Lab 8: UART Controller

UART Transmitter

Objective

The objective of this lab is to design and implement a **UART Transmitter** module capable of serial communication with configurable baud rates, transmit buffering using FIFO, and proper state machine control. The transmitter must follow the UART protocol format with start, data, parity, and stop bits.

Specifications

- **Configurable Baud Rate**: 9600, 19200, 38400, 115200 bps.
- **Frame Format**:
    - 8-bit data.
    - 1 start bit.
    - 1 stop bit.
    - Optional parity (even/odd).
- **Transmit FIFO**: Configurable depth (default: 16 entries).
- **Status Flags**: Busy, FIFO full, FIFO empty, almost full, almost empty.
- **Full Integration**: Baud rate generator, controller FSM, FIFO, and datapath combined in a single top module.

Design Methodology

1. **Baud Rate Generator**
    - Generates `tick_tx` and `tick_rx` pulses based on system clock.
    - Provides timing for bit transmission and reception.

    - Supports configurable baud rates by dividing the system clock.

2. **FIFO Buffer**
    - Stores data before transmission to prevent data loss.
    - Provides full, empty, almost full, and almost empty status.
    - Supports concurrent read/write operations.
3. **Controller (FSM)**
    - Controls data flow from FIFO to datapath.
    - States:
        - **IDLE**: Waits for new data.
        - **LOAD**: Loads data from FIFO into shift register.
        - **TRANSMIT**: Serially shifts data bits out with proper framing.
    - Generates control signals for start, load, transmit, and read enable.
4. **Datapath (Transmitter Logic)**
    - Constructs UART frame: start bit, data bits, optional parity, and stop bit(s).
    - Shifts bits out on the `tx` line synchronized with baud ticks.
    - Generates `load_done` and `transmit_done` flags to synchronize with FSM.
5. **Top Module Integration**
    - Connects all submodules: baud generator, FIFO, controller, and datapath.
    - Ensures proper synchronization between modules.
    - Provides final transmit output (`tx`) and FIFO monitoring (`count`).

Simulation and Verification

- **Testbench Setup**:
    - System clock: 50 MHz.
    - Reset applied initially, then multiple data values written into FIFO.
    - Transmission observed for three test bytes (`0xFF`, `0x3C`, `0xF0`).
- **Verification Points**:
    - **Baud Rate Generation**: Confirmed correct timing of `tick_tx` and `tick_rx`.
    - **FIFO Operation**: Verified correct storage and retrieval of data with proper count tracking.
    - **State Transitions**: FSM correctly moved through IDLE → LOAD → TRANSMIT → IDLE.
    - **Frame Format**: Each byte transmitted with correct start, data, parity, and stop bits.
    - **Flags**: Busy asserted during transmission; FIFO flags updated accurately.
- **Waveform Analysis**:
    - Observed clean transitions of start, data, and stop bits on `tx` line.
    - Verified proper timing alignment with baud rate ticks.
    - Confirmed multiple bytes transmitted sequentially without data loss.

Results

- UART transmitter successfully transmitted multiple bytes (`0xFF`, `0x3C`, `0xF0`).
- Output waveform matched expected UART frame format.
- Baud rate timing aligned with configured divisor.
- FIFO correctly buffered data and prevented loss during back-to-back writes.

- FSM control signals (`tx_start`, `load`, `tx_done`) worked as intended.

Conclusion

The **UART Transmitter** was successfully designed, implemented, and verified. It demonstrated proper serial data transmission with configurable baud rates and FIFO buffering.

This lab enhanced understanding of:

- UART protocol and frame format.
- Baud rate generation using clock dividers.
- FSM-based control for serial communication.
- FIFO-based buffering in communication systems.