

```

1  ==> d_flip_flop.v <==
2  module d_flip_flop (
3      input clk,
4      input rst,
5      input data,
6      output reg q
7  );
8      always @(posedge clk or rst) begin
9          q = (rst) ? ~rst : data;
10     end
11 endmodule
12
13 ==> t_flip_flop.v <==
14 module t_flip_flop (
15     input clk,
16     input rst,
17     input t,
18     output reg q
19 );
20     always @(posedge clk or rst) begin
21         q = (rst) ? ~rst : ((t) ? ~q : q);
22     end
23 endmodule
24
25 ==> jk_flip_flop.v <==
26 module jk_flip_flop (
27     input j,
28     input k,
29     input clk,
30     output reg q
31 );
32     always @(posedge clk) begin
33         case ({
34             j, k
35         })
36             2'b00: q = q;
37             2'b01: q = 1'b0;
38             2'b10: q = 1'b1;
39             2'b11: q = ~q;
40             default: q = q;
41         endcase
42     end
43 endmodule
44
45 ==> sr_flip_flop.v <==
46 module sr_flip_flop (
47     input clk,
48     input set,
49     input rst,
50     output reg q
51 );
52     always @(posedge clk) begin
53         case ({
54             set, rst
55         })
56             2'b00: q = q;
57             2'b01: q = 1'b0;
58             2'b10: q = 1'b1;
59             2'b11: q = 1'bx;
60             default q = q;
61         endcase
62     end
63 endmodule
64
65 ==> d_from_t.v <==
66 module d_from_t (
67     input data,
68     input clk,
69     input rst,
70     output out

```

```

71 );
72     wire med0;
73     xor (med0, out, data);
74     t_flip_flop U0 (
75         .clk(clk),
76         .rst(rst),
77         .t  (med0),
78         .q  (out)
79     );
80 endmodule
81
82 ==> flip_flop_tb.v <==
83 module flip_flop_tb ();
84     reg inp1, inp2, clk, rst;
85     wire t_out, d_out, jk_out, sr_out, dft_out;
86
87     initial begin
88         $monitor("Inputs=%b %b :: T=%b D=%b JK=%b SR=%b DfT=%b :: rst=%b", inp1, inp2, t_out, d_out,
89                 jk_out, sr_out, dft_out, rst);
90         {inp1, inp2} = 2'b01;
91         rst = 1'b1;
92         clk = 1'b0;
93
94         #5 rst = 1'b0;
95         #10{inp1, inp2} = 2'b00;
96         #10{inp1, inp2} = 2'b01;
97         #10{inp1, inp2} = 2'b10;
98         #10{inp1, inp2} = 2'b11;
99         #20 $finish(0);
100 end
101
102 initial begin
103     $dumpfile("flip_flops.vcd");
104     $dumpvars(0, flip_flop_tb);
105 end
106
107 always
108     forever begin
109         #5 clk = ~clk;
110     end
111
112 t_flip_flop U0 (
113     .clk(clk),
114     .rst(rst),
115     .t  (inp1),
116     .q  (t_out)
117 );
118
119 d_flip_flop U1 (
120     .clk(clk),
121     .rst(rst),
122     .data(inp1),
123     .q(d_out)
124 );
125
126 jk_flip_flop U2 (
127     .clk(clk),
128     .j  (inp1),
129     .k  (inp2),
130     .q  (jk_out)
131 );
132
133 sr_flip_flop U3 (
134     .clk(clk),
135     .set(inp1),
136     .rst(inp2),
137     .q  (sr_out)
138 );
139
140 d_from_t U4 (
141     .data(inp1),

```

```
142     .clk (clk),
143     .rst (rst),
144     .out (dft_out)
145 );
146 endmodule
147
148 ==> output.txt <==
149 VCD info: dumpfile flip_flops.vcd opened for output.
150 Inputs=0 1 :: T=0 D=0 JK=x SR=x DfT=0 :: rst=1
151 Inputs=0 1 :: T=0 D=0 JK=0 SR=0 DfT=0 :: rst=0
152 Inputs=0 0 :: T=0 D=0 JK=0 SR=0 DfT=0 :: rst=0
153 Inputs=0 1 :: T=0 D=0 JK=0 SR=0 DfT=0 :: rst=0
154 Inputs=1 0 :: T=1 D=1 JK=1 SR=1 DfT=1 :: rst=0
155 Inputs=1 1 :: T=0 D=1 JK=0 SR=x DfT=1 :: rst=0
156 Inputs=1 1 :: T=1 D=1 JK=1 SR=x DfT=1 :: rst=0
157 Inputs=1 1 :: T=0 D=1 JK=0 SR=x DfT=1 :: rst=0
```