uart_loopback on FPGA

1. Objective

The purpose of this project is to implement a **UART (Universal Asynchronous Receiver/Transmitter) loopback** on an **FPGA**. In this setup, the **transmitted data (TX)** is directly fed back into the **received data (RX)**, allowing for **self-testing** of the UART functionality.

---

2. FPGA Board and Connections

This project is implemented on the **VSDSquadron FPGA Mini**, which is based on the **Lattice iCE40** series.

Pin Configuration

Signal	FPGA Pin	Description
clk	35	System clock (12 MHz)
tx	6	UART Transmit (TX)
rx	7	UART Receive (RX)

- The **tx (transmit)** and **rx (receive)** pins are internally connected within the FPGA to enable **loopback mode**.
- The system operates on a **12 MHz clock**, which is used to generate the required baud rate for UART communication.

---

3. Working Principle

1. **Data Transmission**:
    - The FPGA sends a **serial data stream** via the **TX** pin.
    - The data is encoded according to the **UART protocol** (Start bit, Data bits, Stop bit).
2. **Loopback Mechanism**:
    - Instead of sending the data to an external device, the **TX pin is internally connected to the RX pin**.
    - This allows the FPGA to immediately receive the same data that it transmitted.
3. **Data Reception**:

- The received data is decoded and stored.
- If the received data matches the transmitted data, it confirms that the **UART communication is functioning correctly**.

---

4. Verilog Code Overview

UART Loopback Module (uart_loopback.v)

This module integrates the UART transmitter and receiver in a loopback configuration.

```
module uart_loopback (
    input clk,        // 12 MHz system clock
    output reg tx,    // UART Transmit
    input rx          // UART Receive (Loopback)
);

reg [7:0] data = 8'b10101010;  // Sample Data
reg [3:0] state = 0;           // State Machine

always @(posedge clk) begin
    case (state)
        0: tx <= 0;        // Start bit
        1: tx <= data[0];  // Transmit bit 0
        2: tx <= data[1];  // Transmit bit 1
        3: tx <= data[2];  // Transmit bit 2
        4: tx <= data[3];  // Transmit bit 3
        5: tx <= data[4];  // Transmit bit 4
        6: tx <= data[5];  // Transmit bit 5
        7: tx <= data[6];  // Transmit bit 6
        8: tx <= data[7];  // Transmit bit 7
        9: tx <= 1;        // Stop bit
       10: state <= 0;     // Reset state for next transmission
    endcase

    state <= state + 1;
end

endmodule
```

Code Explanation:

- **State Machine**: Handles **data transmission** by shifting out bits one at a time.
- **Start & Stop Bits**: Ensures proper UART frame structure.
- **Loopback Mode**: The tx output is read back into rx for verification.

---

5. FPGA Implementation Steps

Step 1: Synthesis (Yosys)

```
yosys -p "read_verilog uart_loopback.v; synth_ice40 -json uart_loopback.json"
```

- Converts the Verilog design into FPGA logic.

Step 2: Place & Route (NextPNR)

```
nextpnr-ice40 --json uart_loopback.json --pcf constraints.pcf --asc uart_loopback.asc --
package hx8k
```

- Maps the synthesized design to FPGA hardware resources.

Step 3: Generate Bitstream (IcePack)

```
icepack uart_loopback.asc uart_loopback.bin
```

- Creates the **binary file** required to program the FPGA.

Step 4: Flash the FPGA (IceProg)

```
iceprog uart_loopback.bin
```

- Uploads the program to the FPGA.
- The UART loopback test begins.

---

6. Simulation & Debugging

Simulation (Icarus Verilog)

```
iverilog -o uart_loopback_tb.vvp uart_loopback_tb.v uart_loopback.v
```

```
vvp uart_loopback_tb.vvp
```

- Tests the design in a simulation environment.

View Waveforms (GTKWave)

```
gtkwave uart_loopback_tb.vcd
```

- Confirms correct transmission and reception of UART data.

---

7. Conclusion

- **Successful UART Loopback Implementation**: The FPGA transmits and receives serial data correctly.
- **Open-Source Tools Used**: Yosys, NextPNR, IceProg, Icarus Verilog.
- **Validated via Simulation and Hardware Testing**.

This project demonstrates **fundamental FPGA-based serial communication**, which can be extended for real-world UART applications.