# UART Packetizer – Project Summary Report

## 1. Objective

The purpose of this project is to design a UART Packetizer system using Verilog HDL and implement it on an FPGA. The system accepts 8-bit parallel input data, stores it in a FIFO buffer, and transmits it serially using a UART transmitter. The flow of data is controlled by a finite state machine (FSM), ensuring synchronized communication. The design is verified using simulation and synthesized for a Kintex-7 FPGA target device.

## 2. Design Architecture

The complete system consists of three main modules:  
  
- FIFO (First In First Out buffer): Stores incoming 8-bit data temporarily. It uses write and read pointers to manage storage. It outputs data when requested by the FSM and provides full and empty signals for flow control.  
  
- Packetizer FSM: A 4-state finite state machine that controls when data is read from the FIFO and when the UART transmission should begin. The FSM states include IDLE, READ, WAIT, and START\_TX.  
  
- UART Transmitter: Converts 8-bit parallel data into a 10-bit serial frame including 1 start bit, 8 data bits (LSB first), and 1 stop bit. It uses a configurable baud rate divider to time each bit.

## 3. Target FPGA and Constraints

FPGA Device: artix-7 xc7a35tcpg236-1  
Tool: AMD-Xilinx Vivado  
Clock Frequency: 50 MHz  
Constraint Applied:  
create\_clock -period 20.000 -name clk -waveform {0 10} [get\_ports clk]

## 4. Synthesis and Implementation Results

The design was synthesized and implemented successfully using Vivado. The following reports summarize the results:  
  
Timing Summary:  
- Worst Negative Slack (WNS): +16.17 ns  
- Hold Slack: Positive  
- Conclusion: The design meets all setup and hold timing requirements for 50 MHz operation.

Resource Utilization (Post-Synthesis):  
Resource | Used | Available | Utilization  
LUTs | 55 | 20,800 | ~0.26%  
Flip-Flops | 61 | 41,600 | ~0.15%  
IOBs | 13 | 200 | ~6.5%  
  
The design is compact and resource-efficient, making it suitable for larger systems or low-power designs.

## 5. FSM Encoding and Performance Improvement Suggestions

FSM Encoding:  
Vivado inferred one-hot encoding for the FSM, which is optimal for smaller state machines as it provides faster switching and reduced logic depth.  
  
Performance Improvement Techniques:  
- Pipelining: Add registers between long logic paths to reduce delay.  
- Retiming: Enable automatic register redistribution using Vivado options.  
- FSM Optimization: Experiment with binary or gray encoding for larger FSMs.  
- Timing Constraints: Apply tighter timing constraints and analyze worst paths.  
- Vivado Strategies: Use Performance\_ExtraTimingOpt in implementation settings.

## 6. Conclusion

The UART Packetizer project demonstrates the successful design, simulation, and synthesis of a serial communication system on FPGA. The modular design ensures reusability, the FSM provides control, and the UART module ensures standard data transmission. The implementation is area- and timing-efficient, verified through waveform inspection and Vivado reports. This project is ready for submission and future enhancements.