Department of Electrical Engineering

ASIC/FPGA Chip Design

Laboratory Manual

ASIC/FPGA Chip Design Laboratory Manual

Dr. Mahdi Shabany

Spring 2025

Contents

Lab 4 - UART Protocol

1.1 Objective

In this lab, you will implement both sender and receiver of the UART protocol. You will connect two FPGA boards to each other and transmit data between them via the UART protocol.

1.2 Introduction

Universal Asynchronous Receiver/Transmitter (UART) is a widely-used serial communication protocol that enables two devices to exchange data using only two wires: one for transmission (TX) and one for reception (RX). In many cases, there is also a third wire that connects the ground of two devices. Unlike synchronous protocols, UART does not require a shared clock signal. Instead, it relies on both ends agreeing on a common configuration, especially the baud rate, which defines the number of bits transmitted per second.

A typical UART frame consists of the following:

- Start Bit: A single low (0) bit indicating the beginning of transmission.
- Data bits: Usually 7 to 9 bits that carry the actual payload, sent least significant bit (LSB) first.
- Optional Parity Bit: A simple error-checking mechanism that can be even or odd.
- Stop Bit(s): One or more high (1) bits signaling the end of the frame.

The timing of each bit is critical and is governed by the baud rate (e.g. 9600, 115200), which must be matched on both sender and receiver sides. Because UART is asynchronous, it is sensitive to timing mismatches and electrical noise. To improve robustness, the receiver samples each bit in the middle of its expected window, reducing the chance of reading transient or noisy values. Also, there may be some instability on the receiver side due to the noise and asynchronous clocks. To solve this issue, there is usually a buffer (2 or 3 flip flops) in the receiver to stabilize the received bits before inferring. This problem is called CDC (Clock Domain Crossing), which you will become familiar with at the end of the semester.

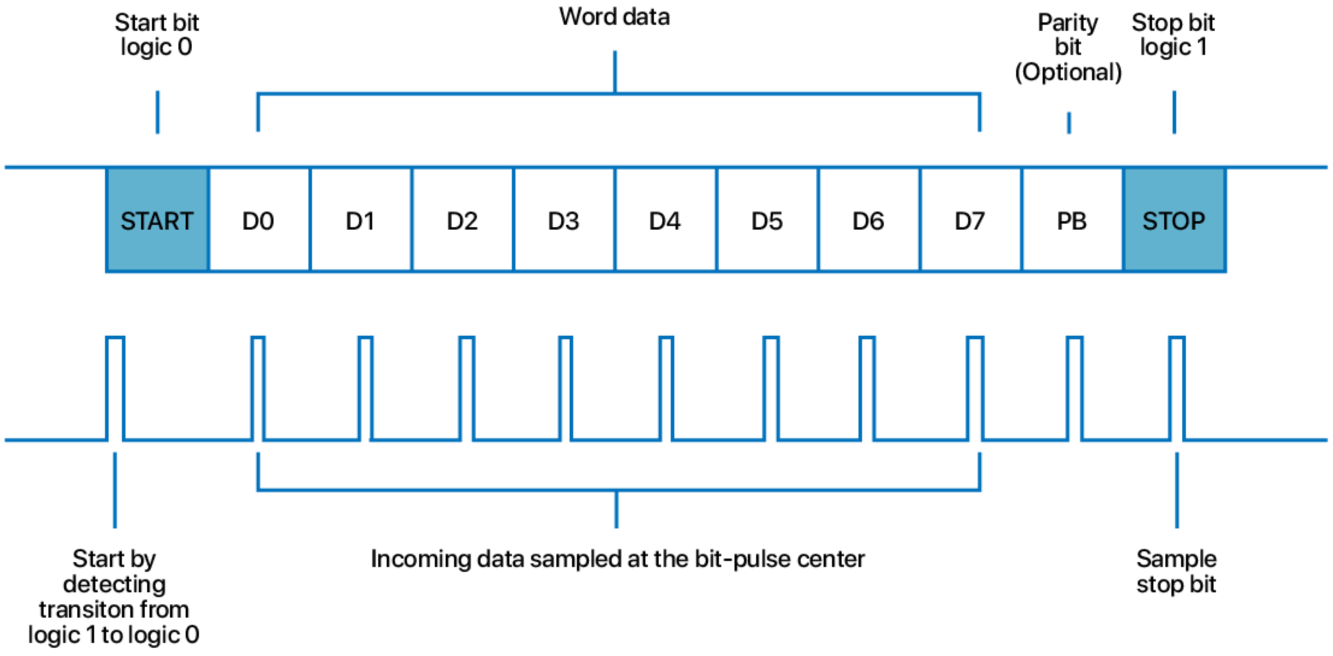

Figure 1.1: UART frame sample

The advantages of UART include simplicity, low overhead, and the minimal number of wires required. It is ideal for short-distance communication and debugging. However, limitations include lack of clock synchronization, fixed baud-rate configurations, and limited data rates compared to high-speed serial protocols.

Real-world applications of UART include communication between microcontrollers, interfacing with GPS modules, Bluetooth devices, serial consoles, and debugging embedded systems. It is also commonly used in FPGA development for logging and control purposes.

1.3 Instructions

In this lab, you will work with two FPGA boards. We call the board with extension board, board A and the other one board B.

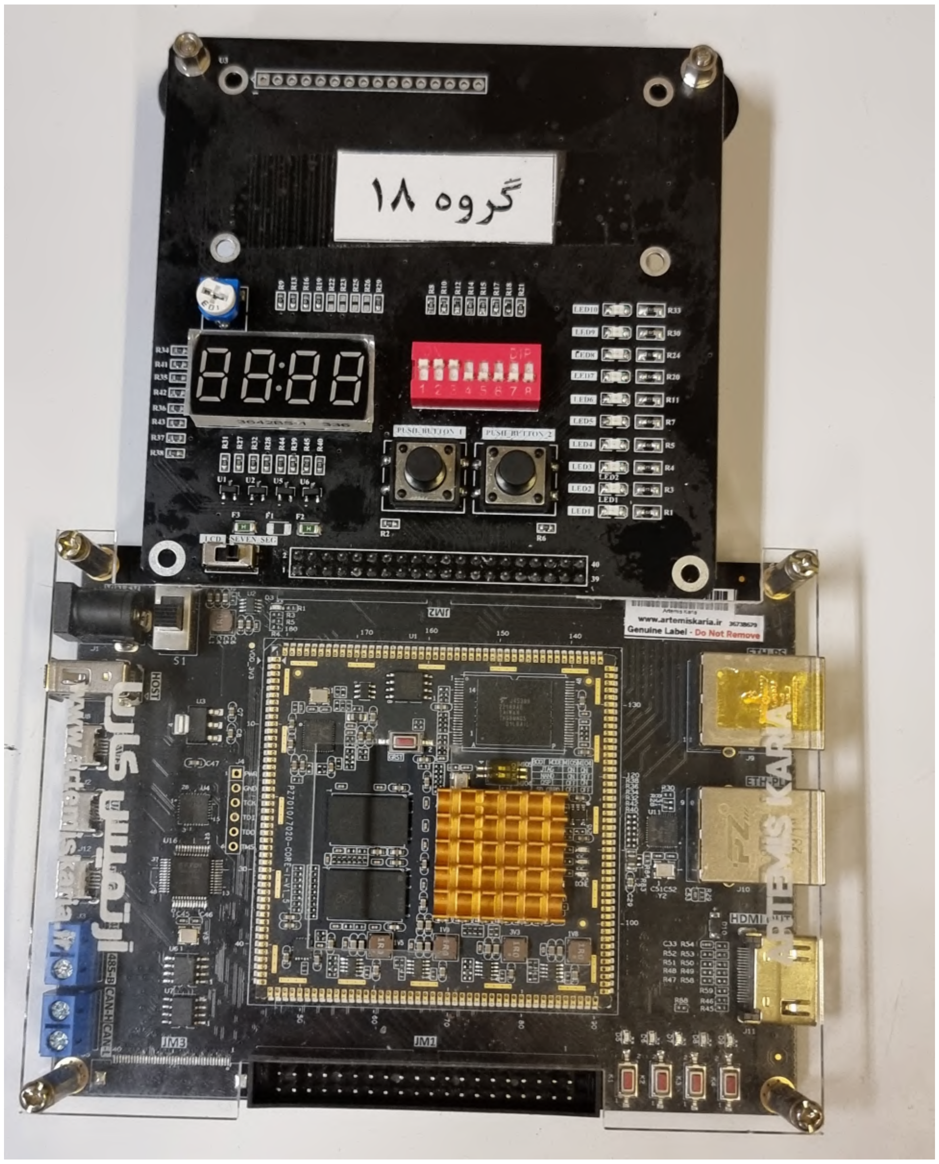

Figure 1.2: Board A

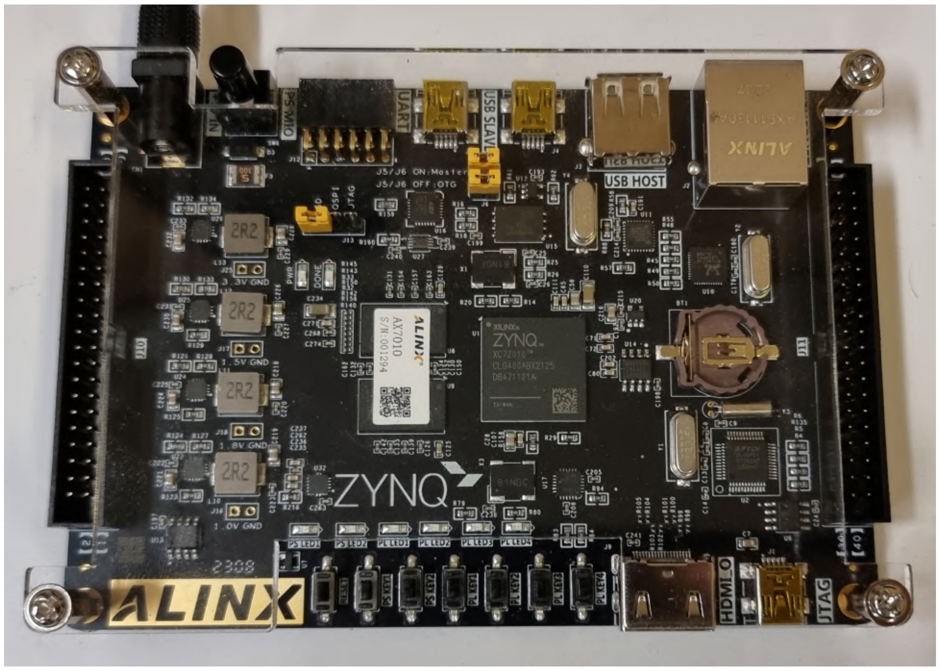

Figure 1.3: Board B

1.3.1 Preparation

As mentioned above, the UART protocol needs 3 wire connections, TX, RX and ground. You need to connect TX of board A to RX of board B and vise versa. Also, you need to connect the ground of two boards to each other. You will use the JM1 40-pin connector of board A and the J11 40-pin connector of board B and jumper wires. You can choose the TX and RX pins yourself, but make sure that they are correctly assigned with the datasheet.

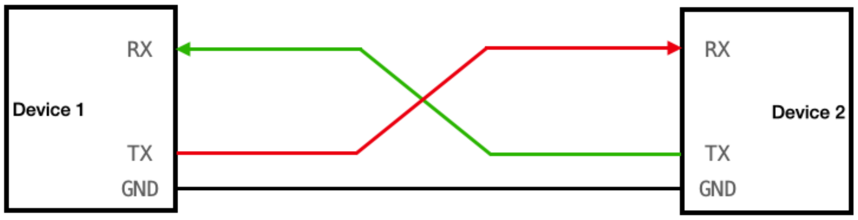

Figure 1.4: UART connection

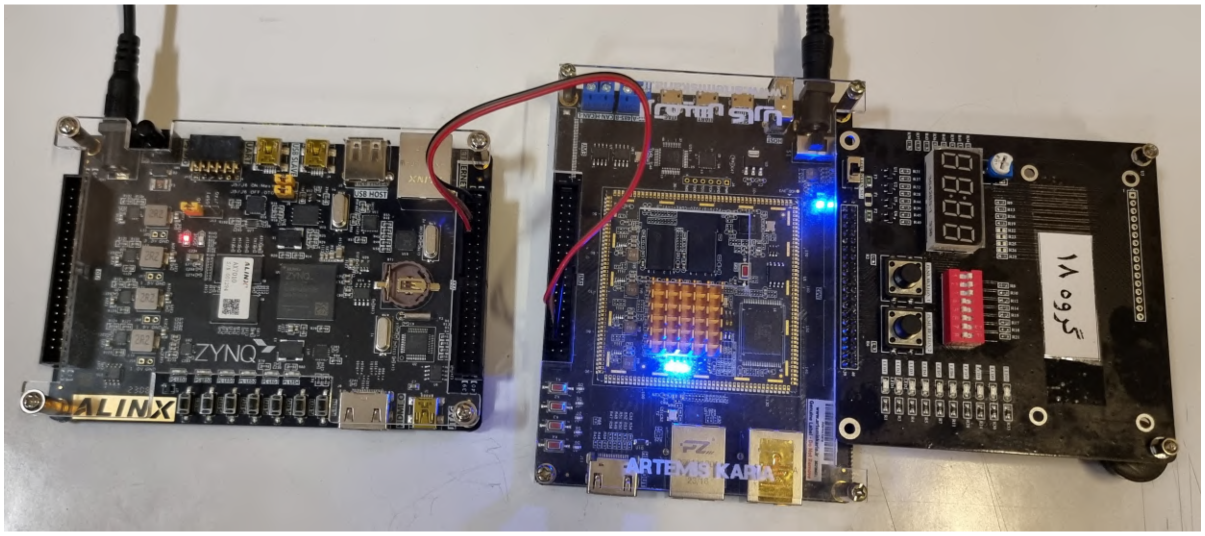

Figure 1.5: Sample connection of two boards

1.3.2 4 digit 7-segment display

For the first step, you should write a module that gets a 28 bit data as the input, and outputs the 4 bit 'dig' and 7 bit 'abcdefg'. This module should show 7 bit data on a digit at a time and iteratively change the data and digit to show 4 digit data completely. To verify that your module works, show '1234' on the four-digit 7-segment.

1.3.2.1 Bonus

Parameterize your module in a way that can be configured with different clock frequencies and custom refresh rates.

1.3.3 Simple UART data transmission

The main configuration of the UART protocl is as follows:

- Baud rate: 9600
- Clock frequency: 50 MHz
- Number of data bits: 7
- Parity bit: Even parity
- Number of stop bits: 1

★There is a good bonus if these configurations are parametrized in your modules.

In this part, you should write both sender and receiver of the UART protocol. Then, you should program the sender module on board A and the receiver module on board B. The sender must send deep switches 1 to 4 states when button 1 is pressed, and the receiver must turn the 4 leds on or off based on the states received and keep their state until the next data. For 7 bit data bits, you may send the 3 arbitrary bits for the remaining bits. For example, if the deep switches 1 to 4 are '1001', and button 1 is pressed, the sender must send '0,1001,000,0,1' as an UART packet (start bit, data bits, three 0 bits, parity bit, one stop bit), and at the receiver, leds 1 and 4 must turn on and leds 2 and 3 turn off and keep their states.

1.3.4 Duplex communication

In this part, in addition to the previous functionality, you should also send data from board B to board A. To do so, you should write 3 counter modules with different counting speeds (3 Hz, 13 Hz, and 23 Hz). Then, by pressing keys 1 to 3, send the counter 1 to 3 value using the UART protocol. The data must be converted to 4 digit 7-segment format. At the receiver, the data received must be shown on the 4 digit 7-segments. You may use the module you wrote in part 1.3.2. Also, implement a reset functionality so that by pressing the key 4, all counters reset and start counting from 0.

Department of Electrical Engineering