```
`timescale 1ns / 1ps


module Ball(
    input clk, Go,
    input Top, Bottom, Left, Right,    // State Machine states

    output Count, Upx, Upy, Downx, Downy,
    output [4:0] NS
    );
        wire [4:0] PS;         // Present State

//  State Machine
        // Chill / Start / Go
    assign NS[0] = PS[0] & ~Go;
        // Down-Right   Top/Left   Start
    assign NS[1] = (PS[0] & Go)              | (PS[1] & ~Bottom & ~Right)   | (PS[2]
& Top & ~Right)    | (PS[3] & ~Bottom & Left) | (PS[4] & Top & Left);
        // Up-Right     Bottom/Left
    assign NS[2] = (PS[1] & Bottom & ~Right) | (PS[2] & ~Top & ~Right)      | (PS[3]
& Bottom & Left)   | (PS[4] & ~Top & Left);
        // Down-Left    Top/Right
    assign NS[3] = (PS[1] & ~Bottom & Right) | (PS[2] & Top & Right)        | (PS[3]
& ~Bottom & ~Left) | (PS[4] & Top & ~Left);
        // Up-Left      Bottom/Right
    assign NS[4] = (PS[1] & Bottom & Right)  | (PS[2] & ~Top & Right)       | (PS[3]
& Bottom & ~Left)  | (PS[4] & ~Top & ~Left);


//  FlipFlops to store ball position
    FDRE #(.INIT(1'b1)) BallStorage_0 (.C(clk), .CE(1'b1), .R(1'b0), .D(NS[0]),
.Q(PS[0]));        //Start State
    FDRE #(.INIT(1'b0)) BallStorage_1to4 [4:1] (.C({4{clk}}), .R({4{1'b0}}),
.CE({4{1'b1}}), .D(NS[4:1]), .Q(PS[4:1]));

    assign Count = (PS[0]);
    assign Upx =   (PS[1] | PS[2]);
    assign Upy =   (PS[1] | PS[3]);
    assign Downx = (PS[3] | PS[4]);
    assign Downy = (PS[2] | PS[4]);


endmodule
```