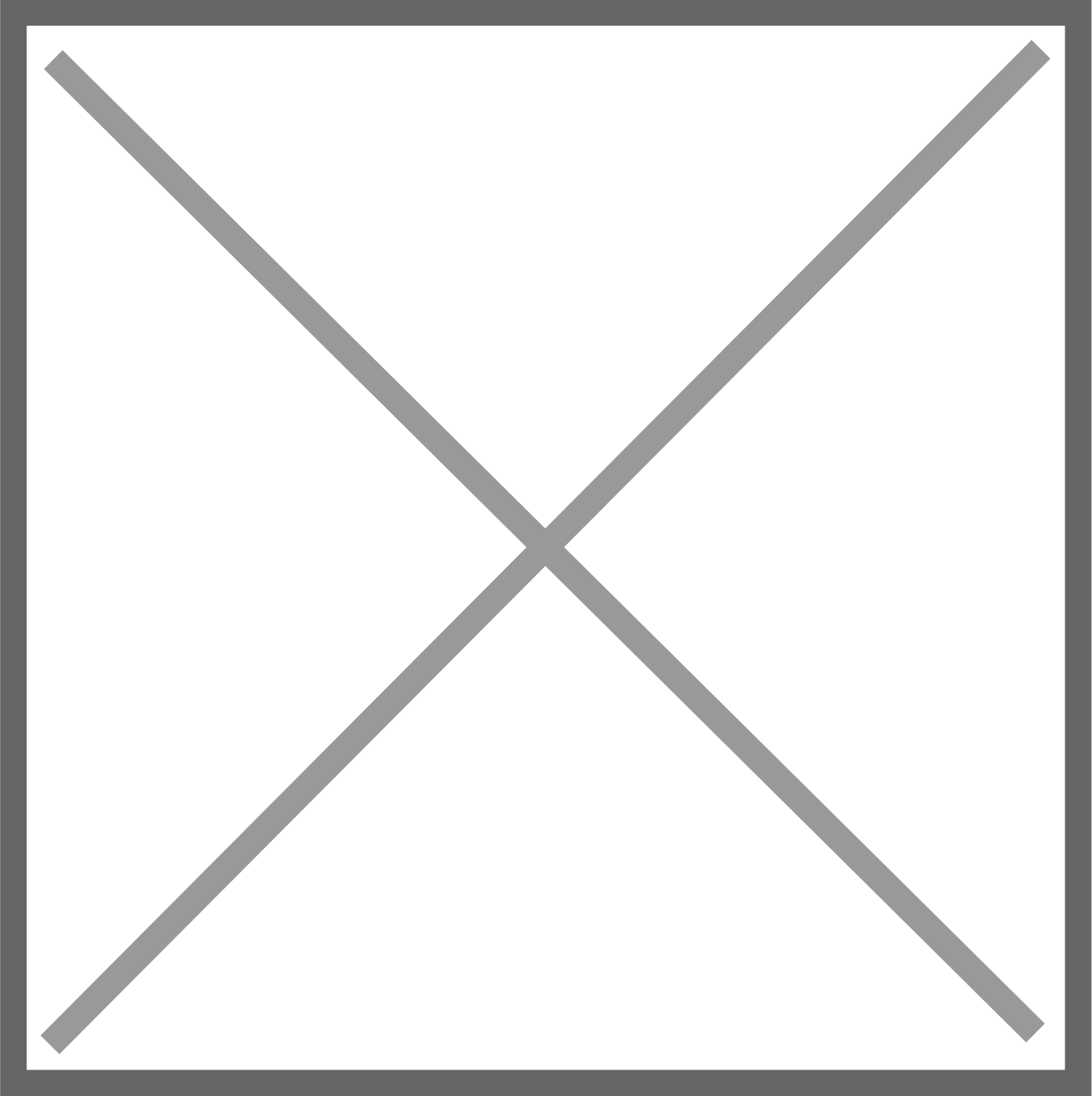
Project Simulation

1. Go to the Simulate tab in the top menu.
 2. Select Advanced Simulation.
 3. Click on Run Iverilog to compile and simulate.
 4. Check if the simulation runs without errors.
- If errors occur, review the block connections as described.

FPGA Synthesis and Programming

Once the simulation is validated:

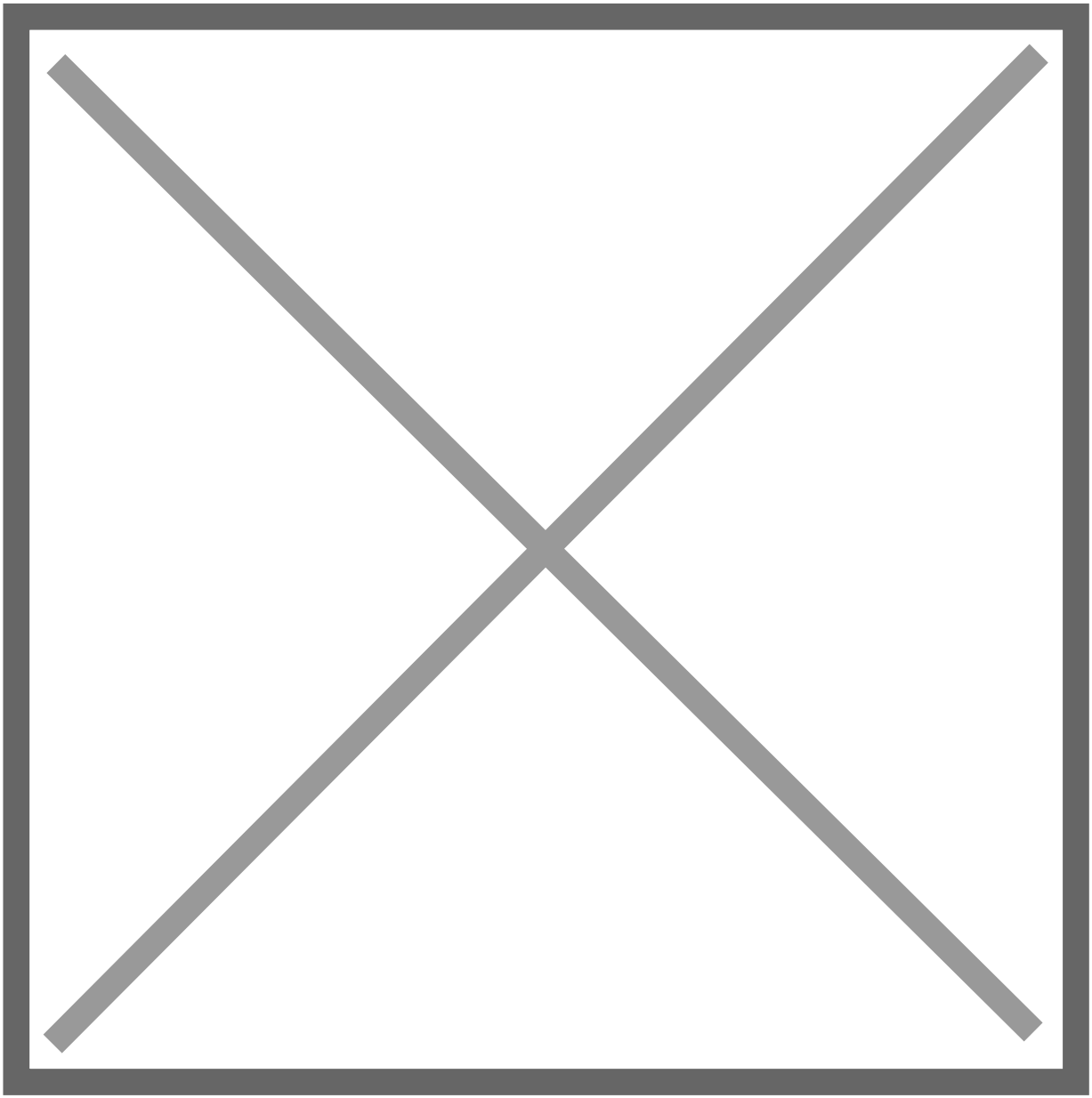
1. Go to the Synthesize tab.
2. Click Start Synthesis.
3. Connect your FPGA to the computer via USB.
4. Select the correct serial port (usually labeled “Enhanced”).
5. Click Flashing to program the FPGA.



Hardware Validation

1. Access the **Main** tab.
2. Click on **Serial Console**.
3. Set the baud rate to **115200**.
4. Send a character via the serial terminal.
 - If the character matches the predefined value (8'h61), the LED should toggle.
 - The same character will also be sent back through the TX pin (echo).





Wrapping Up

Congratulations! You've implemented a complete UART system with receiving, comparison, and data echo. This practice reinforces your understanding of serial communication, logical comparison, and module integration in FPGA using ChipInventor.

Try experimenting with different characters, adding more conditions, or expanding the system with multiple LEDs and control commands!