```
 1
 2  module conditional_ff_logic (
 3     input [1:0] IR_20_19,
 4     input [31:0] BusMuxOut,
 5     input CONin, clk,
 6     output reg CON
 7     );
 8
 9     wire bus_zero = ~| BusMuxOut;
10     wire bus_neg = BusMuxOut[31];
11
12     wire CON_D;
13
14     assign CON_D = (IR_20_19 == 2'b00) && bus_zero || (IR_20_19 == 2'b01) && !bus_zero
15        || (IR_20_19 == 2'b10) && !bus_neg || (IR_20_19 == 2'b11) && bus_neg;
16
17     initial CON <= 1'b0;
18     always @(posedge clk) if (CONin) CON <= CON_D;
19
20  endmodule
21
22
23
24
25  module con_ff_tb();
26     reg [1:0] IR_20_19;
27     reg [31:0] BusMuxOut;
28     reg CONin, clk;
29     wire CON;
30
31     // Design Under Test
32     conditional_ff_logic DUT (IR_20_19, BusMuxOut, CONin, clk, CON);
33
34     // Establishing clock behaviour.
35     parameter clock_period = 20;
36     initial begin
37        clk <= 0;
38        forever #(clock_period/2) clk <= ~clk;
39     end
40
41     initial begin
42        // Branch if 0
43        IR_20_19 <= 2'b00; BusMuxOut <= 32'b0; CONin <= 1'b0; @(posedge clk) // No impact.
44        IR_20_19 <= 2'b00; BusMuxOut <= 32'b0; CONin <= 1'b1; @(posedge clk) // Success
45        IR_20_19 <= 2'b00; BusMuxOut <= 32'b1; CONin <= 1'b1; @(posedge clk) // Failure
46
47        // Branch if not 0
48        IR_20_19 <= 2'b01; BusMuxOut <= 32'b1; CONin <= 1'b1; @(posedge clk) // Success
49        IR_20_19 <= 2'b01; BusMuxOut <= 32'b0; CONin <= 1'b1; @(posedge clk) // Failure
50
51        // Branch if pos
52        IR_20_19 <= 2'b10; BusMuxOut <= 32'b1; CONin <= 1'b1; @(posedge clk) // Success
53        IR_20_19 <= 2'b10; BusMuxOut <= 32'hFFFF0000; CONin <= 1'b1; @(posedge clk) // Failure
54
55        // Branch if neg
56        IR_20_19 <= 2'b11; BusMuxOut <= 32'hFFFF0000; CONin <= 1'b1; @(posedge clk) // Success
57        IR_20_19 <= 2'b11; BusMuxOut <= 32'b1; CONin <= 1'b1; @(posedge clk); // Failure
58
59        @(posedge clk)
60        $stop;
61     end
62  endmodule
63
```