COMSATS UNIVERSITY
ISLAMABAD ATTOCK CAMPUS

DSD Project Report

NAME: ABDULLAH ASIF

REG NO: FA21-BCE-008

INSTRUCTOR: Sir. Qazi Zia

DATE: 31 – December – 2024

Design of Parallel to Serial Converter for Converting the Parallel Output of 8-Point Fast Fourier Transform to Serial Form Using Verilog HDL

Objective

The objective of this project is to design and implement a **Parallel to Serial Converter (PISO)** to convert the parallel output of an 8-point Fast Fourier Transform (FFT) to serial form using **Verilog HDL**. The primary goal is to facilitate efficient data serialization for further processing or transmission. The design was simulated and thoroughly verified for implementation on an FPGA board (**SP601**), ensuring practical applicability and reliability.

Design Description

- **Parallel to Serial Converter (PISO) Module**

The **Parallel to Serial Converter (PISO)** was implemented with the following features and design considerations to achieve high performance and robustness:

Inputs:

- **clk**: Clock signal to synchronize operations and ensure proper timing.
- **reset**: Active-high signal to clear all internal states and outputs, providing a reliable initialization mechanism.
- **parallel_in**: 8-bit parallel input simulating the FFT output, representing the input data to be serialized.
- **load**: Load signal to latch the parallel input data into the shift register for subsequent serialization.

Output:

- **serial_out**: Produces serialized output, transmitting one bit per clock cycle in a sequential manner.

Verilog Code:

```
module piso (
    input wire clk,           // Clock signal
    input wire reset,         // Reset signal (active high)
    input wire [7:0] parallel_in, // 8-bit parallel input (FFT output)
    input wire load,          // Load signal to latch parallel data
    output reg serial_out     // Serial output
);
    reg [7:0] shift_reg;      // Shift register to hold parallel data
    reg [2:0] count;          // 3-bit counter for 8 cycles

    always @(posedge clk or posedge reset) begin
        if (reset) begin
```

```
            shift_reg <= 8'b0;      // Clear shift register
            serial_out <= 1'b0;     // Reset serial output
            count <= 3'b0;          // Reset counter
        end else if (load) begin
            shift_reg <= parallel_in; // Load parallel data into shift reg
            count <= 3'b0;            // Reset counter for shifting
        end else if (count < 8) begin
            serial_out <= shift_reg[7];   // Output the MSB first
            shift_reg <= {shift_reg[6:0], 1'b0}; // Shift left
            count <= count + 1;          // Increment counter
        end
    end
endmodule
```

Testbench Implementation

A testbench was developed to rigorously verify the functionality of the **PISO** module. The testbench validates the following key aspects of the design:

1. **Loading Parallel Data**: Ensures parallel_in is latched into the shift register on receiving the load signal.
2. **Serial Transmission**: Verifies correct bit-by-bit transmission on the serial_out line, ensuring proper serialization of data.
3. **Reset Functionality**: Checks that the reset signal clears all internal states during operation, allowing for a clean restart.

Testbench Code:

```
`timescale 1ns / 1ps
module piso_test;

    // Inputs
    reg clk;
    reg reset;
    reg [7:0] parallel_in;
    reg load;

    // Outputs
    wire serial_out;

    // Instantiate the Unit Under Test (UUT)
    piso uut (
        .clk(clk),
        .reset(reset),
        .parallel_in(parallel_in),
        .load(load),
        .serial_out(serial_out)
    );

    // Clock generation
    always #5 clk = ~clk; // 10ns clock period
```

```
    initial begin
        // Initialize inputs
        clk = 0;
        reset = 0;
        parallel_in = 8'b00000000;
        load = 0;

        // Apply reset
        #10 reset = 1; // Activate reset
        #10 reset = 0; // Deactivate reset

        // Test Case 1: Load and serialize data
        parallel_in = 8'b10101010; // Input data
        #10 load = 1; // Load the data
        #10 load = 0; // Start serialization
        #100; // Wait for all bits to shift out

        // Test Case 2: Load a different value
        parallel_in = 8'b11001100;
        #10 load = 1; // Load new data
        #10 load = 0; // Start serialization
        #100; // Wait for all bits to shift out

        // Test Case 3: Reset during operation
        parallel_in = 8'b11110000;
        #10 load = 1;
        #10 load = 0;
        #30 reset = 1; // Reset during shifting
        #10 reset = 0; // Resume operation

        // End of simulation
        #50 $stop;
    end

    // Monitor outputs
    initial begin
        $monitor("Time: %0t | Serial Out: %h", $time, serial_out);
    end

endmodule
```

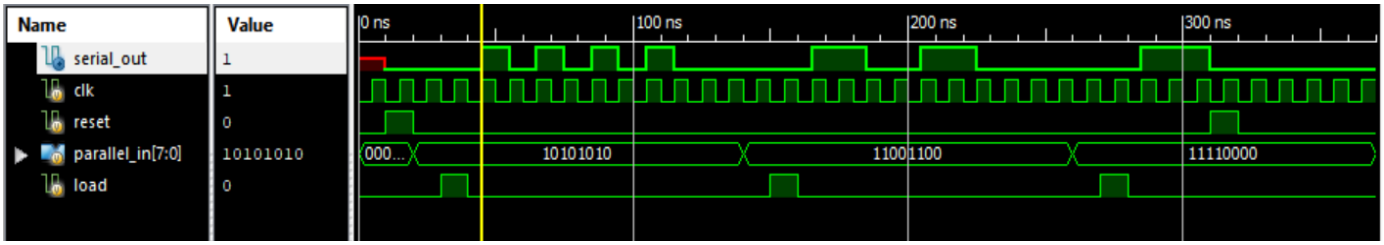
Results and Verification

- **Simulation Results**

  - **Data Serialization**: The serial_out line transmitted data bit-by-bit in each clock cycle after receiving the load signal, demonstrating correct functionality.
  - **Reset Functionality**: The reset signal cleared all internal states, including the shift register and output, as expected, ensuring a reliable and predictable reset mechanism.
  - **New Data Load**: The design successfully loaded new parallel data for serialization after the previous transmission, highlighting its capability to handle consecutive data loads.

- **Simulation Waveform Analysis**

The simulation waveform demonstrated:



5. Conclusion

The Parallel to Serial Converter (PISO) was successfully designed, implemented, and simulated using Verilog HDL. The design was verified for correct functionality, ensuring accurate serialization of the 8-bit parallel input. This robust design is now ready for FPGA implementation on the SP601 board, providing a practical solution for serializing FFT output data. Furthermore, the project serves as a foundation for future enhancements, such as increasing the data width or integrating additional functionality to support advanced applications.