```
module collision (clk, reset, RedPixels, frog_x, frog_y, gameover);

   input logic [15:0][15:0]RedPixels;
   input logic [3:0] frog_x;
   input logic [3:0] frog_y;
   input logic clk;
   input logic reset;
   output logic gameover;


   always_ff @(posedge clk) begin
      integer i;
      integer j;

      if (reset) begin
         gameover <= 0;

      end

      else if ( RedPixels[frog_x][frog_y] == 1) begin
         gameover <= 1;
      end
   end

endmodule

module collision_testbench();
   logic [15:0][15:0]RedPixels;
   logic frog_x;
   logic frog_y;
   logic clk;
   logic reset;
   logic gameover;

   collision dut(clk, reset, RedPixels, frog_x, frog_y, gameover);

      // Set up the clock.
   parameter CLOCK_PERIOD=100;
   initial begin
   clk <= 0;
   forever #(CLOCK_PERIOD/2) clk <= ~clk;
   end

   initial begin
                                    @(posedge clk);
      reset <= 1;                   @(posedge clk);
      reset <= 0;                   @(posedge clk);
                                    @(posedge clk);
      // have scenario of no collision -> collision
      frog_x <= 0; frog_y <= 0;                         @(posedge clk);
                                    @(posedge clk);
      RedPixels[1] <= 16'b1110000110011111;                         @(posedge clk);
                                    @(posedge clk);
      frog_x <= 1; frog_y <= 1;                         @(posedge clk);
                                    @(posedge clk);
                                    @(posedge clk);
                                    @(posedge clk);
                                    @(posedge clk);
                                    @(posedge clk);
                                    @(posedge clk);
                                    @(posedge clk);
                                    @(posedge clk);
                                    @(posedge clk);
                                    @(posedge clk);
                                    @(posedge clk);
                                    @(posedge clk);
                                    @(posedge clk);
                                    @(posedge clk);
                                    @(posedge clk);
                                    @(posedge clk);
                                    @(posedge clk);

      $stop; // End the simulation.
   end

endmodule
```