📘 **Day 22 Topic: Verilog codes for SISO Register Using Cadence (Xcelium)**

SISO (Serial-In Serial-Out) Register

A SISO register is a type of shift register where data enters and leaves serially, i.e., one bit at a time.

It is made up of a series of flip-flops connected in a chain, and all flip-flops share the same clock signal.
On each clock pulse, the data in each flip-flop shifts to the next one, and finally, the last flip-flop gives the serial output.

So, data moves through the register bit by bit with each clock pulse.

Key Points

- Serial Input: Data enters one bit at a time.
- Serial Output: Data exits one bit at a time.
- Common Clock: Controls the shifting of bits.
- Storage Capacity: Equal to the number of flip-flops.

Example

If the input bits are 1, 0, 1, 1, each clock pulse moves the bits through the flip-flops until they appear at the output.

🔹 **1.1) Design Code**

siso.v

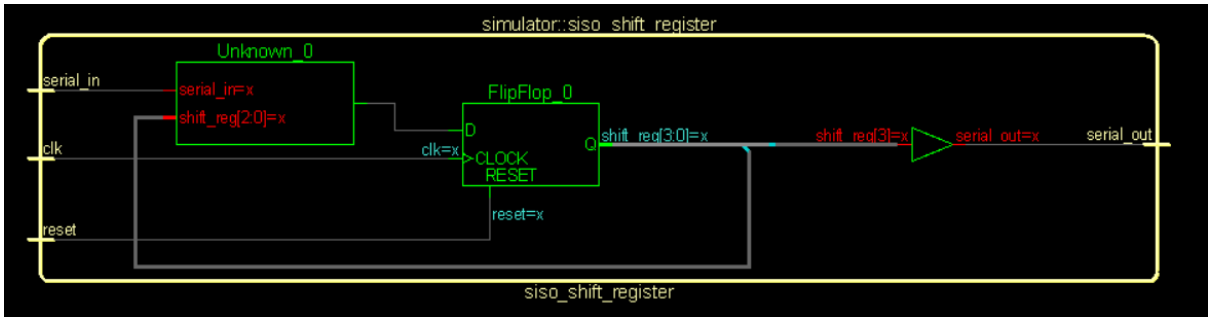
```
// Module: Serial In Serial Out (SISO) Shift Register

module siso_shift_register (
    input clk,          // clock signal
    input reset,        // asynchronous reset
    input serial_in,    // serial data input
    output serial_out   // serial data output
);
    reg [3:0] shift_reg;  // 4-bit shift register storage

    // On each clock, shift data to the right
    always @(posedge clk or posedge reset) begin
        if (reset)
            shift_reg <= 4'b0000;                  // clear register on reset
        else
            shift_reg <= {shift_reg[2:0], serial_in}; // shift left and insert new bit
    end

    assign serial_out = shift_reg[3]; // MSB as serial output
endmodule
```

- **1.2) Test Bench Code**

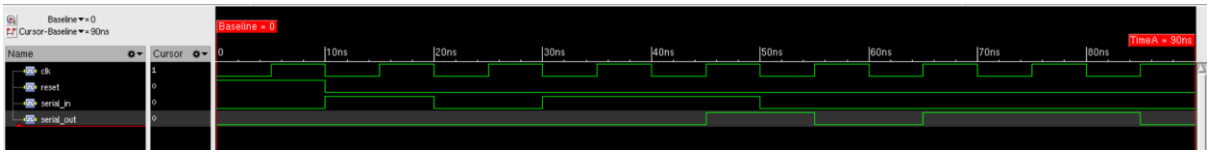
tb.v

```
// Testbench: SISO Shift Register

module tb_siso_shift_register;
    reg clk;
    reg reset;
    reg serial_in;
    wire serial_out;

    // Instantiate DUT (Design Under Test)
    siso_shift_register uut (
        .clk(clk),
        .reset(reset),
        .serial_in(serial_in),
        .serial_out(serial_out)
    );

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

    initial begin
        // Initialize signals
        clk = 0;
        reset = 1;
        serial_in = 0;
        #10 reset = 0;

        // 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 = 1; #10;   // shift in 1
        serial_in = 0; #10;   // shift in 0

        // Wait and finish simulation
        #30;
        $finish;
    end
endmodule
```

- **1.3) Schematic**



- **1.4) Wave Forms**



Applications

- **Used in serial communication (sending or receiving data bit by bit).**
- **Acts as a time delay element.**
- **Used for temporary data storage and data shifting.**