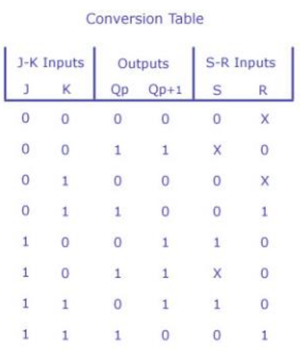
Lab 10: Flip Flops Conversion

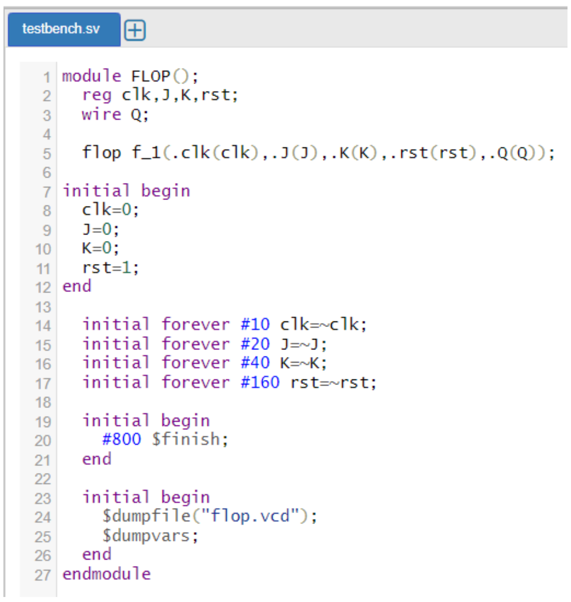
Question 1: Modify the S-R flip flop created in last lab into JK flip flop. Verify the functionality of J-K flip flop using suitable Testbench.



Conversion Table

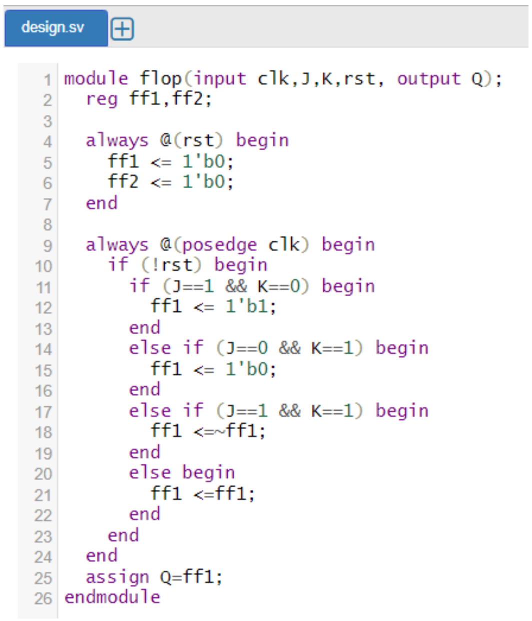
J-K Inputs		Outputs		S-R Inputs	
J	K	Qp	Qp+1	S	R
0	0	0	0	0	X
0	0	1	1	X	0
0	1	0	0	0	X
0	1	1	0	0	1
1	0	0	1	1	0
1	0	1	1	X	0
1	1	0	1	1	0
1	1	1	0	0	1

Conversion Expression: S = J'Q and R = KQ



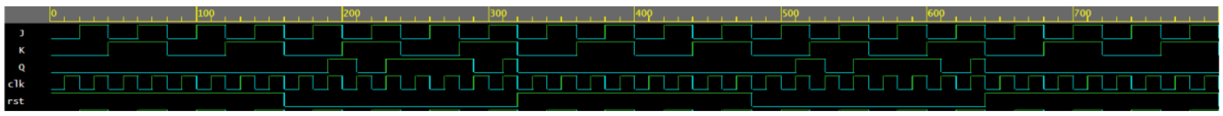
testbench.sv

```verilog
module FLOP();
  reg clk,J,K,rst;
  wire Q;

  flop f_1(.clk(clk),.J(J),.K(K),.rst(rst),.Q(Q));

initial begin
  clk=0;
  J=0;
  K=0;
  rst=1;
end

  initial forever #10 clk=~clk;
  initial forever #20 J=~J;
  initial forever #40 K=~K;
  initial forever #160 rst=~rst;

  initial begin
    #800 $finish;
  end

  initial begin
    $dumpfile("flop.vcd");
    $dumpvars;
  end
endmodule
```

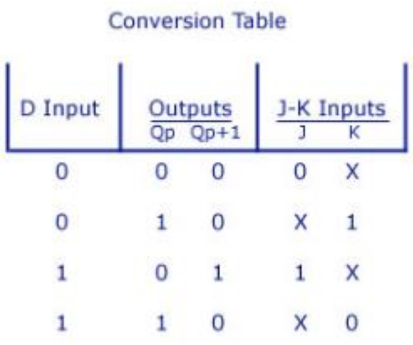


design.sv

```verilog
module flop(input clk,J,K,rst, output Q);
  reg ff1,ff2;

  always @(rst) begin
    ff1 <= 1'b0;
    ff2 <= 1'b0;
  end

  always @(posedge clk) begin
    if (!rst) begin
      if (J==1 && K==0) begin
        ff1 <= 1'b1;
      end
      else if (J==0 && K==1) begin
        ff1 <= 1'b0;
      end
      else if (J==1 && K==1) begin
        ff1 <=~ff1;
      end
      else begin
        ff1 <=ff1;
      end
    end
  end
  assign Q=ff1;
endmodule
```



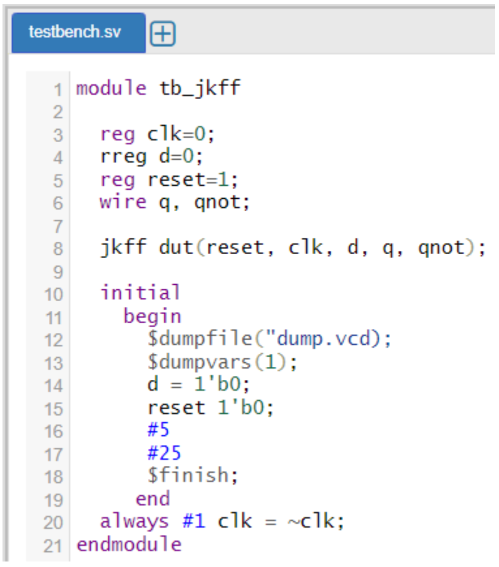
Question 2: Develop the behavioural Verilog module of D flip-flop, converting it from a J-K flip flop. You may utilize if-else statements to develop your Verilog code. Verify the functionality via a suitable test bench code.



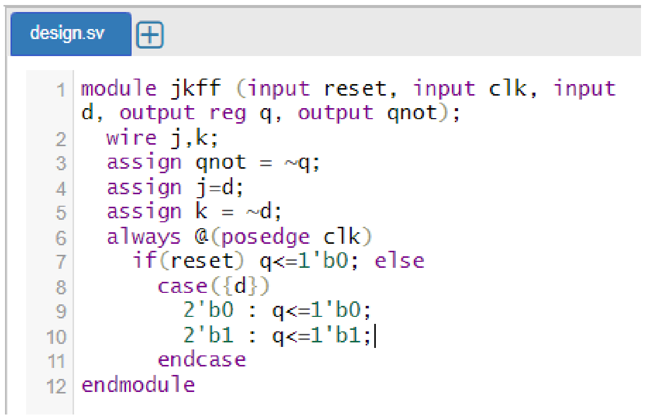
Conversion Table

D Input	Outputs		J-K Inputs	
	Qp	Qp+1	J	K
0	0	0	0	X
0	1	0	X	1
1	0	1	1	X
1	1	0	X	0

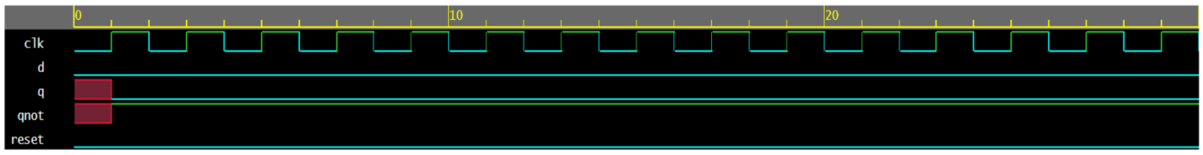
Conversion Expression: J = D and K = D’



```
module tb_jkff

  reg clk=0;
  rreg d=0;
  reg reset=1;
  wire q, qnot;

  jkff dut(reset, clk, d, q, qnot);

  initial
    begin
      $dumpfile("dump.vcd);
      $dumpvars(1);
      d = 1'b0;
      reset 1'b0;
      #5
      #25
      $finish;
     end
  always #1 clk = ~clk;
endmodule
```

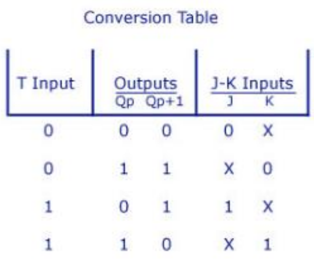


```
module jkff (input reset, input clk, input
d, output reg q, output qnot);
  wire j,k;
  assign qnot = ~q;
  assign j=d;
  assign k = ~d;
  always @(posedge clk)
    if(reset) q<=1'b0; else
      case({d})
        2'b0 : q<=1'b0;
        2'b1 : q<=1'b1;
      endcase
endmodule
```



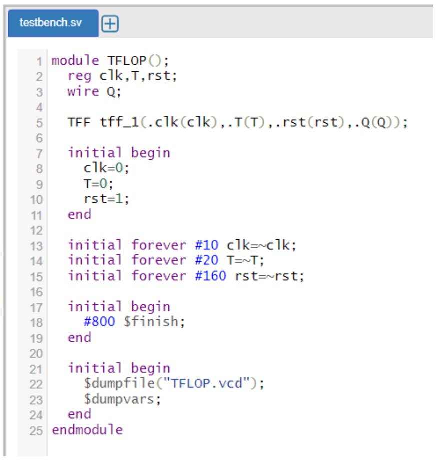
Question 3: Develop a behavioural Verilog module for JK flip flop using case statements. Modify it to act like a T flip flop. Validate the modification via a suitable test bench.

Conversion Expression: J = T and K = T

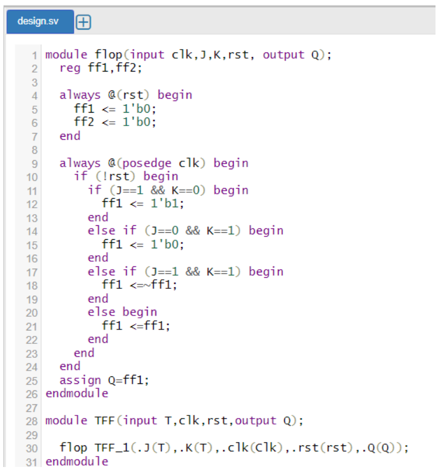


Conversion Table

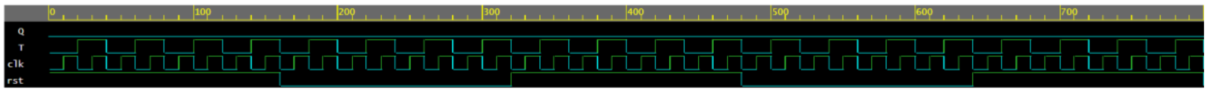
T Input	Outputs		J-K Inputs	
	Qp	Qp+1	J	K
0	0	0	0	X
0	1	1	X	0
1	0	1	1	X
1	1	0	X	1



```
module TFLOP();
  reg clk,T,rst;
  wire Q;

  TFF tff_1(.clk(clk),.T(T),.rst(rst),.Q(Q));

  initial begin
    clk=0;
    T=0;
    rst=1;
  end

  initial forever #10 clk=~clk;
  initial forever #20 T=~T;
  initial forever #160 rst=~rst;

  initial begin
    #800 $finish;
  end

  initial begin
    $dumpfile("TFLOP.vcd");
    $dumpvars;
  end
endmodule
```



```
module flop(input clk,J,K,rst, output Q);
  reg ff1,ff2;

  always @(rst) begin
    ff1 <= 1'b0;
    ff2 <= 1'b0;
  end

  always @(posedge clk) begin
    if (!rst) begin
      if (J==1 && K==0) begin
        ff1 <= 1'b1;
      end
      else if (J==0 && K==1) begin
        ff1 <= 1'b0;
      end
      else if (J==1 && K==1) begin
        ff1 <=~ff1;
      end
      else begin
        ff1 <=ff1;
      end
    end
  end
  assign Q=ff1;
endmodule

module TFF(input T,clk,rst,output Q);

  flop TFF_1(.J(T),.K(T),.clk(Clk),.rst(rst),.Q(Q));
endmodule
```


Q
T
clk
rst