Submitted in partial fulfillment of the course requirements

For CSARCH1

Daniel Gavrie Y. Clemente

Jan Robee E. Feliciano

Derrick M. Valdellon

Ms. Francesca Laguna

March 22, 2024

Table of Contents

Chapter 1: The Project and Its Background

Introduction

Bo Bing, also known as the Mooncake Festival Dice Game, is a customary game that originates from China and is played during the Mooncake Festival, also known as the Mid-Autumn Festival. According to cultural traditions, the game involves the rolling of six six-sided dice in an attempt to test one's luck and win prizes. Players take turns rolling the dice into a bowl made of ceramic or glass, which is placed on a table. The prizes awarded to the players are determined based on the combination of numbers displayed on the dice. The game typically requires a minimum of three players and can be enjoyed by individuals of all age groups (*A Beginner's Guide to the Mooncake Dice Game*, 2022).

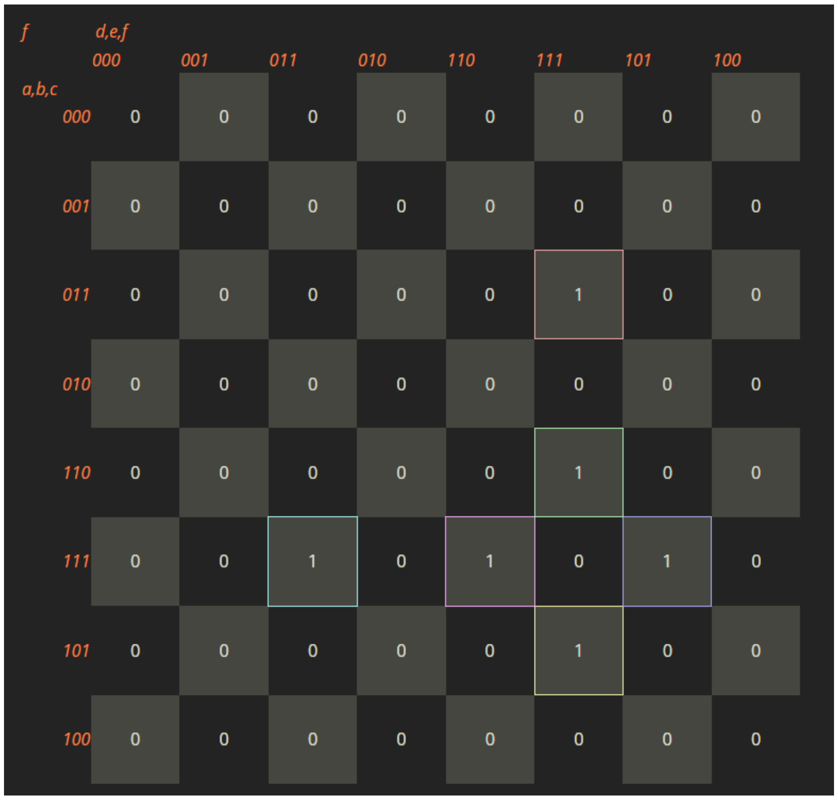
The project will simulate the Mooncake Festival Dice Game and implement the scoring logic based on the rolled dice values. It will be tested against a test bench to determine whether the project waveform matches the waveform of the test bench.

Chapter 2: Circuit Design

Karnaugh Maps

The Karnaugh Map shows which signals would have a value based on the input or the dice rolled. The K-Map helps us map out the logic in which dice were rolled and will return the appropriate prize number.

5 Input Decoder



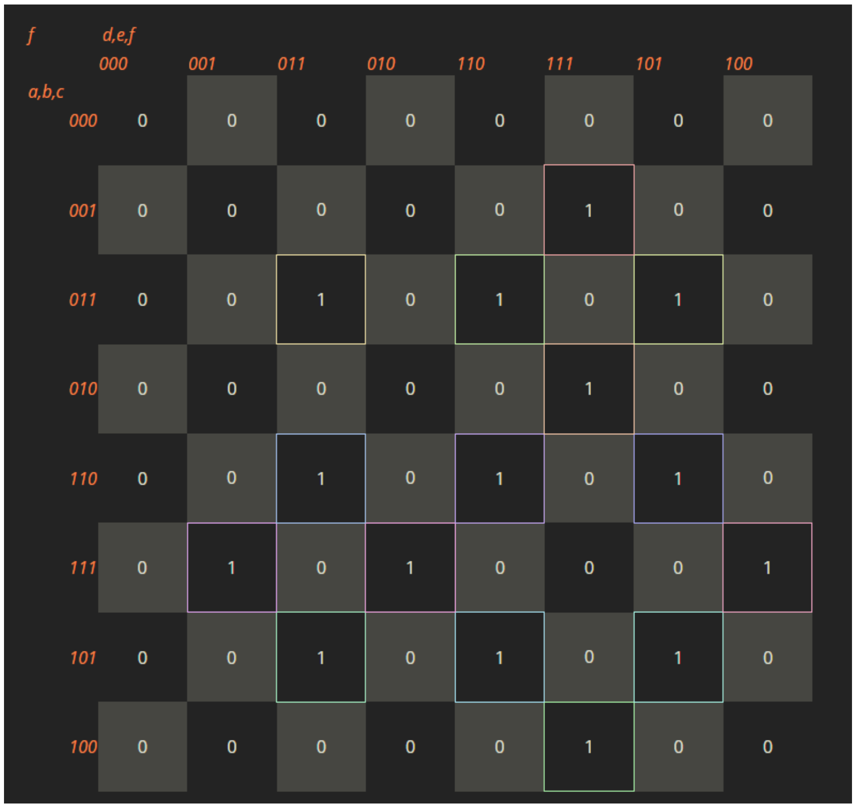
f

a,b,c \ d,e,f	000	001	011	010	110	111	101	100
000	0	0	0	0	0	0	0	0
001	0	0	0	0	0	0	0	0
011	0	0	0	0	0	1	0	0
010	0	0	0	0	0	0	0	0
110	0	0	0	0	0	1	0	0
111	0	0	1	0	1	0	1	0
101	0	0	0	0	0	1	0	0
100	0	0	0	0	0	0	0	0

$$f(a, b, c, d, e, f) = a'bcdef + ab'cdef + abc'def + abcd'ef + abcde'f + abcdef'$$

The Karnaugh map (KMap) above displays the logic for obtaining the five inputs of the same number for the scoring system based on the specifications. The inputs can be used to determine whether the roll is for the 1st place prize.

4 Input Decoder

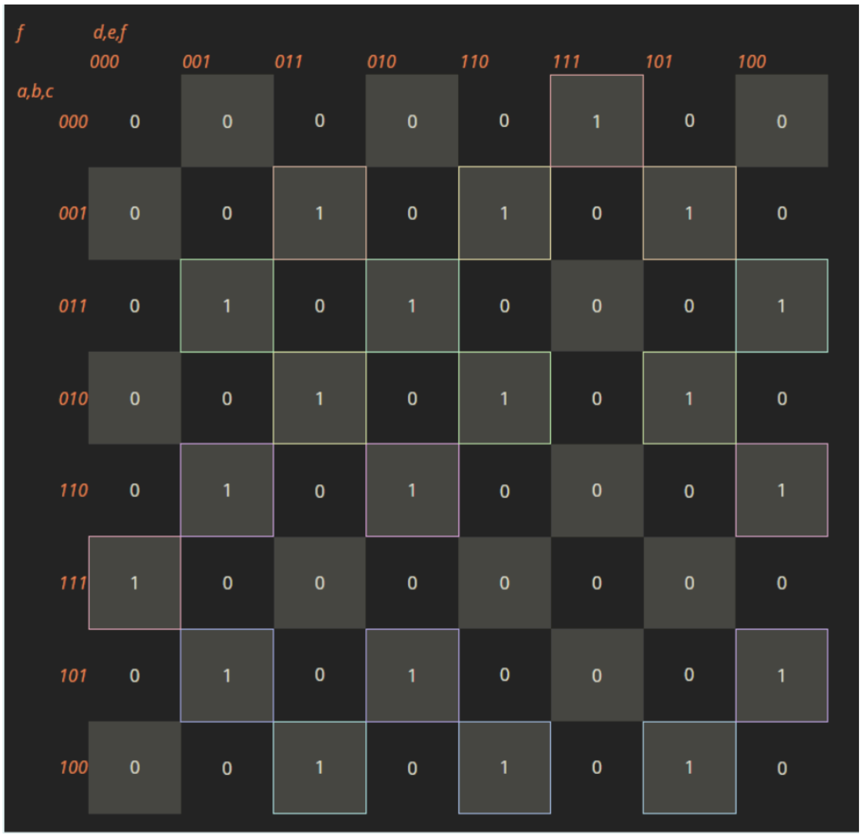


f	d,e,f 000	001	011	010	110	111	101	100
a,b,c 000	0	0	0	0	0	0	0	0
001	0	0	0	0	0	1	0	0
011	0	0	1	0	1	0	1	0
010	0	0	0	0	0	1	0	0
110	0	0	1	0	1	0	1	0
111	0	1	0	1	0	0	0	1
101	0	0	1	0	1	0	1	0
100	0	0	0	0	0	1	0	0

$$f(a, b, c, d, e, f) = a'bc'def + a'bcd'ef + a'bcde'f + a'bcdef' + ab'c'def + ab'cd'ef + ab'cde'f + ab'cdef' + abc'd'ef + abc'de'f + abc'def' + abcd'e'f + abcd'ef' + abcde'f'$$

The KMap above displays the logic for the 4-input decoder. The equation above determines when there are four inputs of the same number. This can be used to determine the first-place prize if the four inputs result in four 4-faces or the fourth-place prize if there are four inputs of the same number except four.

3 Input Decoder



f a,b,c \ d,e,f	000	001	011	010	110	111	101	100
000	0	0	0	0	0	1	0	0
001	0	0	1	0	1	0	1	0
011	0	1	0	1	0	0	0	1
010	0	0	1	0	1	0	1	0
110	0	1	0	1	0	0	0	1
111	1	0	0	0	0	0	0	0
101	0	1	0	1	0	0	0	1
100	0	0	1	0	1	0	1	0

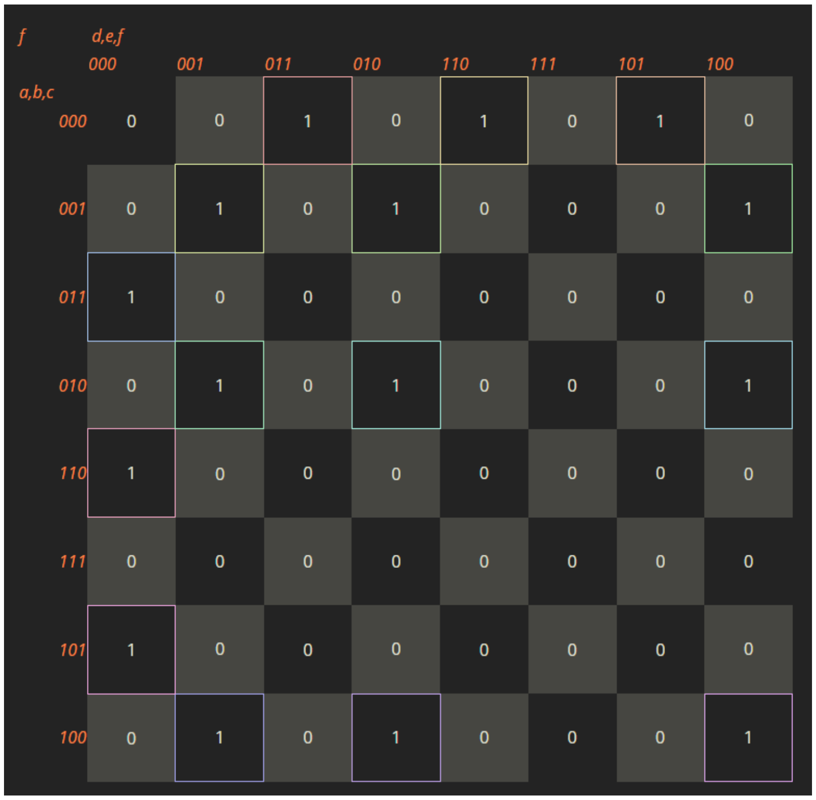
$$f(a, b, c, d, e, f) = a'b'c'def + a'b'cd'ef + a'b'cde'f + a'b'cdef' + a'bc'd'ef + a'bc'de'f + a'bc'def' +$$

$$a'bcd'e'f + a'bcd'ef' + a'bcde'f' + ab'c'd'ef + ab'c'de'f + ab'c'def' + ab'cd'e'f + ab'cd'ef' +$$

$$ab'cde'f' + abc'd'e'f + abc'd'ef' + abc'de'f' + abcd'e'f'$$

The KMap above displays the logic behind getting the prize for a 3-input decoder. It can be used to determine the third-place prize when the input is three 4-face dice.

2 Input Decoder

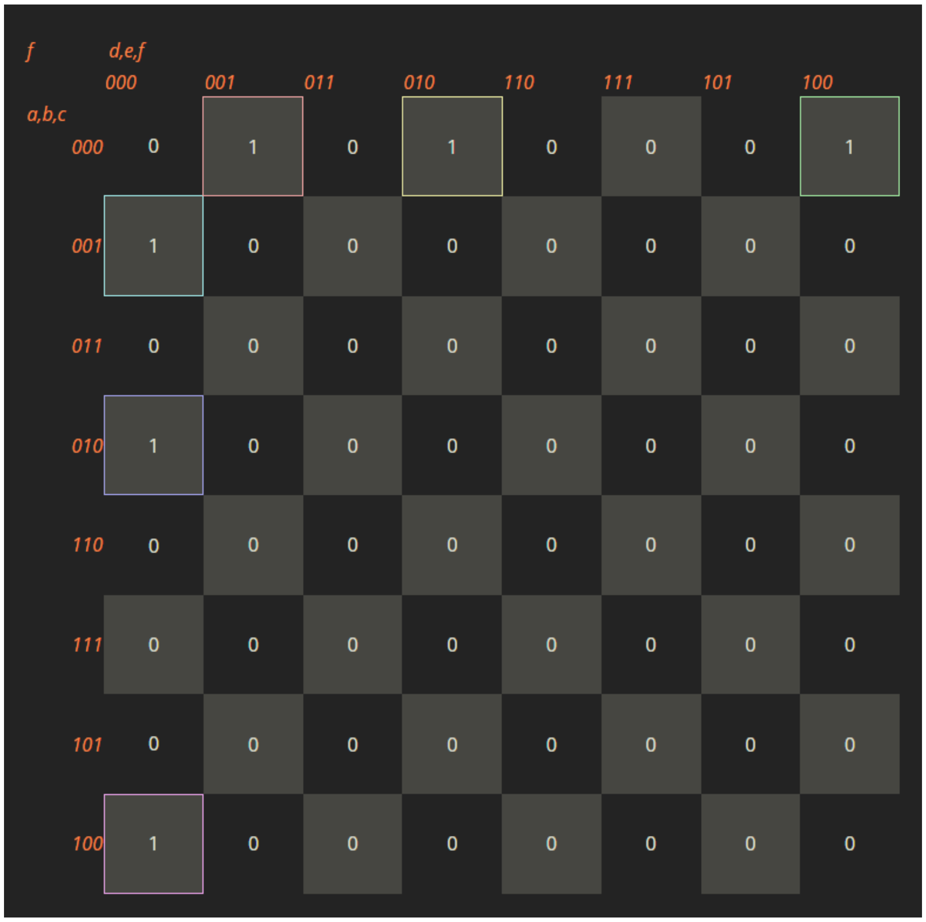


f	d,e,f							
a,b,c	000	001	011	010	110	111	101	100
000	0	0	1	0	1	0	1	0
001	0	1	0	1	0	0	0	1
011	1	0	0	0	0	0	0	0
010	0	1	0	1	0	0	0	1
110	1	0	0	0	0	0	0	0
111	0	0	0	0	0	0	0	0
101	1	0	0	0	0	0	0	0
100	0	1	0	1	0	0	0	1

$$f(a, b, c, d, e, f) = a'b'c'd'ef + a'b'c'de'f + a'b'c'def' + a'b'cd'e'f + a'b'cd'ef' + a'b'cde'f' + a'bc'd'e'f$$
$$+ a'bc'd'ef' + a'bc'de'f' + a'bcd'e'f' + ab'c'd'e'f + ab'c'd'ef' + ab'c'de'f' + ab'cd'e'f' + abc'd'e'f'$$

The Karnaugh Map shows the logic for the 2-input decoder. This is used to help determine when the inputs are two 4-face dice.

1 Input Decoder

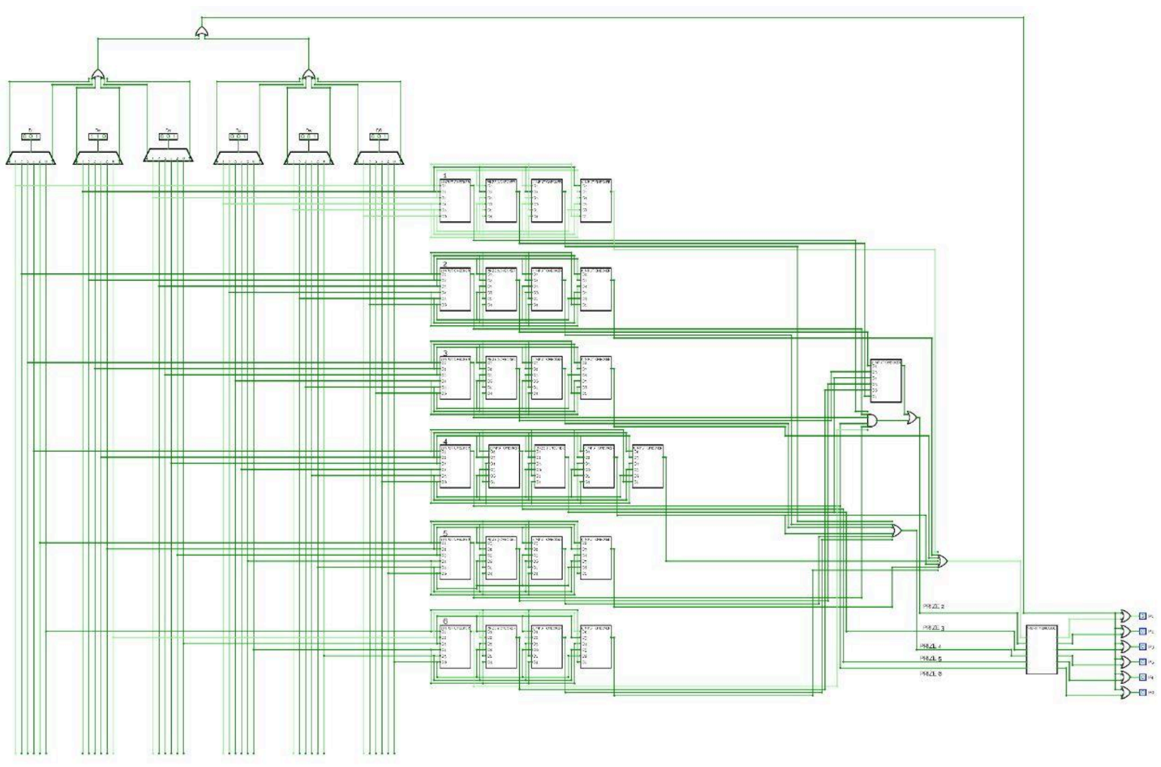


f	d,e,f							
a,b,c	000	001	011	010	110	111	101	100
000	0	1	0	1	0	0	0	1
001	1	0	0	0	0	0	0	0
011	0	0	0	0	0	0	0	0
010	1	0	0	0	0	0	0	0
110	0	0	0	0	0	0	0	0
111	0	0	0	0	0	0	0	0
101	0	0	0	0	0	0	0	0
100	1	0	0	0	0	0	0	0

$$f(a, b, c, d, e, f) = a'b'c'd'e'f + a'b'c'd'ef' + a'b'c'de'f' + a'b'cd'e'f' + a'bc'd'e'f' + ab'c'd'e'f'$$

The KMap above displays the logic behind the one-input decoder. The whole boolean equation evaluates to true when only one of the variables a,b,c,d,e,f evaluates to true—which happens when the die corresponding to that variable is a 4-face. This is used to determine the sixth-place prize.

Logic Circuit



Based on the Karnaugh maps we made, we created a circuit design showing the scoring system based on the dice rolled and the corresponding combinations of dice rolled to show which price will be given.

Chapter 3: Verilog Observations

Verilog Code

```
module main(D2, D6, D3, D4, D1, D5, inp_6);
  input [2:0] D2, D6, D3, D4, D1, D5, inp_6;
  wire Decoder_6_out_0, Decoder_6_out_1, Decoder_6_out_2, Decoder_6_out_3,
Decoder_6_out_4, Decoder_6_out_5, Decoder_6_out_6, Decoder_6_out_7, Decoder_4_out_0,
Decoder_4_out_1, Decoder_4_out_2, Decoder_4_out_3, Decoder_4_out_4, Decoder_4_out_5,
Decoder_4_out_6, Decoder_4_out_7, or_1_out, or_2_out, or_3_out, or_4_out, or_5_out,
or_6_out, or_7_out, or_8_out, \6_out , \5_out , \4_out , \3_out , \2_out , \1_out ,
Decoder_1_out_0, Decoder_1_out_1, Decoder_1_out_2, Decoder_1_out_3, Decoder_1_out_4,
Decoder_1_out_5, Decoder_1_out_6, Decoder_1_out_7, or_0_out, Decoder_3_out_0,
Decoder_3_out_1, Decoder_3_out_2, Decoder_3_out_3, Decoder_3_out_4, Decoder_3_out_5,
Decoder_3_out_6, Decoder_3_out_7, Decoder_2_out_0, Decoder_2_out_1, Decoder_2_out_2,
Decoder_2_out_3, Decoder_2_out_4, Decoder_2_out_5, Decoder_2_out_6, Decoder_2_out_7,
Decoder_5_out_0, Decoder_5_out_1, Decoder_5_out_2, Decoder_5_out_3, Decoder_5_out_4,
Decoder_5_out_5, Decoder_5_out_6, Decoder_5_out_7, Decoder_0_out_0, Decoder_0_out_1,
Decoder_0_out_2, Decoder_0_out_3, Decoder_0_out_4, Decoder_0_out_5, Decoder_0_out_6,
Decoder_0_out_7;
  Decoder8 #(3) Decoder_6(Decoder_6_out_0, Decoder_6_out_1, Decoder_6_out_2,
Decoder_6_out_3, Decoder_6_out_4, Decoder_6_out_5, Decoder_6_out_6, Decoder_6_out_7,
inp_6);
  Multiplexer8 Multiplexer_0(, Decoder_6_out_0, Decoder_6_out_1, Decoder_6_out_2,
Decoder_6_out_3, Decoder_6_out_4, Decoder_6_out_5, Decoder_6_out_6, Decoder_6_out_7,
);
  Decoder8 #(3) Decoder_4(Decoder_4_out_0, Decoder_4_out_1, Decoder_4_out_2,
Decoder_4_out_3, Decoder_4_out_4, Decoder_4_out_5, Decoder_4_out_6, Decoder_4_out_7,
D5);
  assign or_1_out = Decoder_3_out_0 | Decoder_3_out_7 | Decoder_4_out_0 |
Decoder_4_out_7 | Decoder_5_out_0 | Decoder_5_out_7;
  assign or_2_out = or_0_out | or_1_out;
  assign or_3_out = \1_out  | or_2_out;

          always @ (*)
          $display("DigitalLed:P1=%d", or_3_out);
  assign or_4_out = \2_out  | or_2_out;

          always @ (*)
          $display("DigitalLed:P2=%d", or_4_out);
  assign or_5_out = \3_out  | or_2_out;

          always @ (*)
          $display("DigitalLed:P3=%d", or_5_out);
  assign or_6_out = \4_out  | or_2_out;

          always @ (*)
```

```
        $display("DigitalLed:P4=%d", or_6_out);
  assign or_7_out = \5_out  | or_2_out;


        always @ (*)
        $display("DigitalLed:P5=%d", or_7_out);
  assign or_8_out = \6_out  | or_2_out;


        always @ (*)
        $display("DigitalLed:P6=%d", or_8_out);
  assign \6_out  = Decoder_2_out_6 & Decoder_0_out_6 & Decoder_1_out_6 &
Decoder_3_out_7 & Decoder_4_out_6 & Decoder_5_out_6;
  assign \5_out  = Decoder_2_out_5 & Decoder_0_out_5 & Decoder_1_out_5 &
Decoder_3_out_5 & Decoder_4_out_5 & Decoder_5_out_5;
  assign \4_out  = Decoder_2_out_4 & Decoder_0_out_4 & Decoder_1_out_4 &
Decoder_3_out_4 & Decoder_4_out_4 & Decoder_5_out_4;
  assign \3_out  = Decoder_2_out_3 & Decoder_0_out_3 & Decoder_1_out_3 &
Decoder_3_out_3 & Decoder_4_out_3 & Decoder_5_out_3;
  assign \2_out  = Decoder_2_out_2 & Decoder_0_out_2 & Decoder_1_out_2 &
Decoder_3_out_2 & Decoder_4_out_2 & Decoder_5_out_2;
  assign \1_out  = Decoder_2_out_1 & Decoder_0_out_1 & Decoder_1_out_1 &
Decoder_3_out_1 & Decoder_4_out_1 & Decoder_5_out_1;
  Decoder8 #(3) Decoder_1(Decoder_1_out_0, Decoder_1_out_1, Decoder_1_out_2,
Decoder_1_out_3, Decoder_1_out_4, Decoder_1_out_5, Decoder_1_out_6, Decoder_1_out_7,
D1);
  assign or_0_out = Decoder_1_out_0 | Decoder_1_out_7 | Decoder_0_out_0 |
Decoder_0_out_7 | Decoder_2_out_0 | Decoder_2_out_7;
  Decoder8 #(3) Decoder_3(Decoder_3_out_0, Decoder_3_out_1, Decoder_3_out_2,
Decoder_3_out_3, Decoder_3_out_4, Decoder_3_out_5, Decoder_3_out_7, Decoder_3_out_7,
D4);
  Decoder8 #(3) Decoder_2(Decoder_2_out_0, Decoder_2_out_1, Decoder_2_out_2,
Decoder_2_out_3, Decoder_2_out_4, Decoder_2_out_5, Decoder_2_out_6, Decoder_2_out_7,
D3);
  Decoder8 #(3) Decoder_5(Decoder_5_out_0, Decoder_5_out_1, Decoder_5_out_2,
Decoder_5_out_3, Decoder_5_out_4, Decoder_5_out_5, Decoder_5_out_6, Decoder_5_out_7,
D6);
  Decoder8 #(3) Decoder_0(Decoder_0_out_0, Decoder_0_out_1, Decoder_0_out_2,
Decoder_0_out_3, Decoder_0_out_4, Decoder_0_out_5, Decoder_0_out_6, Decoder_0_out_7,
D2);
endmodule


module \1_INPUT_CHECKER (out_0, D1, D2, D3, D4, D5, D6);
  output out_0;
  input D1, D2, D3, D4, D5, D6;
  wire D6_out, or_0_out, not_5_out, and_5_out, and_4_out, D3_out, D4_out, D5_out,
not_4_out, not_3_out, not_2_out, not_1_out, not_0_out;
  assign D6_out = not_2_out & not_1_out & not_0_out & not_3_out & not_4_out & D6;
  assign or_0_out = D6_out | D5_out | D4_out | D3_out | and_4_out | and_5_out;
  assign out_0 = or_0_out;


          always @ (*)
          $display("DigitalLed:or_0_out=%d", or_0_out);
  assign not_5_out = ~D6;
  assign and_5_out = not_2_out & not_1_out & D1 & not_3_out & not_4_out & not_5_out;
  assign and_4_out = not_2_out & D2 & not_0_out & not_3_out & not_4_out & not_5_out;
  assign D3_out = D3 & not_1_out & not_0_out & not_3_out & not_4_out & not_5_out;
  assign D4_out = not_2_out & not_1_out & not_0_out & D4 & not_4_out & not_5_out;
  assign D5_out = not_2_out & not_1_out & not_0_out & not_3_out & D5 & not_5_out;
  assign not_4_out = ~D5;
  assign not_3_out = ~D4;
  assign not_2_out = ~D3;
  assign not_1_out = ~D2;
  assign not_0_out = ~D1;
endmodule


module \2_INPUT_CHECKER (out_0, D1, D2, D3, D4, D5, D6);
  output out_0;
  input D1, D2, D3, D4, D5, D6;
  wire \D1,_D6_out , or_3_out, or_2_out, \D3,_D6_out , or_1_out, \D2,_D6_out , \D4,_D6_out
, or_0_out, \D5,_D6_out , not_5_out, \D1,_D4_out , \D1,_D5_out , \D2,_D4_out ,
\D2,_D5_out , \D3,_D4_out , \D3,_D5_out , \D1,_D2_out , \D1,_D3_out , \D2,_D3_out ,
\D4,_D5_out , not_4_out, not_3_out, not_2_out, not_1_out, not_0_out;
  assign \D1,_D6_out  = not_2_out & not_1_out & D1 & not_3_out & not_4_out & D6;
  assign or_3_out = \D1,_D6_out  | \D1,_D5_out  | \D1,_D4_out ;
  assign or_2_out = or_0_out | or_1_out | or_3_out;
  assign out_0 = or_2_out;


        always @ (*)
        $display("DigitalLed:or_2_out=%d", or_2_out);
  assign \D3,_D6_out  = D3 & not_1_out & not_0_out & not_3_out & not_4_out & D6;
  assign or_1_out = \D2,_D6_out  | \D3,_D5_out  | \D3,_D4_out  | \D3,_D6_out  | \D2,_D5_out
| \D2,_D4_out ;
  assign \D2,_D6_out  = not_2_out & D2 & not_0_out & not_3_out & not_4_out & D6;
  assign \D4,_D6_out  = not_2_out & not_1_out & not_0_out & D4 & not_4_out & D6;
  assign or_0_out = \D5,_D6_out  | \D4,_D6_out  | \D4,_D5_out  | \D2,_D3_out  | \D1,_D3_out
| \D1,_D2_out ;
  assign \D5,_D6_out  = not_2_out & not_1_out & not_0_out & not_3_out & D5 & D6;
  assign not_5_out = ~D6;
  assign \D1,_D4_out  = not_2_out & not_1_out & D1 & D4 & not_4_out & not_5_out;
  assign \D1,_D5_out  = not_2_out & not_1_out & D1 & not_3_out & D5 & not_5_out;
  assign \D2,_D4_out  = not_2_out & D2 & not_0_out & D4 & not_4_out & not_5_out;
  assign \D2,_D5_out  = not_2_out & D2 & not_0_out & not_3_out & D5 & not_5_out;
  assign \D3,_D4_out  = D3 & not_1_out & not_0_out & D4 & not_4_out & not_5_out;
  assign \D3,_D5_out  = D3 & not_1_out & not_0_out & not_3_out & D5 & not_5_out;
  assign \D1,_D2_out  = not_2_out & D2 & D1 & not_3_out & not_4_out & not_5_out;
  assign \D1,_D3_out  = D3 & not_1_out & D1 & not_3_out & not_4_out & not_5_out;
  assign \D2,_D3_out  = D3 & D2 & not_0_out & not_3_out & not_4_out & not_5_out;
  assign \D4,_D5_out  = not_2_out & not_1_out & not_0_out & D4 & D5 & not_5_out;
  assign not_4_out = ~D5;
  assign not_3_out = ~D4;
  assign not_2_out = ~D3;
  assign not_1_out = ~D2;
  assign not_0_out = ~D1;
endmodule


module PRIZE_3_CHECKER(out_0, D1, D2, D3, D4, D5, D6);
  output out_0;
  input D1, D2, D3, D4, D5, D6;
  wire \D1,_D4,_D6_out , or_3_out, or_4_out, \D1,_D5,_D6_out , \D1,_D3,_D6_out ,
or_2_out, \D1,_D2,_D6_out , \D2,_D4,_D6_out , \D2,_D5,_D6_out , or_1_out,
\D2,_D3,_D6_out , \D3,_D4,D6_out , or_0_out, \D3,_D5,_D6_out , \D4,_D5,_D6_out ,
not_5_out, \D1,_D4,_D5_out , \D1,_D3,_D4_out , \D1,_D3,_D5_out , \D1,_D2,_D4_out ,
\D1,_D2,_D5_out , \D2,_D4,_D5_out , \D2,_D3,_D4_out , \D2,_D3,_D5_out ,
\D3,_D4,D5_out , \D1,_D2,_D3_out , not_4_out, not_3_out, not_2_out, not_1_out, not_0_out;
  assign \D1,_D4,_D6_out  = not_2_out & not_1_out & D1 & D4 & not_4_out & D6;
  assign or_3_out = \D1,_D3,_D5_out  | \D1,_D3,_D4_out  | \D1,_D5,_D6_out  |
\D1,_D4,_D5_out  | \D1,_D4,_D6_out ;
  assign or_4_out = or_0_out | or_1_out | or_2_out | or_3_out;
  assign out_0 = or_4_out;


          always @ (*)
          $display("DigitalLed:or_4_out=%d", or_4_out);
  assign \D1,_D5,_D6_out  = not_2_out & not_1_out & D1 & not_3_out & D5 & D6;
  assign \D1,_D3,_D6_out  = D3 & not_1_out & D1 & not_3_out & not_4_out & D6;
  assign or_2_out = \D2,_D4,_D6_out  | \D1,_D2,_D6_out  | \D1,_D2,_D5_out  |
\D1,_D2,_D4_out  | \D1,_D3,_D6_out ;
  assign \D1,_D2,_D6_out  = not_2_out & D2 & D1 & not_3_out & not_4_out & D6;
  assign \D2,_D4,_D6_out  = not_2_out & D2 & not_0_out & D4 & not_4_out & D6;
  assign \D2,_D5,_D6_out  = not_2_out & D2 & not_0_out & not_3_out & D5 & D6;
  assign or_1_out = \D2,_D3,_D6_out  | \D2,_D3,_D5_out  | \D2,_D3,_D4_out  |
\D2,_D5,_D6_out  | \D2,_D4,_D5_out ;
  assign \D2,_D3,_D6_out  = D3 & D2 & not_0_out & not_3_out & not_4_out & D6;
  assign \D3,_D4,D6_out  = D3 & not_1_out & not_0_out & D4 & not_4_out & D6;
  assign or_0_out = \D1,_D2,_D3_out  | \D4,_D5,_D6_out  | \D3,_D5,_D6_out  |
\D3,_D4,D5_out  | \D3,_D4,D6_out ;
  assign \D3,_D5,_D6_out  = D3 & not_1_out & not_0_out & not_3_out & D5 & D6;
  assign \D4,_D5,_D6_out  = not_2_out & not_1_out & not_0_out & D4 & D5 & D6;
  assign not_5_out = ~D6;
  assign \D1,_D4,_D5_out  = not_2_out & not_1_out & D1 & D4 & D5 & not_5_out;
  assign \D1,_D3,_D4_out  = D3 & not_1_out & D1 & D4 & not_4_out & not_5_out;
  assign \D1,_D3,_D5_out  = D3 & not_1_out & D1 & not_3_out & D5 & not_5_out;
  assign \D1,_D2,_D4_out  = not_2_out & D2 & D1 & D4 & not_4_out & not_5_out;
  assign \D1,_D2,_D5_out  = not_2_out & D2 & D1 & not_3_out & D5 & not_5_out;
  assign \D2,_D4,_D5_out  = not_2_out & D2 & not_0_out & D4 & D5 & not_5_out;
  assign \D2,_D3,_D4_out  = D3 & D2 & not_0_out & D4 & not_4_out & not_5_out;
  assign \D2,_D3,_D5_out  = D3 & D2 & not_0_out & not_3_out & D5 & not_5_out;
  assign \D3,_D4,D5_out  = D3 & not_1_out & not_0_out & D4 & D5 & not_5_out;
  assign \D1,_D2,_D3_out  = D3 & D2 & D1 & not_3_out & not_4_out & not_5_out;
  assign not_4_out = ~D5;
  assign not_3_out = ~D4;
  assign not_2_out = ~D3;
  assign not_1_out = ~D2;
  assign not_0_out = ~D1;
endmodule


module try(D2, D6, D3, D4, D1, D5);
  input [2:0] D2, D6, D3, D4, D1, D5;
  wire Decoder_4_out_0, Decoder_4_out_1, Decoder_4_out_2, Decoder_4_out_3,
Decoder_4_out_4, Decoder_4_out_5, Decoder_4_out_6, Decoder_4_out_7,
PRIZE_3_CHECKER_2_out, \2_INPUT_CHECKER_1_out , \1_INPUT_CHECKER_0_out ,
\1_out , Decoder_1_out_0, Decoder_1_out_1, Decoder_1_out_2, Decoder_1_out_3,
Decoder_1_out_4, Decoder_1_out_5, Decoder_1_out_6, Decoder_1_out_7, Decoder_3_out_0,
Decoder_3_out_1, Decoder_3_out_2, Decoder_3_out_3, Decoder_3_out_4, Decoder_3_out_5,
Decoder_3_out_6, Decoder_3_out_7, Decoder_2_out_0, Decoder_2_out_1, Decoder_2_out_2,
Decoder_2_out_3, Decoder_2_out_4, Decoder_2_out_5, Decoder_2_out_6, Decoder_2_out_7,
Decoder_5_out_0, Decoder_5_out_1, Decoder_5_out_2, Decoder_5_out_3, Decoder_5_out_4,
Decoder_5_out_5, Decoder_5_out_6, Decoder_5_out_7, Decoder_0_out_0, Decoder_0_out_1,
Decoder_0_out_2, Decoder_0_out_3, Decoder_0_out_4, Decoder_0_out_5, Decoder_0_out_6,
Decoder_0_out_7;
  Decoder8 #(3) Decoder_4(Decoder_4_out_0, Decoder_4_out_1, Decoder_4_out_2,
Decoder_4_out_3, Decoder_4_out_4, Decoder_4_out_5, Decoder_4_out_6, Decoder_4_out_7,
D5);
  PRIZE_3_CHECKER PRIZE_3_CHECKER_2(PRIZE_3_CHECKER_2_out,
Decoder_4_out_4, Decoder_0_out_4, Decoder_1_out_4, Decoder_5_out_4, Decoder_2_out_4,
Decoder_3_out_4);


        always @ (*)
        $display("DigitalLed:PRIZE_3_CHECKER_2_out=%d",
PRIZE_3_CHECKER_2_out);
  \2_INPUT_CHECKER  \2_INPUT_CHECKER_1 (\2_INPUT_CHECKER_1_out ,
Decoder_5_out_4, Decoder_1_out_4, Decoder_0_out_4, Decoder_2_out_4, Decoder_3_out_4,
Decoder_4_out_4);


        always @ (*)
        $display("DigitalLed:\2_INPUT_CHECKER_1_out =%d",
\2_INPUT_CHECKER_1_out );
  \1_INPUT_CHECKER  \1_INPUT_CHECKER_0 (\1_INPUT_CHECKER_0_out ,
Decoder_1_out_4, Decoder_0_out_4, Decoder_2_out_4, Decoder_3_out_4, Decoder_4_out_4,
Decoder_5_out_4);


            always @ (*)
            $display("DigitalLed:\1_INPUT_CHECKER_0_out =%d",
\1_INPUT_CHECKER_0_out );
  \1_INPUT_CHECKER  \1 (\1_out , Decoder_1_out_1, Decoder_0_out_1, Decoder_2_out_1,
Decoder_3_out_1, Decoder_4_out_1, Decoder_5_out_1);
  Decoder8 #(3) Decoder_1(Decoder_1_out_0, Decoder_1_out_1, Decoder_1_out_2,
Decoder_1_out_3, Decoder_1_out_4, Decoder_1_out_5, Decoder_1_out_6, Decoder_1_out_7,
D1);
  Decoder8 #(3) Decoder_3(Decoder_3_out_0, Decoder_3_out_1, Decoder_3_out_2,
Decoder_3_out_3, Decoder_3_out_4, Decoder_3_out_5, Decoder_3_out_7, Decoder_3_out_7,
D4);
  Decoder8 #(3) Decoder_2(Decoder_2_out_0, Decoder_2_out_1, Decoder_2_out_2,
Decoder_2_out_3, Decoder_2_out_4, Decoder_2_out_5, Decoder_2_out_6, Decoder_2_out_7,
D3);
  Decoder8 #(3) Decoder_5(Decoder_5_out_0, Decoder_5_out_1, Decoder_5_out_2,
Decoder_5_out_3, Decoder_5_out_4, Decoder_5_out_5, Decoder_5_out_6, Decoder_5_out_7,
D6);
  Decoder8 #(3) Decoder_0(Decoder_0_out_0, Decoder_0_out_1, Decoder_0_out_2,
Decoder_0_out_3, Decoder_0_out_4, Decoder_0_out_5, Decoder_0_out_6, Decoder_0_out_7,
D2);
endmodule


module \4_INPUT_CHECKER (out_0, D1, D2, D3, D4, D5, D6);
  output out_0;
  input D1, D2, D3, D4, D5, D6;
  wire \D1,_D2,_D3,_D6_out , or_3_out, or_1_out, \D1,_D2,_D4,_D6_out , or_2_out,
\D1,_D2,_D5,_D6_out , \D1,_D3,_D4,_D6_out , \D1,_D3,_D5,_D6_out ,
\D1,_D4,_D5,_D6_out , or_0_out, \D2,_D3,_D4,_D6_out , \D2,_D3,_D5,_D6_out ,
\D2,_D4,_D5,_D6_out , \D3,_D4,_D5,_D6_out , not_5_out, \D1,_D2,_D3,_D4_out ,
\D1,_D2,_D3,_D5_out , \D1,_D2,_D4,_D5_out , \D1,_D3,_D4,_D5_out ,
\D2,_D3,_D4,_D5_out , not_4_out, not_3_out, not_2_out, not_1_out, not_0_out;
  assign \D1,_D2,_D3,_D6_out  = D3 & D2 & D1 & not_3_out & not_4_out & D6;
  assign or_3_out = \D1,_D2,_D3,_D6_out  | \D1,_D2,_D3,_D5_out  | \D1,_D2,_D3,_D4_out ;
  assign or_1_out = or_0_out | or_2_out | or_3_out;
  assign out_0 = or_1_out;


        always @ (*)
        $display("DigitalLed:or_1_out=%d", or_1_out);
  assign \D1,_D2,_D4,_D6_out  = not_2_out & D2 & D1 & D4 & not_4_out & D6;
  assign or_2_out = \D1,_D3,_D5,_D6_out  | \D1,_D3,_D4,_D6_out  | \D1,_D3,_D4,_D5_out  |
\D1,_D2,_D5,_D6_out  | \D1,_D2,_D4,_D6_out  | \D1,_D2,_D4,_D5_out ;
```

```
  assign \D1,_D2,_D5,_D6_out  = not_2_out & D2 & D1 & not_3_out & D5 & D6;

  assign \D1,_D3,_D4,_D6_out  = D3 & not_1_out & D1 & D4 & not_4_out & D6;

  assign \D1,_D3,_D5,_D6_out  = D3 & not_1_out & D1 & not_3_out & D5 & D6;

  assign \D1,_D4,_D5,_D6_out  = not_2_out & not_1_out & D1 & D4 & D5 & D6;

  assign or_0_out = \D3,_D4,_D5,_D6_out  | \D2,_D4,_D5,_D6_out  | \D2,_D3,_D5,_D6_out  |
\D2,_D3,_D4,_D6_out  | \D2,_D3,_D4,_D5_out  | \D1,_D4,_D5,_D6_out ;

  assign \D2,_D3,_D4,_D6_out  = D3 & D2 & not_0_out & D4 & not_4_out & D6;

  assign \D2,_D3,_D5,_D6_out  = D3 & D2 & not_0_out & not_3_out & D5 & D6;

  assign \D2,_D4,_D5,_D6_out  = not_2_out & D2 & not_0_out & D4 & D5 & D6;

  assign \D3,_D4,_D5,_D6_out  = D3 & not_1_out & not_0_out & D4 & D5 & D6;

  assign not_5_out = ~D6;

  assign \D1,_D2,_D3,_D4_out  = D3 & D2 & D1 & D4 & not_4_out & not_5_out;

  assign \D1,_D2,_D3,_D5_out  = D3 & D2 & D1 & not_3_out & D5 & not_5_out;

  assign \D1,_D2,_D4,_D5_out  = not_2_out & D2 & D1 & D4 & D5 & not_5_out;

  assign \D1,_D3,_D4,_D5_out  = D3 & not_1_out & D1 & D4 & D5 & not_5_out;

  assign \D2,_D3,_D4,_D5_out  = D3 & D2 & not_0_out & D4 & D5 & not_5_out;

  assign not_4_out = ~D5;

  assign not_3_out = ~D4;

  assign not_2_out = ~D3;

  assign not_1_out = ~D2;

  assign not_0_out = ~D1;

endmodule
```

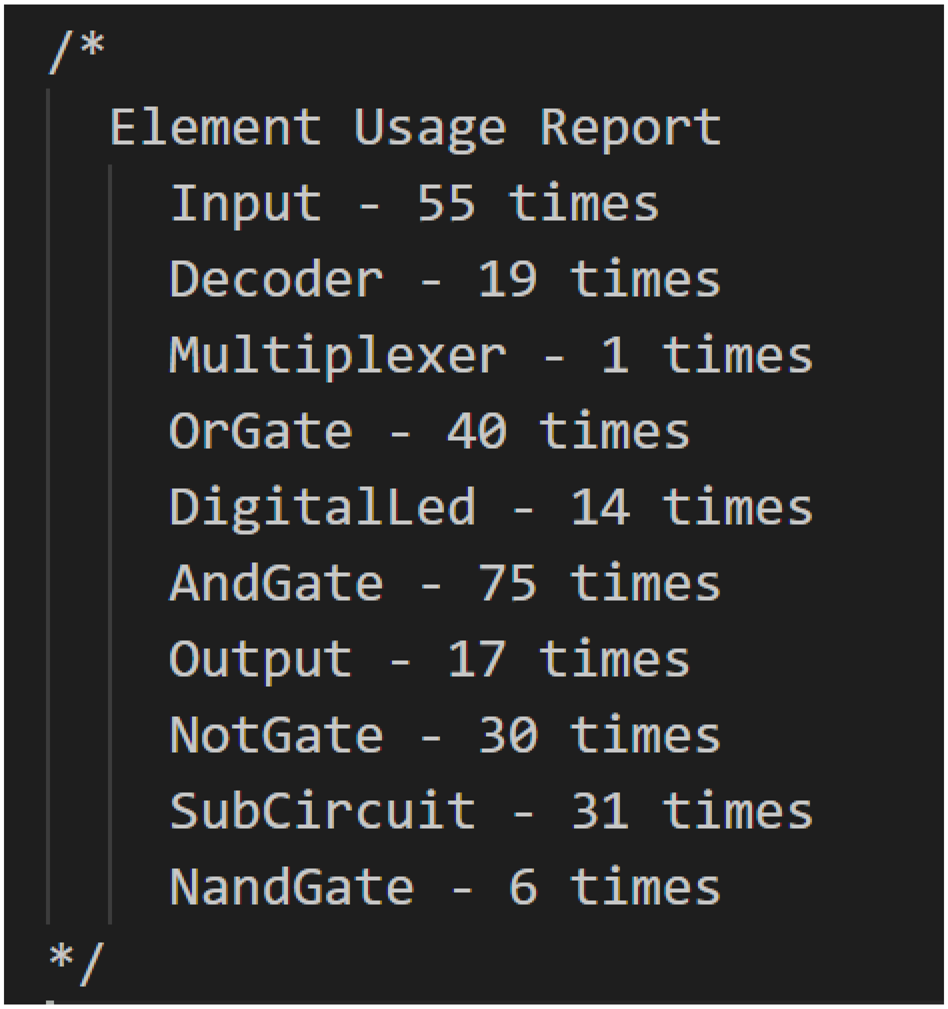
```
module \5_INPUT_CHECKER (out_0, D1, D2, D3, D4, D5, D6);
  output out_0;
  input D1, D2, D3, D4, D5, D6;
  wire \D1,_D2,_D3,_D4,_D6_out , or_0_out, \D1,_D2,_D3,_D5,_D6_out ,
\D1,_D2,_D4,_D5,_D6_out , \D1,_D3,_D4,_D5,_D6_out , \D2,_D3,_D4,_D5,_D6_out ,
not_5_out, \D1,_D2,_D3,_D4,_D5_out , not_4_out, not_3_out, not_2_out, not_1_out,
not_0_out;
  assign \D1,_D2,_D3,_D4,_D6_out  = D3 & D2 & D1 & D4 & not_4_out & D6;
  assign or_0_out = \D2,_D3,_D4,_D5,_D6_out  | \D1,_D3,_D4,_D5,_D6_out  |
\D1,_D2,_D4,_D5,_D6_out  | \D1,_D2,_D3,_D5,_D6_out  | \D1,_D2,_D3,_D4,_D6_out  |
\D1,_D2,_D3,_D4,_D5_out ;
  assign out_0 = or_0_out;


        always @ (*)
        $display("DigitalLed:or_0_out=%d", or_0_out);
  assign \D1,_D2,_D3,_D5,_D6_out  = D3 & D2 & D1 & not_3_out & D5 & D6;
  assign \D1,_D2,_D4,_D5,_D6_out  = not_2_out & D2 & D1 & D4 & D5 & D6;
  assign \D1,_D3,_D4,_D5,_D6_out  = D3 & not_1_out & D1 & D4 & D5 & D6;
  assign \D2,_D3,_D4,_D5,_D6_out  = D3 & D2 & not_0_out & D4 & D5 & D6;
  assign not_5_out = ~D6;
  assign \D1,_D2,_D3,_D4,_D5_out  = D3 & D2 & D1 & D4 & D5 & not_5_out;
  assign not_4_out = ~D5;
  assign not_3_out = ~D4;
  assign not_2_out = ~D3;
  assign not_1_out = ~D2;
  assign not_0_out = ~D1;
endmodule


module PRIORITY_ENCODER(out_0, out_1, out_2, out_3, out_4, out_5, inp_0, inp_1,
inp_2, inp_3, inp_4, inp_5);
  output out_0,  out_1,  out_2,  out_3,  out_4,  out_5;
  input inp_0, inp_1, inp_2, inp_3, inp_4, inp_5;
  wire and_5_out, nand_5_out, or_3_out, and_4_out, nand_4_out, and_3_out, nand_3_out,
or_2_out, and_2_out, nand_2_out, or_1_out, or_0_out, and_1_out, nand_1_out, nand_0_out,
and_0_out;
  assign and_5_out = nand_5_out & inp_5;
  assign out_5 = and_5_out;
  assign nand_5_out = ~(or_3_out & inp_5);
  assign or_3_out = or_2_out | inp_4;
  assign and_4_out = nand_4_out & inp_4;
  assign out_4 = and_4_out;
  assign nand_4_out = ~(or_2_out & inp_4);
  assign and_3_out = nand_3_out & inp_3;
  assign out_3 = and_3_out;
  assign nand_3_out = ~(or_1_out & inp_3);
  assign or_2_out = or_1_out | inp_3;
  assign and_2_out = nand_2_out & inp_2;

  assign out_2 = and_2_out;

  assign nand_2_out = ~(or_0_out & inp_2);

  assign or_1_out = or_0_out | inp_2;

  assign or_0_out = inp_0 | inp_1;

  assign and_1_out = nand_1_out & inp_1;

  assign out_1 = and_1_out;

  assign nand_1_out = ~(inp_0 & inp_1);

  assign nand_0_out = ~( & inp_0);

  assign and_0_out = nand_0_out & inp_0;

  assign out_0 = and_0_out;

endmodule









module BoBingScoring(D1,D2,D3,D4,D5,D6,P1,P2,P3,P4,P5,P6);

  output P1,  P2,  P3,  P4,  P5,  P6;

  input [2:0] D1, D2, D3, D4, D5, D6;

  wire Decoder_5_out_0, Decoder_5_out_1, Decoder_5_out_2, Decoder_5_out_3,
Decoder_5_out_4, Decoder_5_out_5, Decoder_5_out_6, Decoder_5_out_7, or_4_out,
or_5_out, or_11_out, or_10_out, or_9_out, or_8_out, or_7_out, or_6_out,
\5_INPUT_CHECKER_24_out , or_2_out, PRIORITY_ENCODER_26_out_0,
PRIORITY_ENCODER_26_out_1, PRIORITY_ENCODER_26_out_2,
PRIORITY_ENCODER_26_out_3, PRIORITY_ENCODER_26_out_4,
PRIORITY_ENCODER_26_out_5, \4_INPUT_CHECKER_23_out , or_1_out,
PRIZE_3_CHECKER_10_out, \2_INPUT_CHECKER_25_out , or_0_out,
\1_INPUT_CHECKER_9_out , and_0_out, \5_INPUT_CHECKER_22_out ,
\4_INPUT_CHECKER_21_out , PRIZE_3_CHECKER_8_out, \1_INPUT_CHECKER_7_out
, \5_INPUT_CHECKER_20_out , \4_INPUT_CHECKER_19_out ,
PRIZE_3_CHECKER_11_out, \2_INPUT_CHECKER_6_out , \1_INPUT_CHECKER_5_out
, \4_INPUT_CHECKER_17_out , \5_INPUT_CHECKER_18_out ,
PRIZE_3_CHECKER_4_out, \1_INPUT_CHECKER_3_out , \5_INPUT_CHECKER_16_out
, \4_INPUT_CHECKER_14_out , PRIZE_3_CHECKER_15_out,
\1_INPUT_CHECKER_1_out , \5_INPUT_CHECKER_13_out ,
\4_INPUT_CHECKER_12_out , PRIZE_3_CHECKER_2_out, \1_INPUT_CHECKER_0_out
, Decoder_4_out_0, Decoder_4_out_1, Decoder_4_out_2, Decoder_4_out_3,
Decoder_4_out_4, Decoder_4_out_5, Decoder_4_out_6, Decoder_4_out_7, Decoder_3_out_0,
Decoder_3_out_1, Decoder_3_out_2, Decoder_3_out_3, Decoder_3_out_4, Decoder_3_out_5,
Decoder_3_out_6, Decoder_3_out_7, or_3_out, Decoder_2_out_0, Decoder_2_out_1,
Decoder_2_out_2, Decoder_2_out_3, Decoder_2_out_4, Decoder_2_out_5, Decoder_2_out_6,
Decoder_2_out_7, Decoder_1_out_0, Decoder_1_out_1, Decoder_1_out_2, Decoder_1_out_3,
Decoder_1_out_4, Decoder_1_out_5, Decoder_1_out_6, Decoder_1_out_7, Decoder_0_out_0,
Decoder_0_out_1, Decoder_0_out_2, Decoder_0_out_3, Decoder_0_out_4, Decoder_0_out_5,
Decoder_0_out_6, Decoder_0_out_7;
  Decoder8 #(3) Decoder_5(Decoder_5_out_0, Decoder_5_out_1, Decoder_5_out_2,
Decoder_5_out_3, Decoder_5_out_4, Decoder_5_out_5, Decoder_5_out_6, Decoder_5_out_7,
D5);
  assign or_4_out = Decoder_2_out_0 | Decoder_2_out_7 | Decoder_5_out_0 |
Decoder_5_out_7 | Decoder_4_out_0 | Decoder_4_out_7;
  assign or_5_out = or_3_out | or_4_out;
  assign or_11_out = or_5_out | PRIORITY_ENCODER_26_out_5;
  assign P6 = or_11_out;
  assign or_10_out = or_5_out | PRIORITY_ENCODER_26_out_4;
  assign P4 = or_10_out;
  assign or_9_out = or_5_out | PRIORITY_ENCODER_26_out_3;
  assign P5 = or_9_out;
  assign or_8_out = or_5_out | PRIORITY_ENCODER_26_out_2;
  assign P3 = or_8_out;
  assign or_7_out = or_5_out | PRIORITY_ENCODER_26_out_1;
  assign P2 = or_7_out;
  assign or_6_out = or_5_out | PRIORITY_ENCODER_26_out_0;
  assign P1 = or_6_out;
  \5_INPUT_CHECKER  \5_INPUT_CHECKER_24 (\5_INPUT_CHECKER_24_out ,
Decoder_1_out_6, Decoder_0_out_6, Decoder_3_out_6, Decoder_2_out_6, Decoder_5_out_6,
Decoder_4_out_6);
```

```
  assign or_2_out = \5_INPUT_CHECKER_13_out  | \5_INPUT_CHECKER_16_out  |
\5_INPUT_CHECKER_18_out  | \5_INPUT_CHECKER_20_out  |
\4_INPUT_CHECKER_19_out  | \5_INPUT_CHECKER_22_out  |
\5_INPUT_CHECKER_24_out ;
  PRIORITY_ENCODER PRIORITY_ENCODER_26(PRIORITY_ENCODER_26_out_0,
PRIORITY_ENCODER_26_out_1, PRIORITY_ENCODER_26_out_2,
PRIORITY_ENCODER_26_out_3, PRIORITY_ENCODER_26_out_4,
PRIORITY_ENCODER_26_out_5, or_2_out, or_0_out, PRIZE_3_CHECKER_11_out,
or_1_out, \2_INPUT_CHECKER_6_out , \1_INPUT_CHECKER_5_out );
  \4_INPUT_CHECKER  \4_INPUT_CHECKER_23 (\4_INPUT_CHECKER_23_out ,
Decoder_1_out_6, Decoder_0_out_6, Decoder_3_out_6, Decoder_2_out_6, Decoder_5_out_6,
Decoder_4_out_6);
  assign or_1_out = \4_INPUT_CHECKER_12_out  | \4_INPUT_CHECKER_14_out  |
\4_INPUT_CHECKER_17_out  | \4_INPUT_CHECKER_19_out  |
\4_INPUT_CHECKER_21_out  | \4_INPUT_CHECKER_23_out ;
  PRIZE_3_CHECKER PRIZE_3_CHECKER_10(PRIZE_3_CHECKER_10_out,
Decoder_1_out_6, Decoder_0_out_6, Decoder_3_out_6, Decoder_2_out_6, Decoder_5_out_6,
Decoder_4_out_6);
  \2_INPUT_CHECKER  \2_INPUT_CHECKER_25 (\2_INPUT_CHECKER_25_out ,
PRIZE_3_CHECKER_2_out, PRIZE_3_CHECKER_15_out, PRIZE_3_CHECKER_4_out,
PRIZE_3_CHECKER_11_out, PRIZE_3_CHECKER_8_out, PRIZE_3_CHECKER_10_out);
  assign or_0_out = \2_INPUT_CHECKER_25_out  | and_0_out;
  \1_INPUT_CHECKER  \1_INPUT_CHECKER_9 (\1_INPUT_CHECKER_9_out ,
Decoder_1_out_6, Decoder_0_out_6, Decoder_3_out_6, Decoder_2_out_6, Decoder_5_out_6,
Decoder_4_out_6);
  assign and_0_out = \1_INPUT_CHECKER_3_out  & \1_INPUT_CHECKER_1_out  &
\1_INPUT_CHECKER_0_out  & \1_INPUT_CHECKER_5_out  &
\1_INPUT_CHECKER_7_out  & \1_INPUT_CHECKER_9_out ;
  \5_INPUT_CHECKER  \5_INPUT_CHECKER_22 (\5_INPUT_CHECKER_22_out ,
Decoder_1_out_5, Decoder_0_out_5, Decoder_3_out_5, Decoder_2_out_5, Decoder_5_out_5,
Decoder_4_out_5);
  \4_INPUT_CHECKER  \4_INPUT_CHECKER_21 (\4_INPUT_CHECKER_21_out ,
Decoder_1_out_5, Decoder_0_out_5, Decoder_3_out_5, Decoder_2_out_5, Decoder_4_out_5,
Decoder_5_out_5);
  PRIZE_3_CHECKER PRIZE_3_CHECKER_8(PRIZE_3_CHECKER_8_out,
Decoder_1_out_5, Decoder_0_out_5, Decoder_3_out_5, Decoder_2_out_5, Decoder_5_out_5,
Decoder_4_out_5);
  \1_INPUT_CHECKER  \1_INPUT_CHECKER_7 (\1_INPUT_CHECKER_7_out ,
Decoder_1_out_5, Decoder_0_out_5, Decoder_3_out_5, Decoder_2_out_5, Decoder_5_out_5,
Decoder_4_out_5);
  \5_INPUT_CHECKER  \5_INPUT_CHECKER_20 (\5_INPUT_CHECKER_20_out ,
Decoder_1_out_4, Decoder_0_out_4, Decoder_3_out_4, Decoder_2_out_4, Decoder_5_out_4,
Decoder_4_out_4);
  \4_INPUT_CHECKER  \4_INPUT_CHECKER_19 (\4_INPUT_CHECKER_19_out ,
Decoder_1_out_4, Decoder_0_out_4, Decoder_3_out_4, Decoder_2_out_4, Decoder_5_out_4,
Decoder_4_out_4);
```

```
  PRIZE_3_CHECKER PRIZE_3_CHECKER_11(PRIZE_3_CHECKER_11_out,
Decoder_1_out_4, Decoder_0_out_4, Decoder_3_out_4, Decoder_2_out_4, Decoder_5_out_4,
Decoder_4_out_4);
  \2_INPUT_CHECKER  \2_INPUT_CHECKER_6 (\2_INPUT_CHECKER_6_out ,
Decoder_1_out_4, Decoder_0_out_4, Decoder_3_out_4, Decoder_2_out_4, Decoder_5_out_4,
Decoder_4_out_4);
  \1_INPUT_CHECKER  \1_INPUT_CHECKER_5 (\1_INPUT_CHECKER_5_out ,
Decoder_1_out_4, Decoder_0_out_4, Decoder_3_out_4, Decoder_2_out_4, Decoder_5_out_4,
Decoder_4_out_4);
  \4_INPUT_CHECKER  \4_INPUT_CHECKER_17 (\4_INPUT_CHECKER_17_out ,
Decoder_1_out_3, Decoder_0_out_3, Decoder_3_out_3, Decoder_2_out_3, Decoder_5_out_3,
Decoder_4_out_3);
  \5_INPUT_CHECKER  \5_INPUT_CHECKER_18 (\5_INPUT_CHECKER_18_out ,
Decoder_1_out_3, Decoder_0_out_3, Decoder_3_out_3, Decoder_4_out_3, Decoder_5_out_3,
Decoder_2_out_3);
  PRIZE_3_CHECKER PRIZE_3_CHECKER_4(PRIZE_3_CHECKER_4_out,
Decoder_1_out_3, Decoder_0_out_3, Decoder_3_out_3, Decoder_2_out_3, Decoder_5_out_3,
Decoder_4_out_3);
  \1_INPUT_CHECKER  \1_INPUT_CHECKER_3 (\1_INPUT_CHECKER_3_out ,
Decoder_1_out_3, Decoder_0_out_3, Decoder_3_out_3, Decoder_2_out_3, Decoder_5_out_3,
Decoder_4_out_3);
  \5_INPUT_CHECKER  \5_INPUT_CHECKER_16 (\5_INPUT_CHECKER_16_out ,
Decoder_1_out_2, Decoder_0_out_2, Decoder_3_out_2, Decoder_2_out_2, Decoder_5_out_2,
Decoder_4_out_2);
  \4_INPUT_CHECKER  \4_INPUT_CHECKER_14 (\4_INPUT_CHECKER_14_out ,
Decoder_1_out_2, Decoder_0_out_2, Decoder_3_out_2, Decoder_2_out_2, Decoder_4_out_2,
Decoder_5_out_2);
  PRIZE_3_CHECKER PRIZE_3_CHECKER_15(PRIZE_3_CHECKER_15_out,
Decoder_1_out_2, Decoder_0_out_2, Decoder_3_out_2, Decoder_2_out_2, Decoder_5_out_2,
Decoder_4_out_2);
  \1_INPUT_CHECKER  \1_INPUT_CHECKER_1 (\1_INPUT_CHECKER_1_out ,
Decoder_1_out_2, Decoder_0_out_2, Decoder_3_out_2, Decoder_2_out_2, Decoder_5_out_2,
Decoder_4_out_2);
  \5_INPUT_CHECKER  \5_INPUT_CHECKER_13 (\5_INPUT_CHECKER_13_out ,
Decoder_1_out_1, Decoder_0_out_1, Decoder_3_out_1, Decoder_2_out_1, Decoder_5_out_1,
Decoder_4_out_1);
  \4_INPUT_CHECKER  \4_INPUT_CHECKER_12 (\4_INPUT_CHECKER_12_out ,
Decoder_1_out_1, Decoder_0_out_1, Decoder_3_out_1, Decoder_2_out_1, Decoder_5_out_1,
Decoder_4_out_1);
  PRIZE_3_CHECKER PRIZE_3_CHECKER_2(PRIZE_3_CHECKER_2_out,
Decoder_1_out_1, Decoder_0_out_1, Decoder_3_out_1, Decoder_2_out_1, Decoder_5_out_1,
Decoder_4_out_1);
  \1_INPUT_CHECKER  \1_INPUT_CHECKER_0 (\1_INPUT_CHECKER_0_out ,
Decoder_1_out_1, Decoder_0_out_1, Decoder_3_out_1, Decoder_2_out_1, Decoder_5_out_1,
Decoder_4_out_1);
  Decoder8 #(3) Decoder_4(Decoder_4_out_0, Decoder_4_out_1, Decoder_4_out_2,
Decoder_4_out_3, Decoder_4_out_4, Decoder_4_out_5, Decoder_4_out_6, Decoder_4_out_7,
D6);
  Decoder8 #(3) Decoder_3(Decoder_3_out_0, Decoder_3_out_1, Decoder_3_out_2,
Decoder_3_out_3, Decoder_3_out_4, Decoder_3_out_5, Decoder_3_out_6, Decoder_3_out_7,
D3);
  assign or_3_out = Decoder_1_out_0 | Decoder_1_out_7 | Decoder_0_out_0 |
Decoder_0_out_7 | Decoder_3_out_0 | Decoder_3_out_7;
  Decoder8 #(3) Decoder_2(Decoder_2_out_0, Decoder_2_out_1, Decoder_2_out_2,
Decoder_2_out_3, Decoder_2_out_4, Decoder_2_out_5, Decoder_2_out_6, Decoder_2_out_7,
D4);
  Decoder8 #(3) Decoder_1(Decoder_1_out_0, Decoder_1_out_1, Decoder_1_out_2,
Decoder_1_out_3, Decoder_1_out_4, Decoder_1_out_5, Decoder_1_out_6, Decoder_1_out_7,
D1);
  Decoder8 #(3) Decoder_0(Decoder_0_out_0, Decoder_0_out_1, Decoder_0_out_2,
Decoder_0_out_3, Decoder_0_out_4, Decoder_0_out_5, Decoder_0_out_6, Decoder_0_out_7,
D2);
endmodule


module Decoder8(out0, out1, out2, out3, out4, out5, out6, out7, sel);
  output reg out0, out1, out2, out3, out4, out5, out6, out7;
  input [2:0] sel;


  always @ (*) begin
```

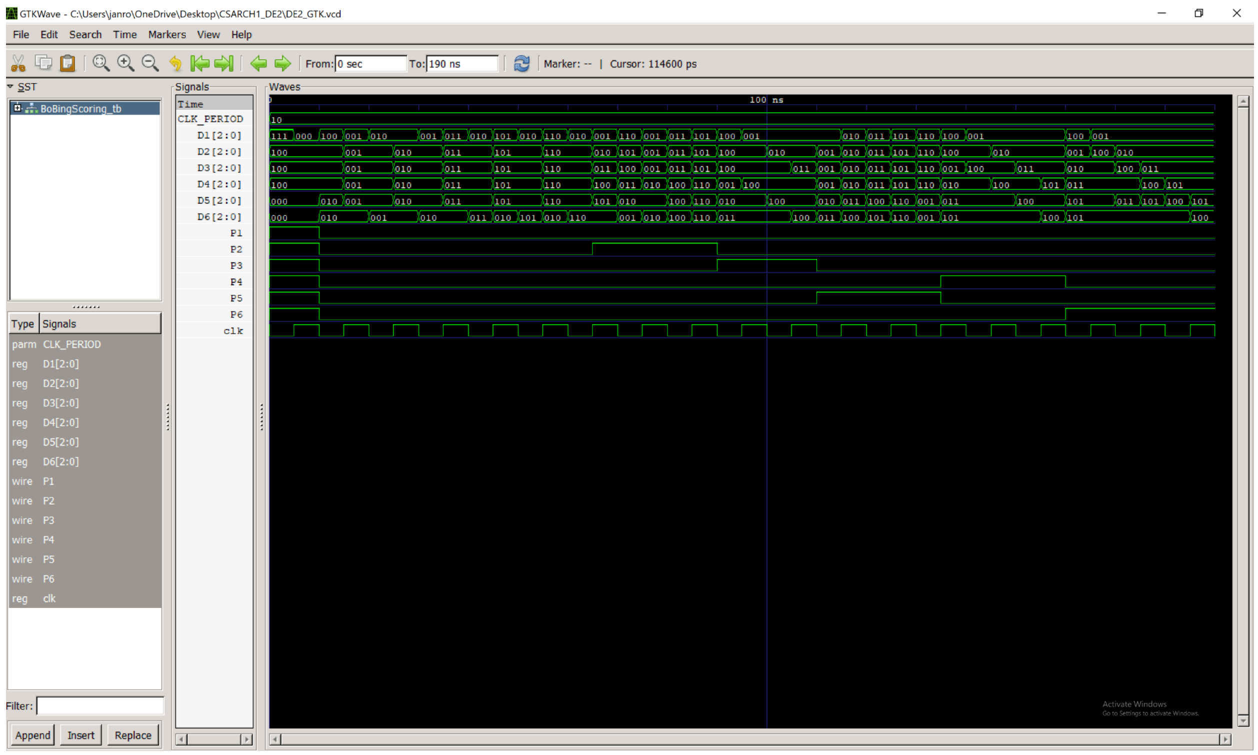
```
          out0 = 0;

          out1 = 0;

          out2 = 0;

          out3 = 0;

          out4 = 0;

          out5 = 0;

          out6 = 0;

          out7 = 0;

          case (sel)

          0 : out0 = 1;

          1 : out1 = 1;

          2 : out2 = 1;

          3 : out3 = 1;

          4 : out4 = 1;

          5 : out5 = 1;

          6 : out6 = 1;

          7 : out7 = 1;

          endcase

  end

endmodule


module Multiplexer8(out, in0, in1, in2, in3, in4, in5, in6, in7, sel);

  parameter WIDTH = 1;
  output reg [WIDTH-1:0] out;

  input [WIDTH-1:0] in0, in1, in2, in3, in4, in5, in6, in7;

  input [2:0] sel;


  always @ (*)

        case (sel)

        0 : out = in0;

        1 : out = in1;

        2 : out = in2;

        3 : out = in3;

        4 : out = in4;

        5 : out = in5;

        6 : out = in6;

        7 : out = in7;

        endcase

endmodule
```

Above are the modules of the exported Verilog file from CircuitVerse. The generated Verilog file also provides the number of gates used in the logic circuit. As can be seen below, there are 151 logic gates in total.



```
/*
  Element Usage Report
    Input - 55 times
    Decoder - 19 times
    Multiplexer - 1 times
    OrGate - 40 times
    DigitalLed - 14 times
    AndGate - 75 times
    Output - 17 times
    NotGate - 30 times
    SubCircuit - 31 times
    NandGate - 6 times
*/
```

WaveForms



GTKWave - C:\Users\janro\OneDrive\Desktop\CSARCH1_DE2\DE2_GTK.vcd

From: 0 sec To: 190 ns | Marker: -- | Cursor: 114600 ps

SST: BoBingScoring_tb

Type	Signals
parm	CLK_PERIOD
reg	D1[2:0]
reg	D2[2:0]
reg	D3[2:0]
reg	D4[2:0]
reg	D5[2:0]
reg	D6[2:0]
wire	P1
wire	P2
wire	P3
wire	P4
wire	P5
wire	P6
reg	clk

Reference List

A beginner's guide to the Mooncake dice game. (2022, September 29). Polland Hopia. https://pollandhopia.com/blogs/polland-hopia/a-beginners-guide-to-the-mooncake-dice-game