


# CS224\_2021\_quiz2\_A

...

Points: 10/20

1. Design 4-bit register. Register is loaded only when the load signal (ld) is high. Data to load comes on input wires 'din'. If 'reset' is high then register is reset synchronous with the clock (clk). The register has an input 'incr' for increment function. The value is incremented when incr is high. Write the Verilog code and paste it in the space below. 

(6/6 Points)

```
module q1(  
    input [3:0] din,  
  
    input incr,  
    output reg [3:0] dout,  
    input load,  
    input reset,  
    input clk  
);  
  
always @(posedge clk)  
begin  
    if(reset)  
        dout = 3'b000;  
    else if(load)  
        dout = din;  
    else if (incr)  
        dout= dout+1;  
end  
  
endmodule
```

2. For the register with increment feature write a testbench and paste output of monitor command below.  
testbench should have following sequence of inputs followed by stop command to stop simulation.

Intiallise all variable to zero.

```
#10 reset = 1;  
#10 load = 1;  
#10 din = 4;  
#10 reset = 0;  
#10 din = 6;  
#10 din = 3;  
#10 incr = 1;  
#10 load = 0;  
#10 din = 8;  
#60;  
$stop;  
(-/4 Points)
```

Enter your answer

3. Design 4-bit register. Register is loaded only when the load signal (ld) is high. Data to load comes on input wires 'din'. If 'reset' is high then register is reset synchronous with the clock (clk).  
The register has an input 'lin' (left-in) as an external 1-bit input and 'sh' (shift) as 1-bit control input.  
When shift is enabled then the contents of register are shifted to the right and external input 'lin' is pushed inside the register. Loading register gets priority over shifting operation.  
Write the Verilog code and paste it in the space below.  
(4/6 Points)

```

`timescale 1ns / 1ps

module q2(
    input load,
    input [3:0] din,
    input lin,
    input sh,
    output reg [3:0] dout,
    input clk,
    input reset
);

always @(posedge clk)
    begin
        if(load)
            dout = din;
        if(reset)
            dout = 4'b0000;
        if(sh)
            begin
                dout[3] = lin;
                dout[2] = din[3];
                dout[1] = din[2];
                dout[0] = din[1];
            end
    end

endmodule

```

☞ "Load operation has higher priority than shift operation"

4. For the register with shift right feature write a testbench and paste output of monitor command below.

testbench should have following sequence of inputs =

Intialise all variable to zero.

```
#10 reset = 1;
#10 load = 1;
#10 din = 4; sh=1; lin=1;
#10 reset = 0;
#10 din = 6;
#10 din = 3;
#10 sh = 1; lin=1;
#10 ld=0;
#10 din = 8; lin=0;
#60;
$stop;
(0/4 Points)
```

```
time :      0 din= 0000, dout = xxxx ,sh = 0, lin = 0, reset = 0, ld = 0
time :     110 din= 0000, dout = xxxx ,sh = 0, lin = 0, reset = 1, ld = 0
time :     115 din= 0000, dout = 0000 ,sh = 0, lin = 0, reset = 1, ld = 0
time :     120 din= 0000, dout = 0000 ,sh = 0, lin = 0, reset = 1, ld = 1
time :     130 din= 0100, dout = 0000 ,sh = 1, lin = 1, reset = 1, ld = 1
time :     135 din= 0100, dout = 1010 ,sh = 1, lin = 1, reset = 1, ld = 1
time :     140 din= 0100, dout = 1010 ,sh = 1, lin = 1, reset = 0, ld = 1
time :     150 din= 0110, dout = 1010 ,sh = 1, lin = 1, reset = 0, ld = 1
time :     155 din= 0110, dout = 1011 ,sh = 1, lin = 1, reset = 0, ld = 1
time :     160 din= 0011, dout = 1011 ,sh = 1, lin = 1, reset = 0, ld = 1
time :     165 din= 0011, dout = 1001 ,sh = 1, lin = 1, reset = 0, ld = 1
time :     180 din= 0011, dout = 1001 ,sh = 1, lin = 1, reset = 0, ld = 0
time :     190 din= 1000, dout = 1001 ,sh = 1, lin = 0, reset = 0, ld = 0
time :     195 din= 1000, dout = 0100 ,sh = 1, lin = 0, reset = 0, ld = 0
```

🗨 "Incorrect output"

