LAB 9: Flip Flops (JK, D, T)

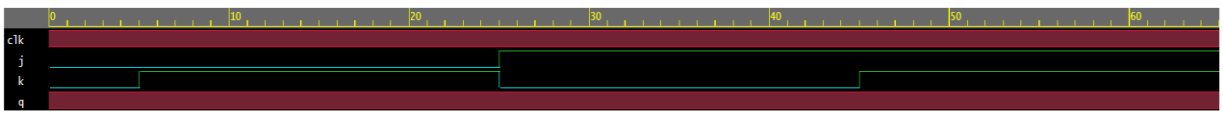
Question 1: Write a Verilog code to implement J-K flip flop and validate the code via a suitable Test bench code.

design.sv

```
module jk_ff ( input j, input k, input clk, output q);
  reg q;
  always @ (posedge clk)
    case ({j,k})
      2'b00 :  q <= q;
      2'b01 :  q <= 0;
      2'b10 :  q <= 1;
      2'b11 :  q <= ~q;
    endcase
endmodule
```

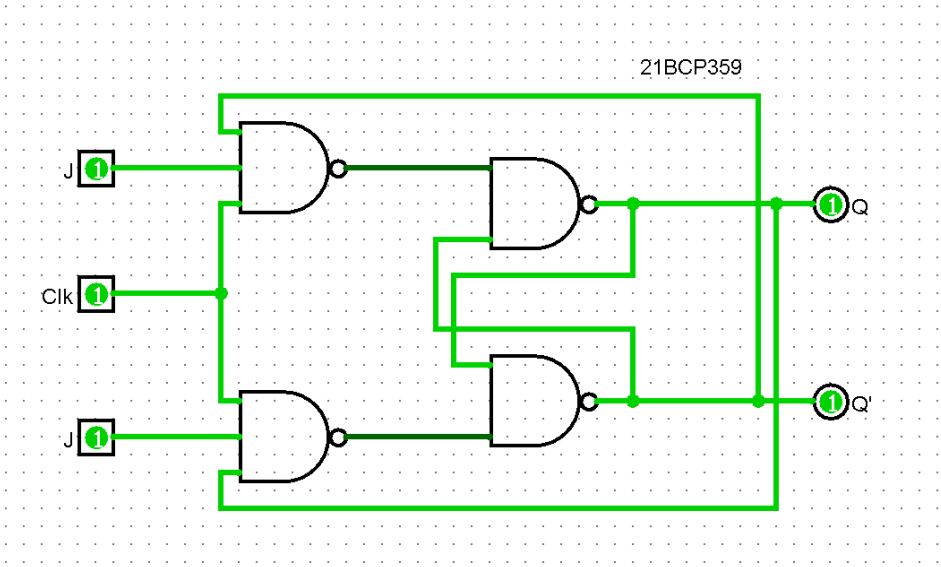
testbench.sv

```
module tb_jk;
  reg j;
  reg k;
  reg clk;
  always #5 clk = ~clk;
  jk_ff jk0 ( .j(j), .k(k), .clk(clk), .q(q));

  initial
    begin
      j <= 0;
      k <= 0; #5

      j <= 0;
      k <= 1; #20

      j <= 1;
      k <= 0; #20

      j <= 1;
      k <= 1; #20
      $finish;
    end
  initial
    begin
      $dumpfile("dump.vcd");
      $dumpvars(1);
    end

  initial
    $monitor ("j=%0d k=%0d q=%0d", j, k, q);
endmodule
```




```
[2022-11-14 09:32:17 EST] iverilog '-Wall' design.sv testbench.sv  && unbuffer vvp a.out
testbench.sv:6: warning: implicit definition of wire logic tb_jk.q.
VCD info: dumpfile dump.vcd opened for output.
j=0 k=0 q=x
j=0 k=1 q=x
j=1 k=0 q=x
j=1 k=1 q=x
Finding VCD file...
./dump.vcd
[2022-11-14 09:32:17 EST] Opening EPWave...
Done
```

Question 2: Design a J-K flip flop in Logisim and validate the circuit.



Clock	J	K	Q_{n+1}	State
0	x	x	Q_n	
1	0	0	Q_n	Hold
1	0	1	0	Reset
1	1	1	1	Set
1	1	1	$\overline{Q}_n$	Toggle

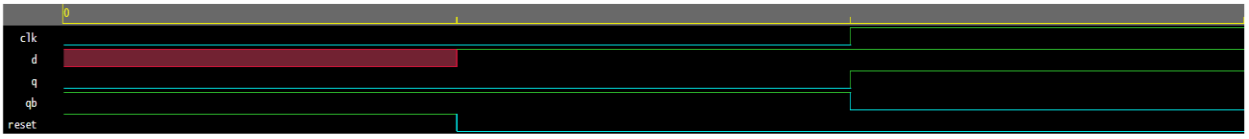
Question 3: Write a Verilog code to implement D flip flop and validate the code via a suitable Test bench code.

design.sv

```
// Design
// D flip-flop
module dff (clk, reset,
  d, q, qb);
  input      clk;
  input      reset;
  input      d;
  output     q;
  output     qb;

  reg        q;

  assign qb = ~q;

  always @(posedge clk or posedge reset)
  begin
    if (reset) begin
      // Asynchronous reset when reset goes high
      q <= 1'b0;
    end else begin
      // Assign D to Q on positive clock edge
      q <= d;
    end
  end
endmodule
```

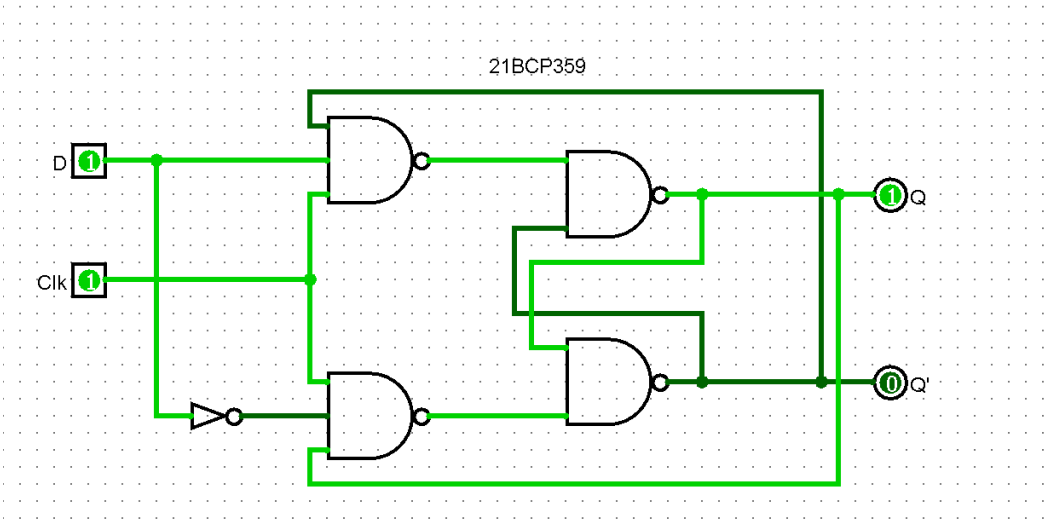
testbench.sv

```
// Testbench
module test;

  reg clk;
  reg reset;
  reg d;
  wire q;
  wire qb;

  // Instantiate design under test
  dff DFF(.clk(clk), .reset(reset),
          .d(d), .q(q), .qb(qb));

  initial begin
    // Dump waves
    $dumpfile("dump.vcd");
    $dumpvars(1);

    $display("Reset flop.");
    clk = 0;
    reset = 1;
    d = 1'bx;
    display;

    $display("Release reset.");
    d = 1;
    reset = 0;
    display;

    $display("Toggle clk.");
    clk = 1;
    display;
  end

  task display;
    #1 $display("d:%0h, q:%0h, qb:%0h",
      d, q, qb);
  endtask

endmodule
```




```
[2022-11-14 09:37:27 EST] iverilog '-Wall' design.sv testbench.sv  && unbuffer vvp a.out
VCD info: dumpfile dump.vcd opened for output.
Reset flop.
d:x, q:0, qb:1
Release reset.
d:1, q:0, qb:1
Toggle clk.
d:1, q:1, qb:0
Finding VCD file...
./dump.vcd
[2022-11-14 09:37:28 EST] Opening EPWave...
Done
```

Question 4: Design a D flip flop in Logisim and validate the circuit.



S	R	Q	Q`
0	0	0	1
0	1	0	1
1	0	1	0
1	1	∞	∞

Question 5: Write a Verilog code to implement T flip flop and validate the code via a suitable Test bench code.

testbench.sv

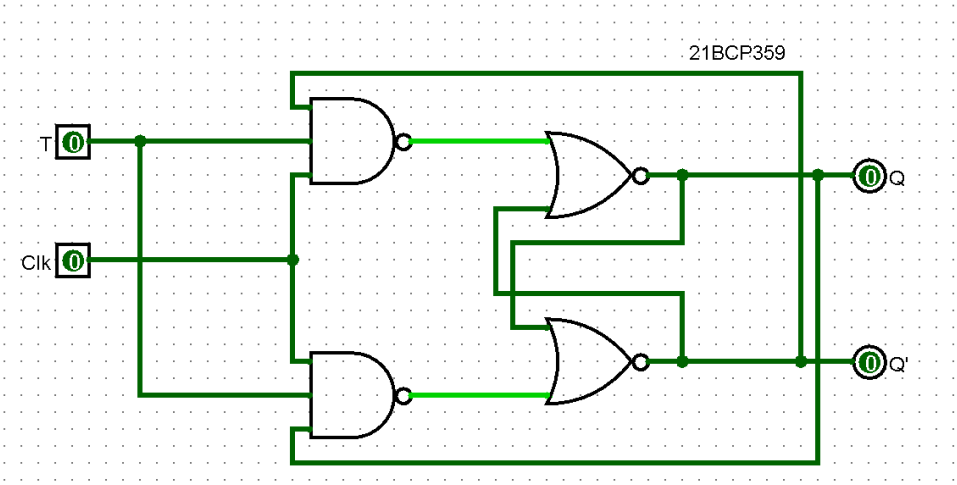
```
module tb;
  reg clk;
  reg rstn;
  reg t;

  tff u0 (  .clk(clk),
            .rstn(rstn),
            .t(t),
           .q(q));

  always #5 clk = ~clk;

  initial
    begin
    {rstn, clk, t} <= 0;

    $monitor ("T=%0t rstn=%0b t=%0d q=%0d", $time, rstn, t, q);
    repeat(2) @(posedge clk);
    rstn <= 1;
    end
  initial
    begin
      $dumpfile("dump.vcd");
      $dumpvars(1);
  end
endmodule
```

design.sv

```
module tff ( input clk, input rstn, input t, output reg q);

  always @ (posedge clk) begin
    if (!rstn)
      q <= 0;
    else
        if (t)
            q <= ~q;
        else
            q <= q;
  end
endmodule
```

```
[2022-11-15 00:51:05 EST] iverilog '-Wall' design.sv testbench.sv  && unbuffer vvp a.out
testbench.sv:9: warning: implicit definition of wire logic tb.q.
VCD info: dumpfile dump.vcd opened for output.
T=0 rstn=0 t=0 q=x
T=5 rstn=0 t=0 q=0
T=15 rstn=1 t=0 q=0
```

Question 6: Design a T flip flop in Logisim and validate the circuit.



Truth table			
CLK	T	Q_{next}	Comment
Rising edge	0	Q	Hold state
Falling edge	0	Q	Hold state
Rising edge	1	$\overline{Q}$	Toggle
Falling edge	1	Q	No change

Q_{next} - "after the clock transition" output
Q - the current output