

```

1 ==> vending_machine.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         #1 q = (rst) ? ~rst : data;
10    end
11 endmodule
12
13 module vending_machine (
14     input coin_type,
15     input clk,
16     input rst,
17     output is_dispensed,
18     output [1:0] state
19 );
20     d_flip_flop U0 (
21         .clk(clk),
22         .rst(rst),
23         .data(state[1] | coin_type & state[0]),
24         .q(is_dispensed)
25     );
26     d_flip_flop U1 (
27         .clk(clk),
28         .rst(rst),
29         .data(~coin_type & ~state[1]),
30         .q(state[0])
31     );
32     d_flip_flop U2 (
33         .clk(clk),
34         .rst(rst),
35         .data((~coin_type & ~state[1] & state[0]) + (coin_type & ~state[0])),
36         .q(state[1])
37     );
38 endmodule
39
40 ==> tb.py <==
41 import random
42
43 code_template = (
44     lambda start, test_cases: f"""
45 module orange_tb();
46     initial begin
47         $dumpfile("orange_tb.vcd");
48         $dumpvars(0, orange_tb);
49     end
50
51     reg coin, clk, rst;
52     wire [1:0] state;
53     wire dispenser_output;
54
55     initial begin
56         clk = 1'b0;
57         rst = 1'b1;
58         coin = 1'bz;
59         $display("----RESET----");
60 {start}
61         #10 rst = 1'b0;
62         $display("----RESET----");
63 {test_cases}
64         #11 $finish(0);
65     end
66
67     always forever begin
68         #5 clk = ~clk;
69         if (clk) begin
70             $display("inp: %b :: out: %b :: state: %2b :: rst: %b", coin, dispenser_output, state, rst);
71         end
72     end
73
74     vending_machine U0 (
75         .coin_type(coin),
76         .clk(clk),

```

```

76         .rst(rst),
77         .rst(rst),
78         .is_dispensed(dispenser_output),
79         .state(state)
80     );
81 endmodule
82 """
83 )
84
85 # Coin Sequence Length
86 N = 5
87 valid_values = [[0, 1], [1, 0], [0, 0, 0]]
88 numbers = []
89 for _ in range(N):
90     numbers.extend(random.choice(valid_values))
91 print(numbers)
92
93 tests = [f"\t#10 coin=1'b{x};" for x in numbers]
94
95 with open("orange_tb.v", "w") as testbench:
96     print(code_template(tests[0], "\n".join(tests[1:])), file=testbench)
97
98 ==> orange_tb.v <==
99
100 module orange_tb();
101     initial begin
102         $dumpfile("orange_tb.vcd");
103         $dumpvars(0, orange_tb);
104     end
105
106     reg coin, clk, rst;
107     wire [1:0] state;
108     wire dispenser_output;
109
110     initial begin
111         clk = 1'b0;
112         rst = 1'b1;
113         coin = 1'bz;
114         $display("----RESET----");
115         #10 coin=1'b0;
116         #10 rst = 1'b0;
117         $display("----RESET----");
118         #10 coin=1'b1;
119         #10 coin=1'b0;
120         #10 coin=1'b1;
121         #10 coin=1'b0;
122         #10 coin=1'b1;
123         #10 coin=1'b0;
124         #10 coin=1'b0;
125         #10 coin=1'b0;
126         #10 coin=1'b0;
127         #10 coin=1'b0;
128         #10 coin=1'b0;
129         #10 $finish(0);
130     end
131
132     always forever begin
133         #5 clk = ~clk;
134
135         if (clk) begin
136             $display("inp: %b :: out: %b :: state: %2b :: rst: %b", coin, dispenser_output, state, rst);
137         end
138     end
139
140     vending_machine U0 (
141         .coin_type(coin),
142         .clk(clk),
143         .rst(rst),
144         .is_dispensed(dispenser_output),
145         .state(state)
146     );
147 endmodule
148
149 ==> output.txt <==
150 VCD info: dumpfile orange_tb.vcd opened for output.
151 ----RESET----
152 inp: z :: out: 0 :: state: 00 :: rst: 1

```

```
153 inp: 0 :: out: 0 :: state: 00 :: rst: 1
154 ----RESET----
155 inp: 0 :: out: 0 :: state: 01 :: rst: 0
156 inp: 1 :: out: 0 :: state: 11 :: rst: 0
157 inp: 0 :: out: 1 :: state: 00 :: rst: 0
158 inp: 1 :: out: 0 :: state: 01 :: rst: 0
159 inp: 0 :: out: 1 :: state: 00 :: rst: 0
160 inp: 1 :: out: 0 :: state: 01 :: rst: 0
161 inp: 0 :: out: 1 :: state: 00 :: rst: 0
162 inp: 0 :: out: 0 :: state: 01 :: rst: 0
163 inp: 0 :: out: 0 :: state: 11 :: rst: 0
164 inp: 0 :: out: 1 :: state: 00 :: rst: 0
165 inp: 0 :: out: 0 :: state: 01 :: rst: 0
166 inp: 0 :: out: 0 :: state: 11 :: rst: 0
```