DSD F14

Homework #4

1. Write a Verilog model for a 74HC192 synchronous 4-bit up/down counter. Don't include any timing data. Make sure you include in the sensitivity list (Down, Up, Clr, Load)

```
module counter_74HC192 (
    input wire Clr,      // Clear the counter
    input wire PL,       // Async parallel load
    input wire Up,       // Count up
    input wire Down,     // Count down
    input wire [3:0] P,  // Data input for loading preset
    input wire Clk,      // Clock input
    output reg [3:0] Q   // 4-bit counter output
);

// Always block, sensitive to Clk, Clr, Load, Up, Down
always @(posedge Clk or posedge Clr or posedge Load) begin
    if (Clr) begin
        Q <= 4'b0000;  // Clear the counter
    end
    else if (Load) begin
        Q <= D;  // Load the counter with input value
    end
    else if (Up && !Down) begin
        Q <= Q + 1;  // Increment counter
    end
    else if (Down && !Up) begin
        Q <= Q - 1;  // Decrement counter
    end
end

endmodule
```

2. Write Verilog code for a module describing a 8-bit shift register that shifts bits in from input SI and out an output SO. Also include control signals Clk and EN (and enable active Hi). Thus: You should have 3 inputs SI, Clk, EN and 1 out SO: all these are single bit signals. The shifting causes

1 bit to go in and 1 bit to go out and remembers the other 7 bits, but the bit just shifted in will not get shifted out for 8 more clock edges. Use a rising edge clock.

```
module shift_register_8bit (
    input wire SI,    // Serial input
    input wire Clk,   // Clock
    input wire EN,    // Enable (active high)
    output reg SO     // Serial output
);

    reg [7:0] shift_reg;

    always @(posedge Clk) begin
        if (EN) begin
            SO <= shift_reg[7];               // Shift out MSB
            shift_reg <= {shift_reg[6:0], SI};  // Shift left, input SI
        end
    end

endmodule
```

3. Write a Verilog module for a SR Latch holding 1 bit. Do it 2 ways:

    a. Use an Boolean equation

```
module sr_latch_boolean (
    input wire Si,    // Set input
    input wire Reset,// Reset input
    output reg Q,    // Output
    output reg Qn    // Inverted output
);

    always @(*) begin
        if (Si && !Reset) begin
            Q <= 1;
            Qn <= 0;
        end
        else if (!S && Reset) begin
            Q <= 0;
            Qn <= 1;
        end
```

```
    end

endmodule
```

b. Use a gate level description

```
module sr_latch_gate_level (
    input wire S,   // Set
    input wire Reset,   // Reset
    output wire Q,  // Output
    output wire Qn  // Inverted output
);

    wire nand0_out, nand1_out;


    nand (nand0_out, S, Qn);
    nand (nand1_out, Reset, Q);

    assign Q = nand0_out;
    assign Qn = nand1_out;

endmodule
```