DAY 42 - 111 DAYS VERIFICATION CHALLENGE

Topic: Advanced verilog codes

Skill: Verilog, RTL design

DAY 42 CHALLENGE:

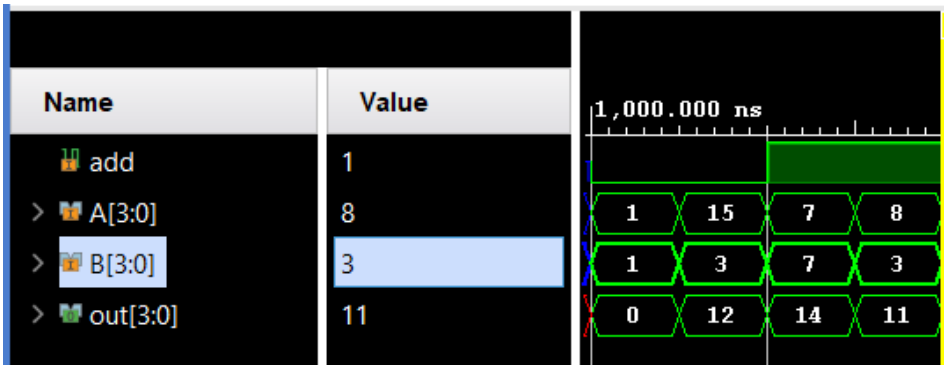
1. Design the following in Verilog:

A circuit that can be used as an inverter/buffer depending on the input signal INV. If INV=1, the circuit acts as an inverter. If INV=0, the circuit acts as a Buffer.

```
module inverter_buffer(
    input data_in,
    input inv,
    output reg data_out
    );
always @(*)
    begin
        if(inv)
            begin
                data_out = ~data_in ;
            end
        else
            begin
                data_out = data_in ;
            end
    end
endmodule
```

2. Design a module 4bit_adder_subtractor_dut which has a function 4b_add_sub which takes an input signal add. If add=1, the function performs addition of two 4-bit inputs 'A' & 'B', else it performs subtraction of 'A' & 'B'

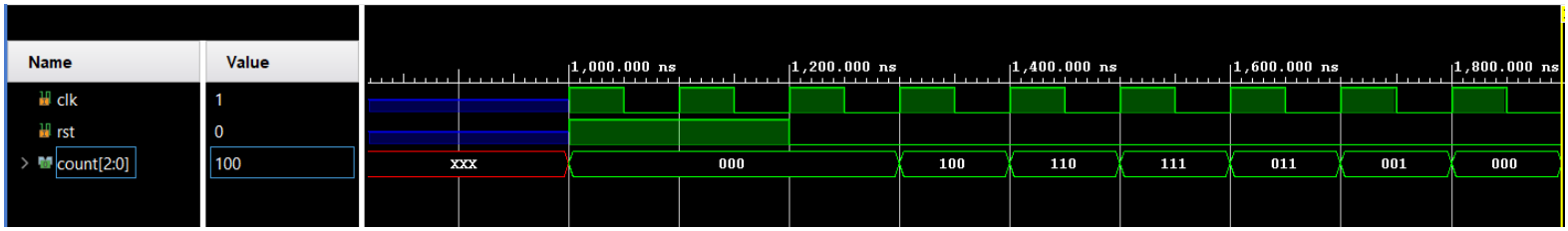
```
module add_sub(
    input add,
    input [3:0] A,
    input [3:0] B,
    output reg[4:0] out
    );
always @(*)
    begin
        if(add)
            out = A + B ;
        else
            out = A - B ;
    end
endmodule
```



3. Design following using Verilog:

I. 3-bit Johnson Counter

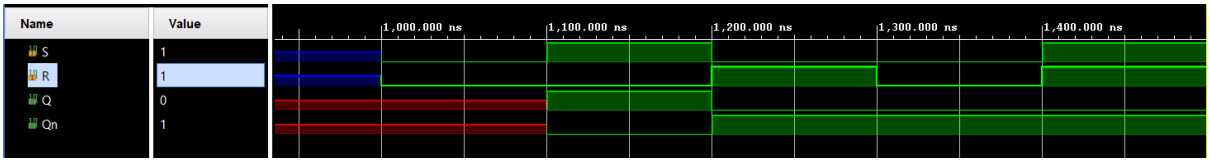
```
module three_bit_johnson_cnt(
    input clk,
    input rst,
    output reg [2:0] count
    );

always @(posedge clk or posedge rst)
    begin
        if (rst)
            count <= 3'b000 ;
        else
            count <= {~count[0],count[2:1]};
    end
endmodule
```



II. SR latch

```
module SR_latch(
    input wire S, R,
    output reg Q, Qn
```

```
);
always @(*) begin
    if (S == 1 && R == 0) begin
        Q <= 1'b1;
        Qn <= 1'b0;
    end else if (S == 0 && R == 1) begin
        Q <= 1'b0;
        Qn <= 1'b1;
    end else if (S == 0 && R == 0) begin
        // Maintain previous state; no changes to Q and Qn
    end else if (S == 1 && R == 1) begin
        // This is typically an invalid state for an SR latch
        //  Outputs  remain  unchanged  or  follow  a  specific
behavior based on design choice
    end
end
endmodule
```



4. Design an FSM in Verilog with below states:

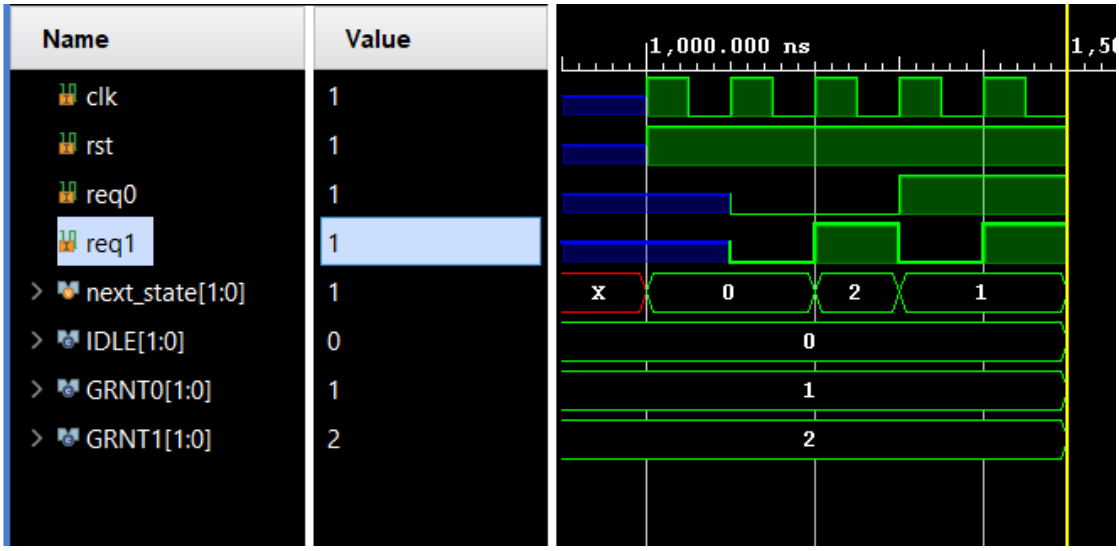
I. **IDLE - This is the default state on reset.**
IDLE state can transition to 2 states GRNTO & GRNT1:
GRNT0- if req0=1 GRNT1- if req1=1

II. **GRNT 0 - This state is achieved from IDLE state when req0=1**
If req0 remains '1', the state GRNTO state remains unchanged.
If req0 = 0, then GRNTO transitions to IDLE state.

III. **GRNT 1 - This state is achieved from IDLE state when req1=1**
If req1 remains '1', the state GRNT1 state remains unchanged.
If req1 =0, then GRNT1 transitions to IDLE state.

```
module FSM (
    input wire clk,      // Clock signal
    input wire rst,      // Reset signal
    input wire req0,     // Request signal 0
    input wire req1,     // Request signal 1
    output reg [1:0] state // Output state (encoded as 2 bits)
);

// State encoding
localparam IDLE  = 2'b00;
localparam GRNT0 = 2'b01;
```

```
localparam GRNT1 = 2'b10;

// State register
reg [1:0] next_state;

// Sequential logic for state transition
always @(posedge clk or posedge rst) begin
    if (rst) begin
        state <= IDLE;
    end else begin
        state <= next_state;
    end
end

// Combinational logic for next state determination
always @(*) begin
    case (state)
        IDLE: begin
            if (req0) begin
                next_state = GRNT0;
            end else if (req1) begin
                next_state = GRNT1;
            end else begin
                next_state = IDLE;
            end
        end
        GRNT0: begin
            if (req0) begin
                next_state = GRNT0; // Remain in GRNT0 if req0
is high
            end else begin
                next_state = IDLE; // Transition to IDLE if req0
goes low
            end
        end
        GRNT1: begin
            if (req1) begin
                next_state = GRNT1; // Remain in GRNT1 if req1
is high
            end else begin
                next_state = IDLE; // Transition to IDLE if req1
goes low
            end
        end
        default: next_state = IDLE; // Default to IDLE for safety
    endcase
end
endmodule
```


Name
Value
clk
1
rst
1
req0
1
req1
1
next_state[1:0]
1
IDLE[1:0]
0
GRNT0[1:0]
1
GRNT1[1:0]
2
1,000.000 ns
x
0
2
1
0
1
2