Name: Vakkanti Lakshmi Revanth Kumar
Ph. No: 6281974219
Email: revi4922251@gmail.com

UART Packetizer Design with FSM and FIFO for Serial Transmission

INTRODUCTION:

This project addresses the design and implementation of a UART Packetizer system, specifically tailored to simulate a UART protocol using Finite State Machine (FSM) control and FIFO buffering, as required in the Digantara Junior FPGA Design Engineer assessment.

The primary objective is to accept parallel 8-bit data, temporarily store it in a FIFO buffer, and then transmit it serially using a UART-like interface. The system integrates multiple digital design modules including a Packetizer FSM, a synchronous FIFO, a UART transmitter, and a baud rate generator. These modules are brought together in a top-level design that supports parameterizable baud rates and standard UART framing (start and stop bits, no parity).

This design not only demonstrates fundamental RTL development skills using Verilog HDL, but also emphasizes module-level interaction, FSM-based control flow, and testbench-driven validation using a 50 MHz system clock.

The project is completed with a comprehensive testbench to simulate real-time data transfer, including waveform analysis for transmitted data, FIFO status signals, and FSM transitions. Furthermore, the implementation is intended to be synthesized and tested using Xilinx Vivado on a Virtex 7 series FPGA, with detailed synthesis and timing reports to evaluate performance.

SYSTEM OVERVIEW & ASSUMPTIONS:

- **Modules Developed:**
    - sync_fifo: A synchronous FIFO buffer for temporary storage of 8-bit data.
    - packetizer_fsm: FSM controlling the reading of FIFO and initiating UART transmission.
    - baud_gen: Generates baud rate tick signals for UART timing.
    - uart_tx: UART transmitter module sending 10-bit packets (1 start, 8 data, 1 stop).
    - uart_packetizer_top: Top module interconnecting all submodules.

Name: Vakkanti Lakshmi Revanth Kumar
Ph. No: 6281974219
Email: revi4922251@gmail.com

❖ **Input/Output Ports:**

- Inputs: clk, rst, data_in [7:0], data_valid, tx_ready
- Outputs: serial_out, fifo_full, tx_busy

❖ **FPGA BOARD:**

Virtex 7 FPGA Board consist of 200MHz clock frequency, by using clock divided by four, I convert 200MHz clock frequency into 50MHz clock frequency for using internal modules.

Feature	Details
FPGA Family	Virtex-7
Part Number	XC7VX690TFFG1761-2
Logic Cells	~693,000
Slices	108,300
Block RAM	~52.9 Mb
DSP Slices	3,600
I/O Pins	1,000+
Transceivers	80 × 13.1 Gbps GTX
Package	FFG1761
Speed Grade	-2 (faster than -1)

- **High Logic Capacity**: Easily accommodates your FIFO, FSM, UART TX, and more.
- **Fast Speed Grade (-2)**: Supports high-frequency designs (good for UART timing).
- **Plenty of BRAM**: Can implement deeper FIFOs or buffers if needed.
- **Development-Friendly**: The VC709 board provides USB JTAG, DDR3, PCIe, and other interfaces, useful for larger embedded systems and debugging.

❖ **ASSUMPTIONS**

- FIFO is implemented synchronously (as per user preference).
- Baud rate is parameterizable (tested with 9600 bps).
- UART packet: 1 start bit (0) + 8 data bits + 1 stop bit (1).
- External module provides valid data with a `data_valid` signal.
- System clock is 50 MHz

Name: Vakkanti Lakshmi Revanth Kumar
Ph. No: 6281974219
Email: revi4922251@gmail.com

OBJECTIVE

Design and implement a UART Packetizer system that accepts parallel 8-bit data, buffers it using a FIFO, and transmits it serially via a UART interface using a finite state machine (FSM), with parameterizable baud rate and proper protocol framing.

SYNCHRONOUS FIFO

A synchronous FIFO (First-In, First-Out) is a memory buffer that temporarily stores data and uses a single clock for both writing and reading operations. Data is read in the same order it was written. In design, it stores 8-bit data and has 16 storage locations (depth).

Key Features:

- Width: 8 bits
- Depth: 16 entries
- Clock Domain: Single clock (synchronous)

How it works:

- **Write Operation:**
  - When wr_en is high and FIFO is not full (fifo_full = 0), data is written to the location pointed by wr_ptr.
  - wr_ptr is then incremented.
- **Read Operation:**
  - When rd_en is high and FIFO is not empty (fifo_empty = 0), data is read from the location pointed by rd_ptr.
  - rd_ptr is then incremented.
  - data_out_valid is asserted to indicate valid output.

Control Signals:

- fifo_full: High when FIFO has no space for new data.
- fifo_empty: High when there is no data to read.
- data_out_valid: Indicates that dout holds valid data after read.

Name: Vakkanti Lakshmi Revanth Kumar
Ph. No: 6281974219
Email: revi4922251@gmail.com

Rational for synchronous/asynchronous FIFO:

A synchronous FIFO is used because the entire UART packetizer system including input logic, FSM, and UART operates on a single system clock.

- Simple Design: Uses one clock for both read and write.
- No Clock Crossing: Safe and easy to use when both ends share the same clock.
- Ideal for small data buffering in your UART Packetizer, where the FSM and FIFO run at the same system clock.

When Asynchronous FIFO is Needed:

- If your system had two different clock domains like data producer and UART using different clocks, then an asynchronous FIFO would be necessary.

FSM STATE DIAGRAM AND STATE TRANSITIONS

The FSM (Finite State Machine) controls how data is read from the FIFO and sent over UART. It ensures proper handshaking and framing of the UART packet.

Purpose of the FSM

The FSM acts as the controller that manages the flow of data from the FIFO to the UART transmitter. Its job is to:

1. Check when data is available in the FIFO.
2. Wait until the UART is ready to send.
3. Read a data byte from the FIFO.
4. Start transmission over the UART line.
5. Wait until the UART finishes sending before going back to handle more data.

This ensures proper timing, synchronization, and efficient use of the UART, which sends data one bit at a time.

Name: Vakkanti Lakshmi Revanth Kumar
Ph. No: 6281974219
Email: revi4922251@gmail.com

State diagram:

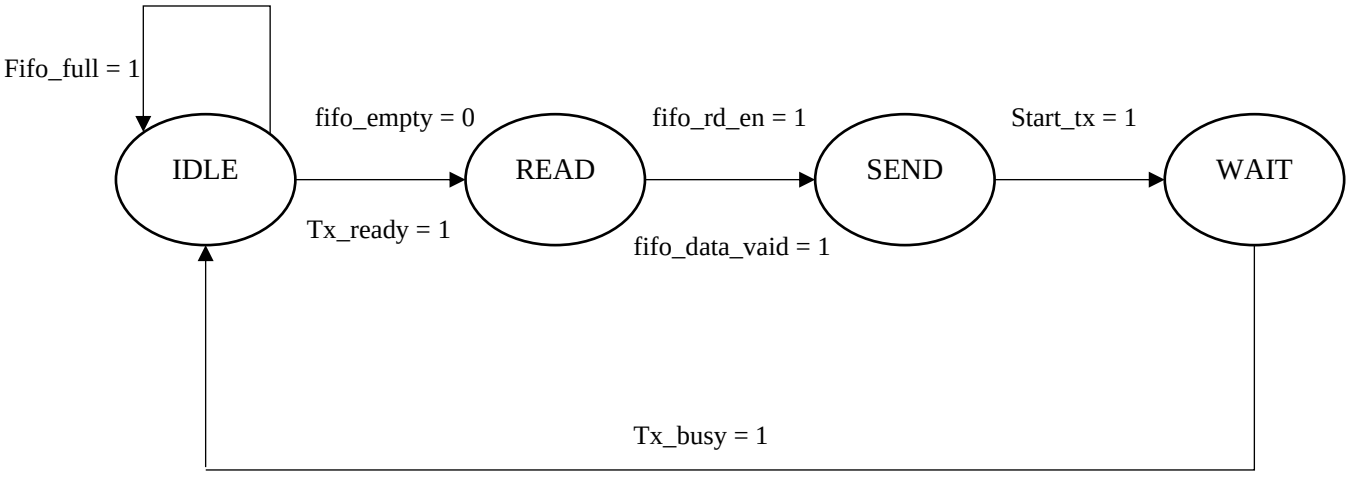

Fig: FSM State Diagram

FSM States and Transitions

- IDLE
  - Role: Waiting state.
  - Condition to leave this state:
    - FIFO must have data (fifo_empty == 0)
    - UART must be ready (tx_ready == 1)
  - Next State: READ
  - Why it's needed: Prevents unnecessary reads when UART isn't ready or FIFO is empty.
- READ
  - Role: Initiates a read from the FIFO by asserting fifo_rd_en = 1.
  - Condition to leave this state:
    - fifo_data_valid == 1, ensures the FIFO has placed valid data on the output
  - Actions:
    - Latches the data from FIFO into a temporary register (uart_data)
  - Next State: SEND
  - Why it's needed: Safely captures data from the FIFO before beginning transmission.

Name: Vakkanti Lakshmi Revanth Kumar
Ph. No: 6281974219
Email: revi4922251@gmail.com

❖ SEND

- Role: Signals the UART transmitter to begin sending the captured data.
- Action:
  - Asserts start_tx = 1 for one cycle.
- Next State: WAIT
- Why it's needed: Provides a clean trigger for the UART module to start transmission.

❖ WAIT

- Role: Waits for the UART to complete the current transmission.
- Condition to return to IDLE:
  - tx_busy == 0, it means UART has finished sending the last byte.
- Why it's needed: Prevents overlap of transmissions, ensures data is sent correctly before starting the next packet.

CONCLUSION

This project implements a UART packetizer system using Verilog, featuring a synchronous FIFO, baud rate generator, UART transmitter, and FSM controller. It accepts 8-bit parallel data, buffers it in the FIFO, and transmits it serially with UART framing (start/stop bits) at a configurable baud rate. The FSM handles data flow, ensuring proper handshaking with tx_ready and tx_busy signals.

The design was synthesized and simulated successfully on the Virtex-7 VC709 FPGA using Vivado. All functional requirements like FIFO operations, UART framing, and FSM transitions, were verified through simulation. The synchronous FIFO ensures stable operation at 50 MHz, and the modular design supports easy scalability and reuse in real-time communication systems.