module siso(

input wire clk,

input wire reset,

input wire serial\_in,

output wire serial\_out

);

reg [3:0] shift\_reg;

always @(posedge clk or posedge reset) begin

if (reset) begin

shift\_reg <= 4'b0000;

end else begin

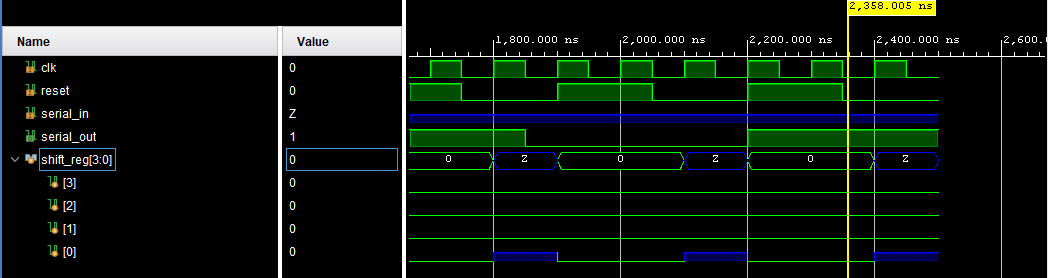
shift\_reg <= {shift\_reg[2:0], serial\_in};

end

end

assign serial\_out = shift\_reg[3];

endmodule



Test Bench:

module tb\_siso\_shift\_register;

// Testbench signals

reg clk; // Clock signal

reg rst; // Reset signal

reg serial\_in; // Serial input

wire serial\_out; // Serial output

// Instantiate the SISO Shift Register

siso\_shift\_register uut (

.clk(clk),

.rst(rst),

.serial\_in(serial\_in),

.serial\_out(serial\_out)

);

// Clock generation

initial begin

clk = 0;

forever #5 clk = ~clk; // 10 time units clock period

end

// Test sequence

initial begin

// Initialize signals

rst = 1; // Activate reset

serial\_in = 0;

#10; // Wait for some time

rst = 0; // Deactivate reset

// Apply serial input bits

serial\_in = 1; #10; // Shift in 1

serial\_in = 0; #10; // Shift in 0

serial\_in = 1; #10; // Shift in 1

serial\_in = 0; #10; // Shift in 0

serial\_in = 1; #10; // Shift in 1

serial\_in = 1; #10; // Shift in 1

serial\_in = 0; #10; // Shift in 0

serial\_in = 1; #10; // Shift in 1

// Final state check

#10; // Wait for one more clock cycle

$stop; // End simulation

end

endmodule