CSARCH1

Dice Game

Design exercise 2

The objective of this Verilog design project is to implement the scoring logic for the Bo Bing game based on the rolled dice values.

Input: Each dice D_i is represented as a 3-bit input. There are 6 dice (D1, D2, D3, D4, D5, D6) in BoBing.
Output: The output consists of P1, P2, P3, P4, P5,P6 representing (1st Prize, 2nd Prize, 3rd Prize, 4th Prize, 5th Prize, 6th Prize). When the first prize is achieved, P1 will light up. If the second prize is achieved P2 will light up and so on. Only one LED should lit up when a valid prize is won. For example, when winning P3, only P3 is lit and everything else is off. If any of the dice is invalid, all the lights will light up.

Limitations: You can use logic gates, multiplexers and decoders.

The dice value of the input and the prize chart are shown below.

3-bit input D	Dice Value
000	Invalid
001	1
010	2
011	3
100	4
101	5
110	6
111	Invalid

Submission:

1. Verilog File (.v)
    a. Naming convention: DE2_[section]_G[Group Number].v (e.g. DE1_S11_G1.v)
2. Verilog Testbench (_tb.v) - It's okay if it's not exhaustive
    a. Naming convention: DE2_[section]_G[Group Number]_tb.v. (e.g. DE1_S11_G1_tb.v)
3. Circuit Verse File (.cv) [If any]
4. PDF of your documentation of the design process. This should include everything necessary during your design process (i.e. truth table, equations, logic circuit, waveforms, etc.)

Grading System:

- Running Verilog Design [70%] – all possibilities outputs
- Documentation [30%]
- Bonus [10%] to the group with the least number of gates, multiplexers and decoders in each section. If there's a tie, the bonus is divided with the number of groups.

Figure from :
https://modernparenting-onemega.com/how-to-hold-the-chinese-mooncake-dice-game-at-home/

// Sample Testbench:

```
`timescale 1ns / 1ps

module BoBingScoring_tb;

  // Parameters
  parameter CLK_PERIOD = 10;  // Clock period in nanoseconds

  // Signals
  reg [2:0] D1, D2, D3, D4, D5, D6;  // Input signals for dice values
  wire P1, P2, P3, P4, P5, P6;          // Output signals for prizes

  // Instantiate the module
  BoBingScoring uut (
    .D1(D1),
    .D2(D2),
    .D3(D3),
    .D4(D4),
    .D5(D5),
    .D6(D6),
    .P1(P1),
    .P2(P2),
    .P3(P3),
    .P4(P4),
    .P5(P5),
    .P6(P6)
  );

  // Clock generation
  reg clk = 0;
  always #((CLK_PERIOD)/2) clk = ~clk;

  // Test stimuli
  initial begin
    // Reset inputs
    D1 = 0; D2 = 0; D3 = 0; D4 = 0; D5 = 0; D6 = 0;
```

```
    // Wait for a few clock cycles
    #10;

    // Test case 1: Single 4
    D1 = 3'b100;
    #10;
     $display("P = %b%b%b%b%b%b", P1,P2,P3,P4,P5,P6);

    // Test case 2: Two 4s
    D2 = 3'b100;
   $display("P = %b%b%b%b%b%b", P1,P2,P3,P4,P5,P6);
    #10;

    // Test case 3: Three 4s
    D3 = 3'b100;
    $display("P = %b%b%b%b%b%b", P1,P2,P3,P4,P5,P6);
    #10;

    // Test case 4: Four 4s
    D4 = 3'b100;
   $display("P = %b%b%b%b%b%b", P1,P2,P3,P4,P5,P6);
    #10;

    // Test case 5: Five 4s
    D5 = 3'b100;
   $display("P = %b%b%b%b%b%b", P1,P2,P3,P4,P5,P6);
    #10;

    // Finish simulation
    $finish;
  end

endmodule
```