Implementation and Verification of Communication Protocol Using FPGA Development Board

By

Shyama Vagadia -

92200133005

A Project Report Submitted to

Marwadi University in Partial Fulfillment of the Requirements for the B. Tech in Information Communication Technology

Spetember 2025

MARWADI UNIVERSITY Rajkot-
Morbi Road, At & Po. Gauridad,
Rajkot- 360003, Gujarat, India.

I. Introduction

Communication protocols are very important in today's world because many devices need to communicate with each other, and protocols make this possible. Communication protocols are simply sets of rules and standards that define how devices exchange data across a network. They ensure that information is sent, received, and interpreted correctly by all participants.

For instance, PCs typically use **UART (Universal Asynchronous Receiver-Transmitter)**, while embedded systems like microcontrollers often rely on **SPI (Serial Peripheral Interface)** or **I²C (Inter-Integrated Circuit)**.
My project is the **implementation of communication protocols on FPGA**. In this project, I have implemented real-time communication between different devices using multiple communication protocols. The protocols I used are **UART** and **SPI**, both of which are serial communication protocols.

The working of the project is as follows: data is sent from one terminal to a USB-UART adaptor connected to port **COM3**. This adaptor transfers the data to the FPGA Cyclone II EP2C5T144 board through the **UART protocol**. UART is a point-to-point, 2-wire asynchronous communication protocol. From the FPGA, the data is then sent to the Arduino through the **SPI protocol**, where the FPGA acts as the **Master** device and the Arduino as the **Slave** device. SPI is a 4-wire synchronous protocol.

The Arduino is connected to another port, and in its terminal, the **hexadecimal** and **decimal** values of the received data can be seen. The Arduino increments the received data by 1 and sends it back to the FPGA, which then returns it to the adaptor in the same way. Finally, in the terminal (COM3) where the adaptor is connected, the incremented value is displayed. This forms a complete communication loop, demonstrating how data travels across different devices and how different protocols work together.
Additionally, the project integrates **simulation using ModelSim** and proposes a **web-based dashboard** for real-time visualization of communication waveforms.

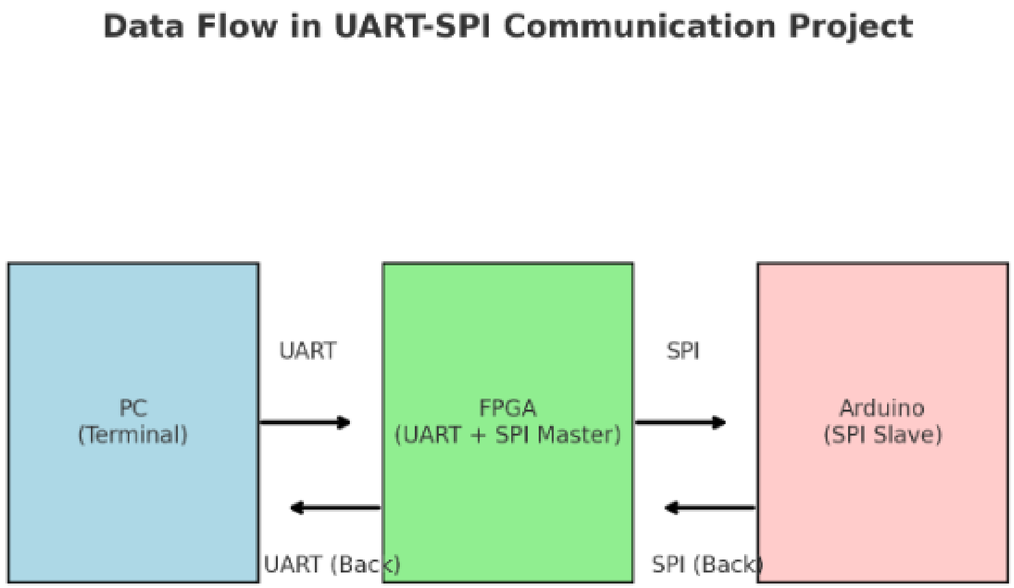
II. Project Overview

The project flow works like this:

1. A terminal on the PC is used to send data.
2. The data goes to the **USB-UART adaptor**.
3. From there, the data is received by the **FPGA through UART protocol**.
4. FPGA then sends this data to an **Arduino board using SPI protocol**.
    - In this case, FPGA acts as **SPI Master**.
    - Arduino acts as **SPI Slave**.

5. Arduino receives the data, increments it by 1, and sends it back to FPGA using SPI.
6. FPGA again sends this incremented data to the PC through UART.
7. On another terminal, we can see the incremented data in both decimal and hexadecimal form.

This shows a full communication cycle: **PC → FPGA (UART + SPI) → Arduino → FPGA → PC**.



Link of testing video:
https://drive.google.com/drive/folders/1en1WtyRCgajVP2L9R_DOUQ6kZULVtpDS?usp=drive_link

III. System Design

The system has three main blocks:

- **PC with terminal** (sending and receiving data)
- **FPGA board** (handling UART and SPI)
- **Arduino board** (acting as SPI slave, performing increment)

Implementation Highlights

- **Quartus** was used for coding and synthesizing Verilog modules for UART and SPI.
- **ModelSim** was used for simulating the communication. I also tested **I2C simulation** as an extra part.

- **Arduino** was programmed as the SPI slave to handle increment operation.
- **Web Dashboard** was created where ports are detected. The future plan is to show real-time waveforms of communication there.

IV. User Manual

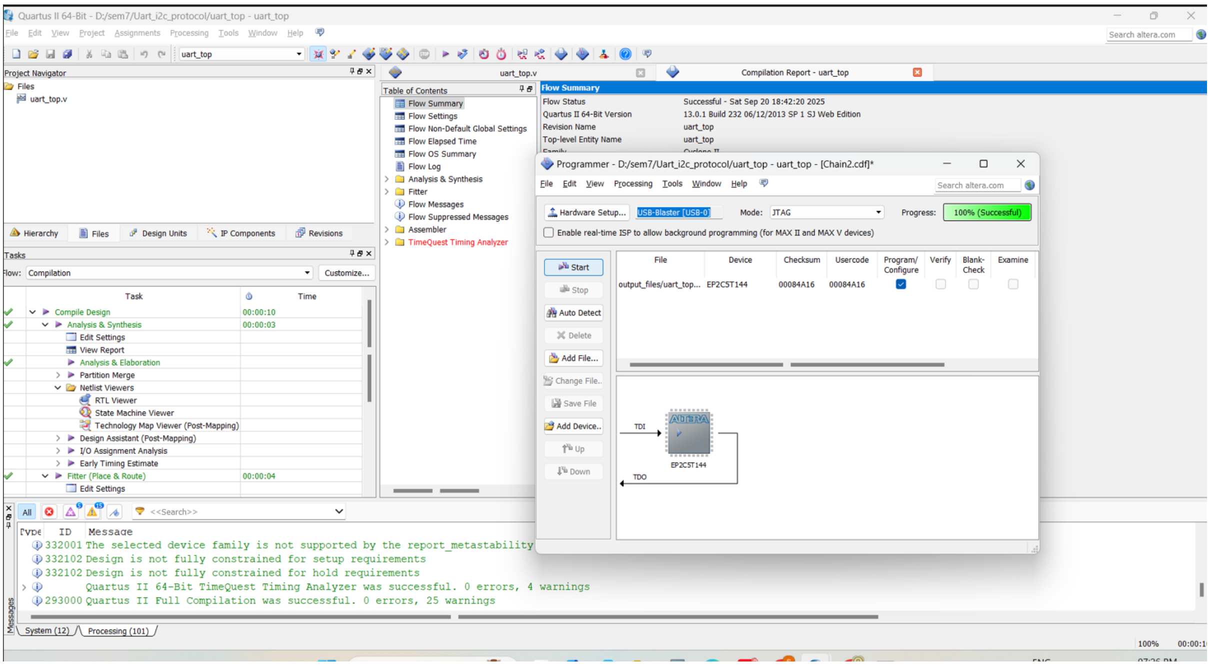
Setup and Connections

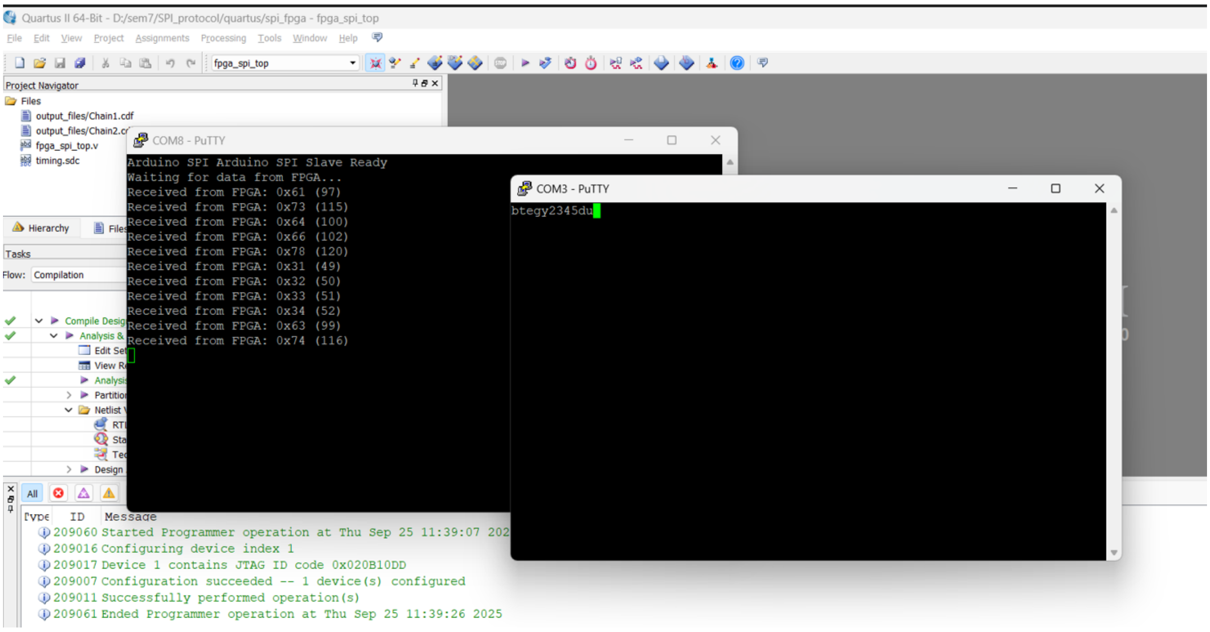
1. Connect **USB-UART adaptor** between PC and FPGA board.
2. Connect **SPI lines** (MOSI, MISO, SCLK, SS) between FPGA and Arduino.
3. Open **Quartus Programmer** to load FPGA bitstream.
4. Open **Arduino IDE** and upload the SPI slave code.
5. Open two terminals on PC (e.g., PuTTY):
    - One terminal for sending input data.
    - Second terminal for receiving incremented data.

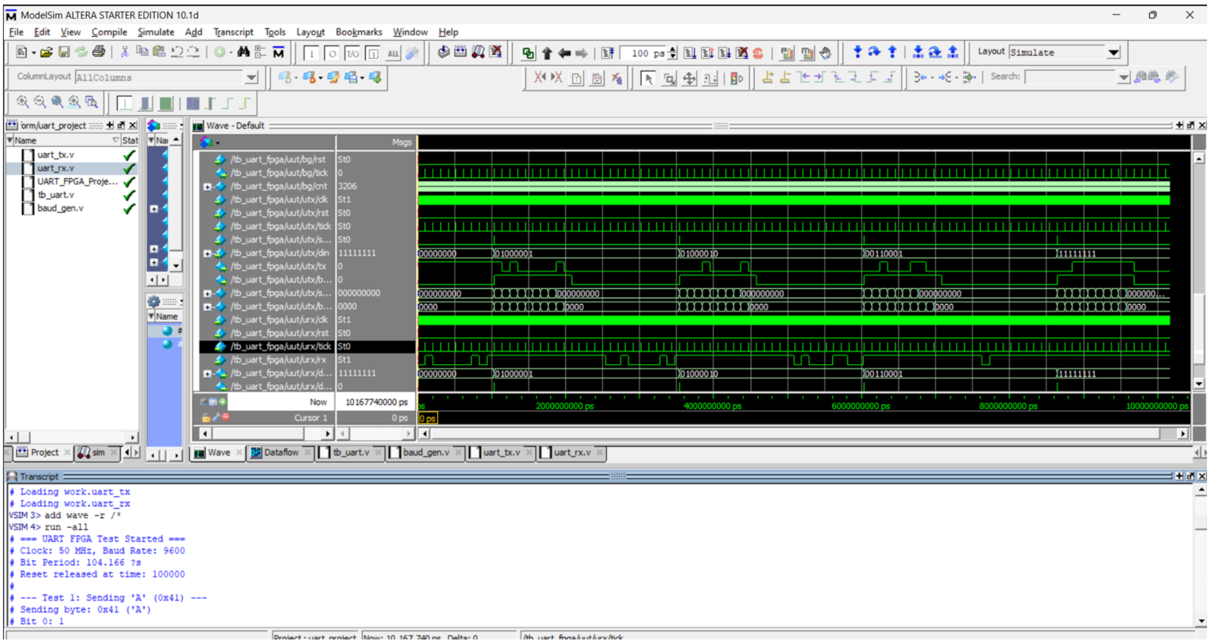
How to Run the Project

1. On the first terminal, type a number (example: 5).
2. FPGA receives it through UART and sends to Arduino via SPI.
3. Arduino increments the number (5 → 6) and sends it back to FPGA.
4. FPGA sends the new number back to PC through UART.
5. The second terminal shows the result in **decimal (6)** and **hexadecimal (0x06)**.

Screenshots:


Quartus II 64-Bit - D:/sem7/Uart_i2c_protocol/uart_top - uart_top
Compilation Report - uart_top
Flow Summary
Flow Status Successful - Sat Sep 20 18:42:20 2025
Quartus II 64-Bit Version 13.0.1 Build 232 06/12/2013 SP 1 SJ Web Edition
Revision Name uart_top
Top-level Entity Name uart_top
Programmer - D:/sem7/Uart_i2c_protocol/uart_top - uart_top - [Chain2.cdf]*
Hardware Setup... USB-Blaster [USB-0]
Mode: JTAG
Progress: 100% (Successful)
output_files/uart_top... EP2C5T144 00084A16 00084A16
332001 The selected device family is not supported by the report_metastability
332102 Design is not fully constrained for setup requirements
332102 Design is not fully constrained for hold requirements
Quartus II 64-Bit TimeQuest Timing Analyzer was successful. 0 errors, 4 warnings
293000 Quartus II Full Compilation was successful. 0 errors, 25 warnings


Quartus II 64-Bit - D:/sem7/SPI_protocol/quartus/spi_fpga - fpga_spi_top
COM8 - PuTTY
Arduino SPI Arduino SPI Slave Ready
Waiting for data from FPGA...
Received from FPGA: 0x61 (97)
Received from FPGA: 0x73 (115)
Received from FPGA: 0x64 (100)
Received from FPGA: 0x66 (102)
Received from FPGA: 0x78 (120)
Received from FPGA: 0x31 (49)
Received from FPGA: 0x32 (50)
Received from FPGA: 0x33 (51)
Received from FPGA: 0x34 (52)
Received from FPGA: 0x63 (99)
Received from FPGA: 0x74 (116)
COM3 - PuTTY
btegy2345du
209060 Started Programmer operation at Thu Sep 25 11:39:07 202
209016 Configuring device index 1
209017 Device 1 contains JTAG ID code 0x020B10DD
209007 Configuration succeeded -- 1 device(s) configured
209011 Successfully performed operation(s)
209061 Ended Programmer operation at Thu Sep 25 11:39:26 2025



V. Code Documentation

UART Modules

- uart_rx.v: Receives serial data from terminal.
- uart_tx.v: Transmits serial data back to terminal.

SPI Modules

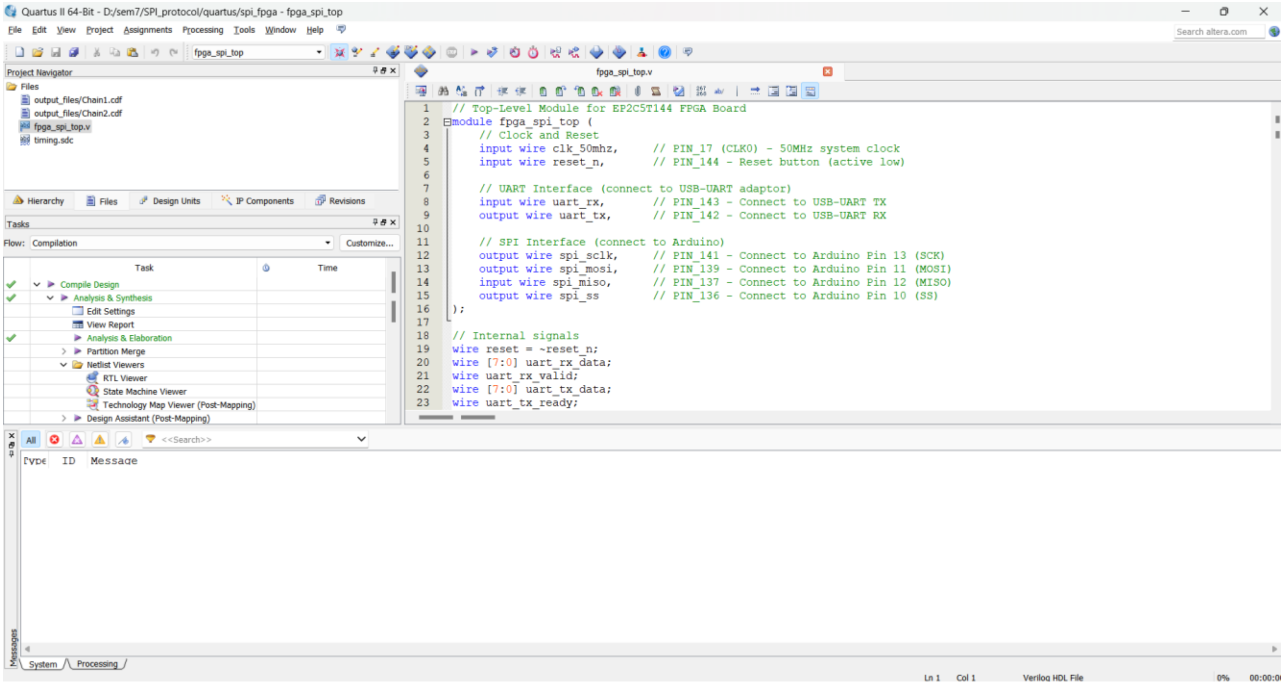
- spi_master.v: Implements SPI master logic in FPGA.
- spi_slave.ino: Arduino code for SPI slave, which increments data.

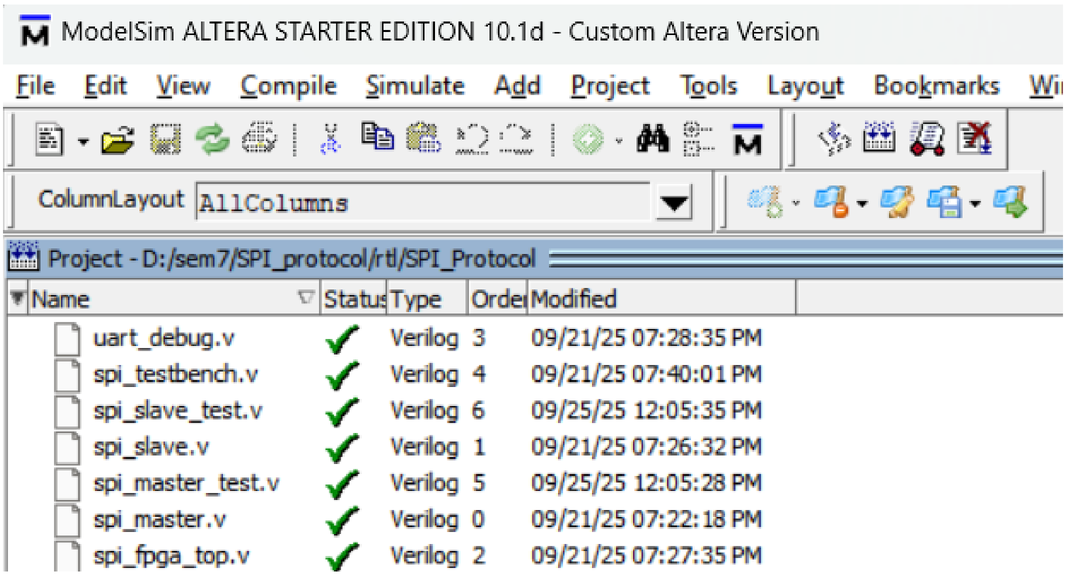
Top Module

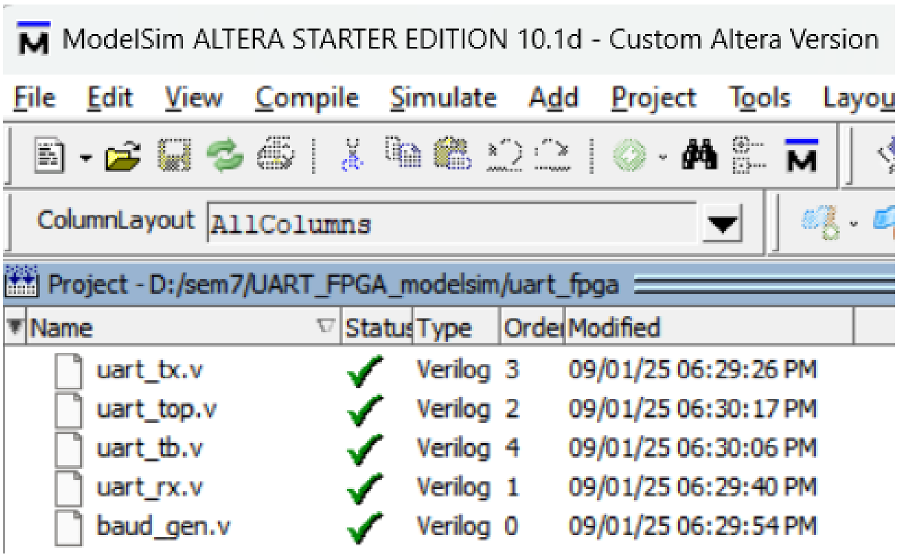
- fpga_comm_top.v: Integrates UART and SPI modules. Handles the full flow: UART RX → SPI Master → Arduino → SPI Master → UART TX.

Simulation Files

- uart_tb.v, spi_tb.v: Testbenches for ModelSim simulation.


Quartus II 64-Bit - D:/sem7/SPI_protocol/quartus/spi_fpga - fpga_spi_top
fpga_spi_top.v
// Top-Level Module for EP2C5T144 FPGA Board
module fpga_spi_top (
// Clock and Reset
input wire clk_50mhz, // PIN_17 (CLK0) - 50MHz system clock
input wire reset_n, // PIN_144 - Reset button (active low)
// UART Interface (connect to USB-UART adaptor)
input wire uart_rx, // PIN_143 - Connect to USB-UART TX
output wire uart_tx, // PIN_142 - Connect to USB-UART RX
// SPI Interface (connect to Arduino)
output wire spi_sclk, // PIN_141 - Connect to Arduino Pin 13 (SCK)
output wire spi_mosi, // PIN_139 - Connect to Arduino Pin 11 (MOSI)
input wire spi_miso, // PIN_137 - Connect to Arduino Pin 12 (MISO)
output wire spi_ss // PIN_136 - Connect to Arduino Pin 10 (SS)
);
// Internal signals
wire reset = ~reset_n;
wire [7:0] uart_rx_data;
wire uart_rx_valid;
wire [7:0] uart_tx_data;
wire uart_tx_ready;


ModelSim ALTERA STARTER EDITION 10.1d - Custom Altera Version
File Edit View Compile Simulate Add Project Tools Layout Bookmarks
ColumnLayout AllColumns
Project - D:/sem7/SPI_protocol/rtl/SPI_Protocol
Name Status Type Order Modified
uart_debug.v Verilog 3 09/21/25 07:28:35 PM
spi_testbench.v Verilog 4 09/21/25 07:40:01 PM
spi_slave_test.v Verilog 6 09/25/25 12:05:35 PM
spi_slave.v Verilog 1 09/21/25 07:26:32 PM
spi_master_test.v Verilog 5 09/25/25 12:05:28 PM
spi_master.v Verilog 0 09/21/25 07:22:18 PM
spi_fpga_top.v Verilog 2 09/21/25 07:27:35 PM


ModelSim ALTERA STARTER EDITION 10.1d - Custom Altera Version
File Edit View Compile Simulate Add Project Tools
ColumnLayout AllColumns
Project - D:/sem7/UART_FPGA_modelsim/uart_fpga
Name Status Type Order Modified
uart_tx.v Verilog 3 09/01/25 06:29:26 PM
uart_top.v Verilog 2 09/01/25 06:30:17 PM
uart_tb.v Verilog 4 09/01/25 06:30:06 PM
uart_rx.v Verilog 1 09/01/25 06:29:40 PM
baud_gen.v Verilog 0 09/01/25 06:29:54 PM