```
==> d_flip_flop.v <==
module d_flip_flop (
    input clk,
    input rst,
    input data,
    output reg q
);
  always @(posedge clk or rst) begin
    q = (rst) ? ~rst : data;
  end
endmodule

==> t_flip_flop.v <==
module t_flip_flop (
    input clk,
    input rst,
    input t,
    output reg q
);
  always @(posedge clk or rst) begin
    q = (rst) ? ~rst : ((t) ? ~q : q);
  end
endmodule

==> jk_flip_flop.v <==
module jk_flip_flop (
    input j,
    input k,
    input clk,
    output reg q
);
  always @(posedge clk) begin
    case ({
      j, k
    })
      2'b00:   q = q;
      2'b01:   q = 1'b0;
      2'b10:   q = 1'b1;
      2'b11:   q = ~q;
      default: q = q;
    endcase
  end
endmodule

==> sr_flip_flop.v <==
module sr_flip_flop (
    input clk,
    input set,
    input rst,
    output reg q
);
  always @(posedge clk) begin
    case ({
      set, rst
    })
      2'b00: q = q;
      2'b01: q = 1'b0;
      2'b10: q = 1'b1;
      2'b11: q = 1'bx;
      default q = q;
    endcase
  end
endmodule

==> d_from_t.v <==
module d_from_t (
    input  data,
    input  clk,
    input  rst,
    output out
```

```
);
  wire med0;
  xor (med0, out, data);
  t_flip_flop U0 (
      .clk(clk),
      .rst(rst),
      .t  (med0),
      .q  (out)
  );
endmodule

==> flip_flop_tb.v <==
module flip_flop_tb ();
  reg inp1, inp2, clk, rst;
  wire t_out, d_out, jk_out, sr_out, dft_out;

  initial begin
    $monitor("Inputs=%b %b :: T=%b D=%b JK=%b SR=%b DfT=%b :: rst=%b", inp1, inp2, t_out, d_out,
             jk_out, sr_out, dft_out, rst);
    {inp1, inp2} = 2'b01;
    rst = 1'b1;
    clk = 1'b0;

    #5 rst = 1'b0;
    #10{inp1, inp2} = 2'b00;
    #10{inp1, inp2} = 2'b01;
    #10{inp1, inp2} = 2'b10;
    #10{inp1, inp2} = 2'b11;
    #20 $finish(0);
  end

  initial begin
    $dumpfile("flip_flops.vcd");
    $dumpvars(0, flip_flop_tb);
  end

  always
    forever begin
      #5 clk = ~clk;
    end

  t_flip_flop U0 (
      .clk(clk),
      .rst(rst),
      .t  (inp1),
      .q  (t_out)
  );

  d_flip_flop U1 (
      .clk(clk),
      .rst(rst),
      .data(inp1),
      .q(d_out)
  );

  jk_flip_flop U2 (
      .clk(clk),
      .j  (inp1),
      .k  (inp2),
      .q  (jk_out)
  );

  sr_flip_flop U3 (
      .clk(clk),
      .set(inp1),
      .rst(inp2),
      .q  (sr_out)
  );

  d_from_t U4 (
      .data(inp1),
```

```
      .clk (clk),
      .rst (rst),
      .out (dft_out)
  );
endmodule

==> output.txt <==
VCD info: dumpfile flip_flops.vcd opened for output.
Inputs=0 1 :: T=0 D=0 JK=x SR=x DfT=0 :: rst=1
Inputs=0 1 :: T=0 D=0 JK=0 SR=0 DfT=0 :: rst=0
Inputs=0 0 :: T=0 D=0 JK=0 SR=0 DfT=0 :: rst=0
Inputs=0 1 :: T=0 D=0 JK=0 SR=0 DfT=0 :: rst=0
Inputs=1 0 :: T=1 D=1 JK=1 SR=1 DfT=1 :: rst=0
Inputs=1 1 :: T=0 D=1 JK=0 SR=x DfT=1 :: rst=0
Inputs=1 1 :: T=1 D=1 JK=1 SR=x DfT=1 :: rst=0
Inputs=1 1 :: T=0 D=1 JK=0 SR=x DfT=1 :: rst=0
```