**Implementing a UART Transmitter Based on Sensor Inputs**

**Objective**

The aim of this project is to build a system that can **send sensor data from an FPGA to another device** (like a computer or microcontroller) using **UART (Universal Asynchronous Receiver-Transmitter)** communication. This allows real-time sensor values to be monitored externally, making it ideal for applications where live data tracking is essential.

**1. Understanding the Code and How It Works**

**What Does the Module Do?**

The main module here is called sense\_uart\_tx. It takes in **sensor readings** and sends them out as **serial data** over UART. It’s structured into four main parts:

* **Reading and preparing sensor data**
* **Generating a 9600 baud UART clock**
* **Handling UART data transmission**
* **Using a state machine to manage the process**

**Step-by-Step Operation**

**Sensor Data Sampling**

* The FPGA reads **32-bit sensor data** at regular intervals.
* When new data is ready, a signal called data\_valid goes high.
* This triggers the transmission process, and the sensor value is loaded into a buffer.

**Generating the UART Clock**

* To match standard UART speed, we need a **9600 baud clock**.
* This is done using a **counter-based clock divider**, which creates accurate timing for each bit being sent.

**Sending the Data Over UART**

Here’s how the UART frame is structured and sent:

* **Start Bit**: A logic **0** marks the beginning of transmission.
* **Data Bits**: The sensor value is sent in **8-bit chunks** (1 byte at a time).
* **Stop Bit**: A logic **1** indicates the end of the frame.
* A **state machine** controls this process, moving from one step to the next in sync with the baud clock.

**Status Signals**

* **tx\_done**: Goes high when the current transmission is finished.
* **ready**: Lets the system know it’s ready to send the next data point—this prevents data from getting lost if new sensor readings come in too quickly.

**2. Breaking Down the Ports**

**System Signals**

* **clk**: Main clock input that drives the system.
* **reset\_n**: Resets the system (active-low).

**Sensor Inputs**

* **sensor\_data [31:0]**: The actual sensor readings that need to be transmitted.
* **data\_valid**: Tells the module when new data is available.

**UART Output**

* **tx\_out**: The UART data line, connected to the device receiving the sensor values.

**Control Signals**

* **tx\_start**: Starts the UART transmission.
* **tx\_done**: Goes high when the data has been fully sent.
* **ready**: Indicates the system is idle and ready for the next transmission.

**3. Internal Logic**

**How the State Machine Works**

The module uses a **finite state machine (FSM)** to handle different stages of data transmission:

1. **IDLE**: Waits for data\_valid to go high.
2. **START**: Sends the start bit (0).
3. **DATA**: Sends the data, 8 bits at a time.
4. **STOP**: Sends the stop bit (1).
5. **DONE**: Signals tx\_done and returns to IDLE.

**Baud Rate Generator**

* This part of the module divides down the main clock to generate a **9600 Hz UART clock**—perfectly timed for reliable data transmission.

**Shift Register**

* The 32-bit sensor data is stored in a register.
* During each UART frame, **8 bits are shifted out** and sent until the whole 32-bit value is transmitted.

**A diagram of a sensor data transfer

Description generated with high confidence4. System Architecture**

**Block Diagram**

This block diagram illustrates an FPGA-based UART transmission system for sensor data.

Sensor Section

Sensor Interface → Captures raw data.

Data Processing → Filters/formats the data.

Data Buffer → Stores processed data before transmission.

FPGA Section

Baud Rate Generator → Generates clock for UART.

Data Buffer → Stores sensor data for transmission.

TX Shift Register → Shifts data bit by bit.

UART TX Logic → Handles start, data, and stop bits.

State Machine → Controls the transmission sequence.

Data Flow

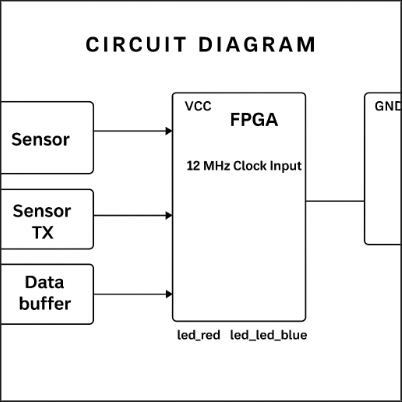
Sensor collects and processes data.

FPGA buffers and prepares it for UART.

TX Shift Register formats the data.

UART TX Logic transmits it serially.

State Machine ensures correct timing.

****A close up of a circuit board

Description generated with high confidence**Circuit Diagram**

**5. Synthesis & Programming**

Here’s how to build and test the design on your FPGA:

**Step 1: Clone the Project Repository**

bash

CopyEdit

git clone

**Step 2: Build the Bitstream**

bash

CopyEdit

make build

This will compile the code and create top.bin—the bitstream file you’ll upload to your FPGA.

**Step 3: Flash the Bitstream to the FPGA**

bash

CopyEdit

sudo make flash

This uploads your design to the FPGA board.

**Step 4: Test the UART Output**

bash

CopyEdit

sudo make terminal

Once the terminal opens, you should see **sensor data being printed**—this confirms the UART transmission is working properly at 9600 baud.