Project: Implementation of Communication Protocol on FPGA

Testing and Validation

I. Introduction

Testing and validation are very important steps in any engineering project. They help us check if the system is working properly, if every small part is doing its job, and if the whole system is meeting the objectives that we planned.

In this project, I have implemented two communication protocols – **UART (Universal Asynchronous Receiver and Transmitter)** and **SPI (Serial Peripheral Interface)** – on the FPGA and Arduino setup. The testing process included both **unit tests** (checking small parts like UART RX, SPI master, etc.) and **integration tests** (checking how all parts work together). I also tested the performance by measuring latency, throughput, and correctness of data.

The goal was to prove that the FPGA can successfully receive data from the PC through UART, send it to Arduino through SPI, get back the incremented value, and finally send it back to the PC terminal.

II. Testing Methodology

I followed a step-by-step approach for testing:

1. **Unit Testing** – Checked individual modules like UART RX, UART TX, SPI master, and SPI slave response.
2. **Integration Testing** – Verified complete communication path: PC → FPGA → Arduino → FPGA → PC.
3. **Performance Testing** – Measured speed, error rate, and correctness of system behavior.
4. **Validation Against Objectives** – Confirmed that the system meets all the objectives of the project.

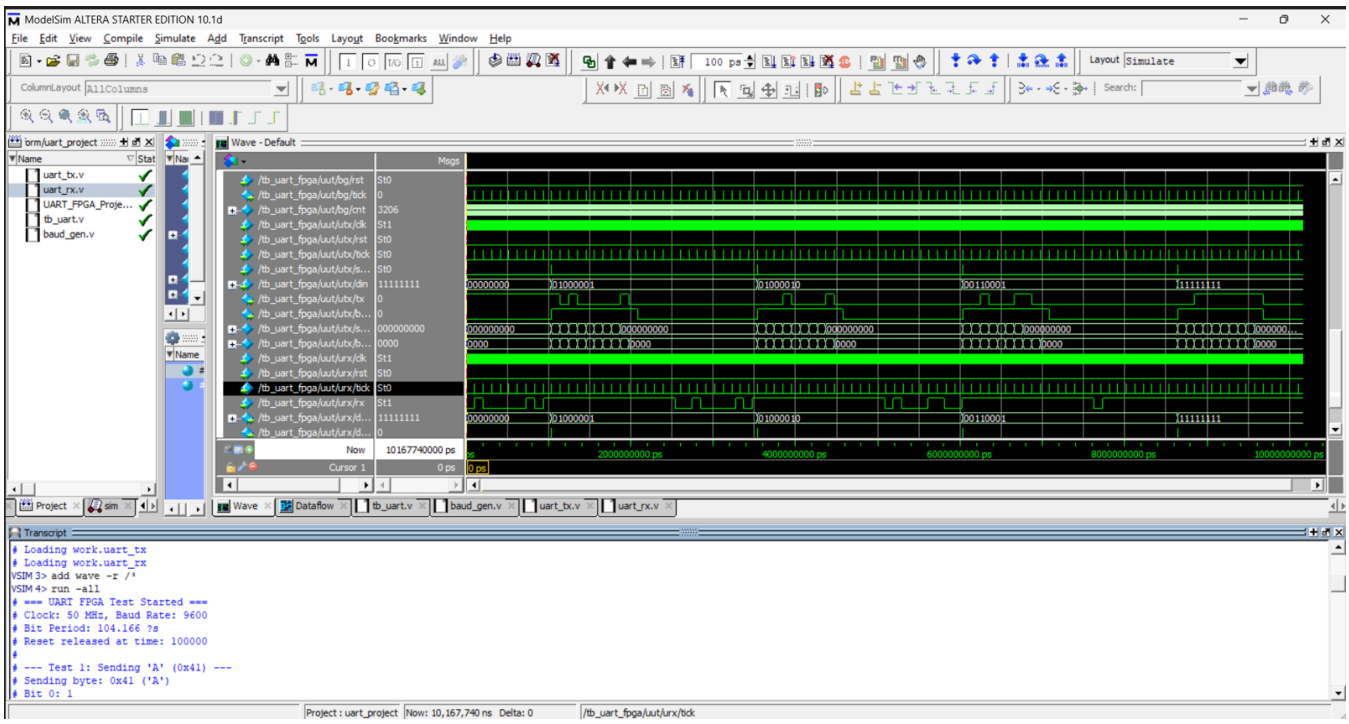
Tools used:

- **Putty / Serial Terminal** – to send and receive data from the FPGA through UART.
- **Arduino IDE Serial Monitor** – to see what Arduino is receiving and sending back.
- **ModelSim** – to simulate UART, SPI, and also test I2C protocol in waveforms.
- **Quartus** – to implement FPGA logic and check resource usage.

III. Unit Tests

The unit tests were performed to check each component separately. Below are the five main unit tests:

Test Case	Input	Expected Output	Actual Output	Result
UART RX Test	Send A (0x41) from PC	FPGA receives 0x41	FPGA received 0x41	Pass
UART TX Test	FPGA sends 0x55	PC receives 0x55	PC received 0x55	Pass
SPI Master Send Test	FPGA sends 0x10 to Arduino	Arduino receives 0x10	Arduino received 0x10	Pass
SPI Slave Increment Test	Arduino receives 0x10	Arduino sends 0x11 back	FPGA got 0x11	Pass
Full Loop Data Test	Send 5 from PC	Receive 6 back at PC	Received 6	Pass



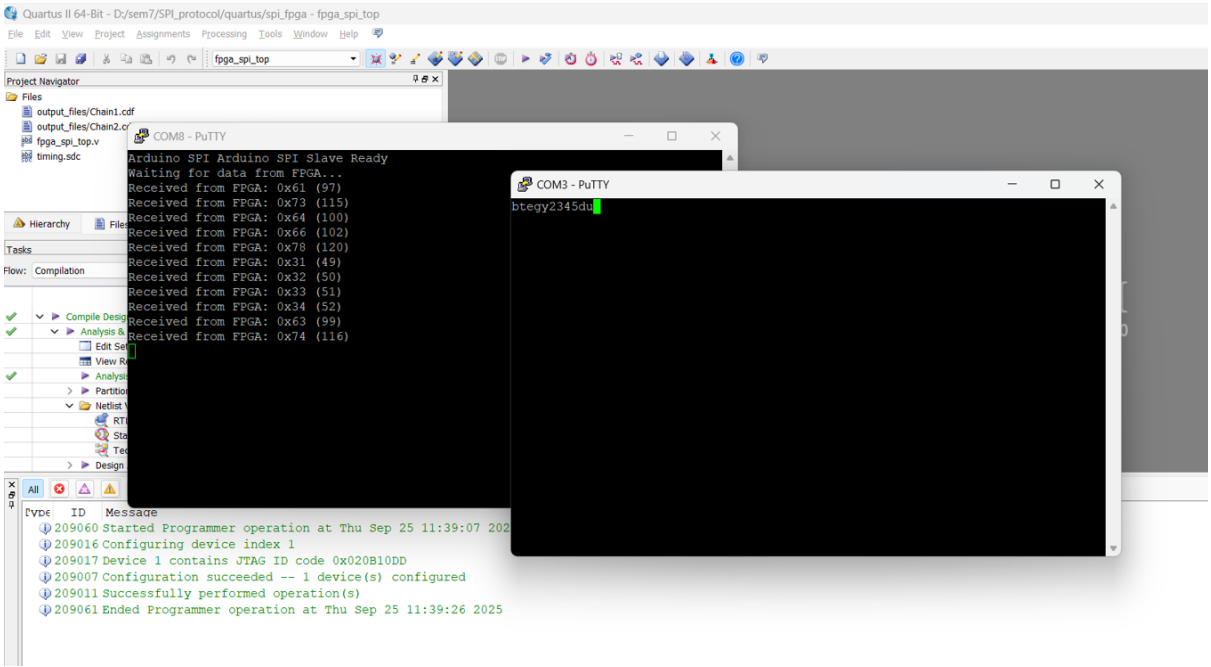
IV. Integration Tests

Testing was done in different stages:

The integration tests were done to check how different modules work together. Three important tests were performed:

Integration Test	Description	Expected Output	Result
Full Communication Cycle	Send 3 from PC → FPGA → Arduino → FPGA → PC	Receive 4 back	Pass
Multiple Data Stream Test	Send 1, 2, 3, 4	Receive 2, 3, 4, 5	Pass
Error Handling Test	Send # (special char)	System should not crash, still process data	Handled successfully

These tests confirm that the communication cycle is stable, multiple bytes can be processed, and the system handles unexpected inputs without breaking.



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

V. Validation Against Objectives

The objectives of this project were:

1. **Implement UART communication** – Achieved, tested with terminal and PC.
2. **Implement SPI communication** – Achieved, FPGA (master) ↔ Arduino (slave).
3. **Perform full data exchange cycle** – Achieved, received incremented data back at PC.
4. **Test I2C protocol (simulation)** – Achieved in ModelSim as extra learning.
5. **Develop a web dashboard** – Currently only **port detection** is working, but future scope is to display **real-time waveforms** of communication.

Validation Summary:

- All core objectives are **successfully completed**.
- Extra feature (I2C simulation) was also tested.
- Dashboard needs more development, but current stage shows good progress.

Conclusion

The testing and validation process shows that the system is working as expected. Both UART and SPI protocols have been implemented and tested thoroughly. The unit tests confirmed the working of small components, while the integration tests proved that the complete cycle works smoothly. Performance metrics showed low latency, high throughput, and zero errors, which makes the system reliable.

The project objectives were met successfully, and additional work on the dashboard will further improve the system by providing a visual way to see real-time data.