Bangladesh University of Engineering and Technology

Department of Electrical and Electronic Engineering

Dhaka-1205, Bangladesh

Course No: EEE 304

Course Name: Digital Electronics Laboratory

A Project on

Implementation of Snake and Ladder Game in Digital Board

Submitted By

1606099 - Sohan Salahuddin Mugdho

1606100 – Sadman Sakib

1606102 – Sadiah Ahmed Moon

Submitted To

Rajat Chakraborty
Assistant Professor
Department of EEE
BUET

Md. Irfan Khan
Lecturer
Department of EEE
BUET

Presentation Video

Table of Contents

SECTION 1

Introduction

Snakes and Ladders is an ancient Indian board game for two or more players regarded today as a worldwide classic. It is played on a game board with numbered, gridded squares. A number of "ladders" and "snakes" are pictured on the board, each connecting two specific board squares. The object of the game is to navigate one's game piece, according to die rolls, from the start to the finish, helped by climbing ladders but hindered by falling down snakes. Snake and Ladders game has been an important part of our childhood. Whether it is with friends or family, this game has shown popularity among people of all ages.

The game will be developed in two different software environments. At first, we will develop this game using Verilog codes in the Quartus software framework. We will also make this game using discrete IC components. We will design and simulate this circuit in the Proteus software environment. This can then later be made on a breadboard or on a printed circuit board. The subsequent sections describe the rules of the game, the different implementation of the game, performance and a detailed discussion on the limitation and the scope for this project.

SECTION 2

How the game works

Each player starts with a token on the starting square. Players take turns rolling a single dice to move their token by the number of squares indicated by the dice roll. Tokens follow a fixed route marked on the game board which usually follows a track from the bottom to the top of the playing area, passing once through every square. If, on completion of a move, a player's token lands on the lower-numbered end of a "ladder", the player moves the token up to the ladder's higher-numbered square. If the player lands on the higher-numbered square of a "snake", the token must be moved down to the snake's lower-numbered square.

An important segment of this game is that each player must roll a 1 for their game to begin.

SECTION 3

Verilog Implementation

The Verilog implementation of the project was done in the Quartus software. We have used edaplayground.com for the convenience of displaying the output. The completecode in Verilog is presented in Figure 3.1.

```
module snake_and_ladder(p1,p2,p3,p4,start,clk,res);
output reg[7:0] p1,p2,p3,p4;
reg[3:0]rand;
input start,clk,res;
integer i;

task throw_dice;
  inout reg[7:0] p;
  output reg[3:0] rand;
  begin
  rand= 1+{$random}%6;
  $display("Advance %d blocks.",rand);
  p=p+rand;
  if(p>8'd100)
    p=p-rand;
  $display("Your are currently at %d.",p);
end
endtask

task update_pos;
  inout reg[7:0] p;
  begin
    case(p)
      8'd99:begin
             p=8'd6;
            $display("You fell on a snake, you will be demoted to position
%d.",p);
          end
      8'd88:begin
             p=8'd67;
            $display("You fell on a snake, you will be demoted to position
%d.",p);
          end
     //lader
      8'd42:begin
            p= 8'd81;
            $display("You found a ladder, you will be promoted to position
%d.",p);
          end
      8'd15:begin
            p= 8'd97;
            $display("You found a ladder, you will be promoted to position
%d.",p);
          end

        endcase
      end
    endtask

always@(posedge clk)
begin
  if (res)
    begin
```

```
    p1=8'd0;
    p2=8'd0;
    p3=8'd0;
    p4=8'd0;
    i=1;
  end
end
always@(posedge clk)
begin
 $display("Current positions are %d %d %d %d.",p1,p2,p3,p4);
  if(start)
  begin
  if(i==1)
    begin
  $display("Player 1's turn.");
 throw_dice(p1,rand);
 update_pos(p1);
  if(p1==8'd100)
    begin
      $display("Congrats Player 1, you are the winner!!!");
      $display("Other players are at p2 %d,p3 %d,p4 %d.",p2,p3,p4);
      #5 $finish;
    end
    i=i+1;
  end
else if(i==2)
    begin
    $display("Player 2's turn.");
  throw_dice(p2,rand);
  update_pos(p2);
  if(p2==8'd100)
    begin
      $display("Congrats Player 2, you are the winner!!!");
      $display("Other players are at p1 %d,p3 %d,p4 %d.",p1,p3,p4);
      #5 $finish;
    end
    i=i+1;
  end
else if(i==3)
    begin
  $display("Player 3's turn.");
  throw_dice(p3,rand);
  update_pos(p3);
  if(p3==8'd100)
    begin
      $display("Congrats Player 3, you are the winner!!!");
      $display("Other players are at p1 %d,p2 %d,p4 %d.",p1,p2,p4);
      #5 $finish;
    end
    i=i+1;
  end
 else if(i==4)
    begin
  $display("Player 4's turn.");
  throw_dice(p4,rand);
  update_pos(p4);
  if(p4==8'd100)
    begin
      $display("Congrats Player 4, you are the winner!!!");
      $display("Other players are at p1 %d,p2 %d,p3 %d.",p1,p2,p3);
```

```
      #5 $finish;
    end
    i=1;
   end
 end
end
endmodule

//Testbench
module testbench();
  wire[7:0] p1,p2,p3,p4;
  reg start,clk,res;
  snake_and_ladder s1(p1,p2,p3,p4,start,clk,res);
  initial
  begin

     clk=1'b1;
     res=1'b1;
     start=1'b1;
    #1 res=1'b0;

   end
   always

    #1 clk=~clk

endmodule
```

Figure 3.1: The complete code for the project

3.1 Throw Dice

The first task of the game is to determine the value of the dice. We use the $random function for generating an arbitrary value of the dice from 1 to 6. This value will later determine the position of each player on the game board.

```
task throw_dice;
  inout reg[7:0] p;
  output reg[3:0] rand;
  begin
  rand= 1+{$random}%6;
  $display("Advance %d blocks.",rand);
  p=p+rand;
  if(p>8'd100)
    p=p-rand;
  $display("Your are currently at %d.",p);
end
endtask
```

Fig 3.2: Code for the dice value

3.2 Snake or Ladder Detection

On this task function we determine, whether there is a snake or a ladder on the player's current position. For a fixed structure of game board, the position of the snake and ladder is unchanged. Therefore, testing the current position of the player will confirm the presence of a snake or a ladder. If either is identified, the placement will be updated accordingly.

```
task update_pos;
  inout reg[7:0] p;
  begin
    case(p)
      8'd99:begin
              p=8'd6;
            $display("You fell on a snake, you will be demoted to position %d.",p);
          end
      8'd88:begin
              p=8'd67;
            $display("You fell on a snake, you will be demoted to position %d.",p);
          end
     //lader
      8'd42:begin
            p= 8'd81;
            $display("You found a ladder, you will be promoted to position %d.",p);
          end
      8'd15:begin
            p= 8'd97;
            $display("You found a ladder, you will be promoted to position %d.",p);
          end

        endcase
      end
    endtask
```

```
Player 3's turn.
Advance  4 blocks.
Your are currently at  15.
You found a ladder, you will be promoted to position  97.
Current positions are   6  18  97  18.
```

Fig 3.3: Snake or Ladder detection

3.3 Calculating Players' Position

Initially, all the players will be in position 0 and each will take a turn to throw the dice and upgrade their position in every round in the sequence player 1 > player 2 > player 3 > player 4

```
47 always@(posedge clk)
48 begin
49   if (res)
50     begin
51     p1=8'd0;
52     p2=8'd0;
53     p3=8'd0;
54     p4=8'd0;
55     i=1;
56   end
57 end
58 always@(posedge clk)
59 begin
60  $display("Current positions are %d %d %d %d.",p1,p2,p3,p4);
```

Fig 3.4: Initial position of the players

When it is the turn of player 1, the throw_dice task will generate an arbitrary dice value from 1 to 6. The position will be checked by the update_pos task in order to determine if there is any snake or ladder on the respective position. The final placement of the player will be determined and shown in the display. The placements of the rest of the players will be unchanged at this stage.

```
59 begin
60  $display("Current positions are %d %d %d %d.",p1,p2,p3,p4);
61   if(start)
62   begin
63   if(i==1)
64     begin
65   $display("Player 1's turn.");
66  throw_dice(p1,rand);
67  update_pos(p1);
68   if(p1==8'd100)
69     begin
70       $display("Congrats Player 1, you are the winner!!!");
71       $display("Other players are at p2 %d,p3 %d,p4 %d.",p2,p3,p4);
72       #5 $finish;
73     end
74     i=i+1;
75   end
```

Fig 3.5: Player 1's turn

Once player 1's turn is over, all the other players will take their turn in the game sequentially.

```
Player 1's turn.
Advance  4 blocks.
Your are currently at   7.
Current positions are   7   4   2   6.
Player 2's turn.
Advance  6 blocks.
Your are currently at  10.
Current positions are   7  10   2   6.
Player 3's turn.
Advance  6 blocks.
Your are currently at   8.
Current positions are   7  10   8   6.
```

Fig 3.6: Each players' turn and position update

SECTION 4

Hardware Implementation for Single Player

The hardware implementation for single player of this project was designed and simulated in the Proteus environment. The project can be divided into different modules that perform separate task. Each of these modules were designed separately and then connected together to produce the final working circuit. The full circuit is displayed in Figure 4.1 and the individual modules are described in details in the following sub sections.

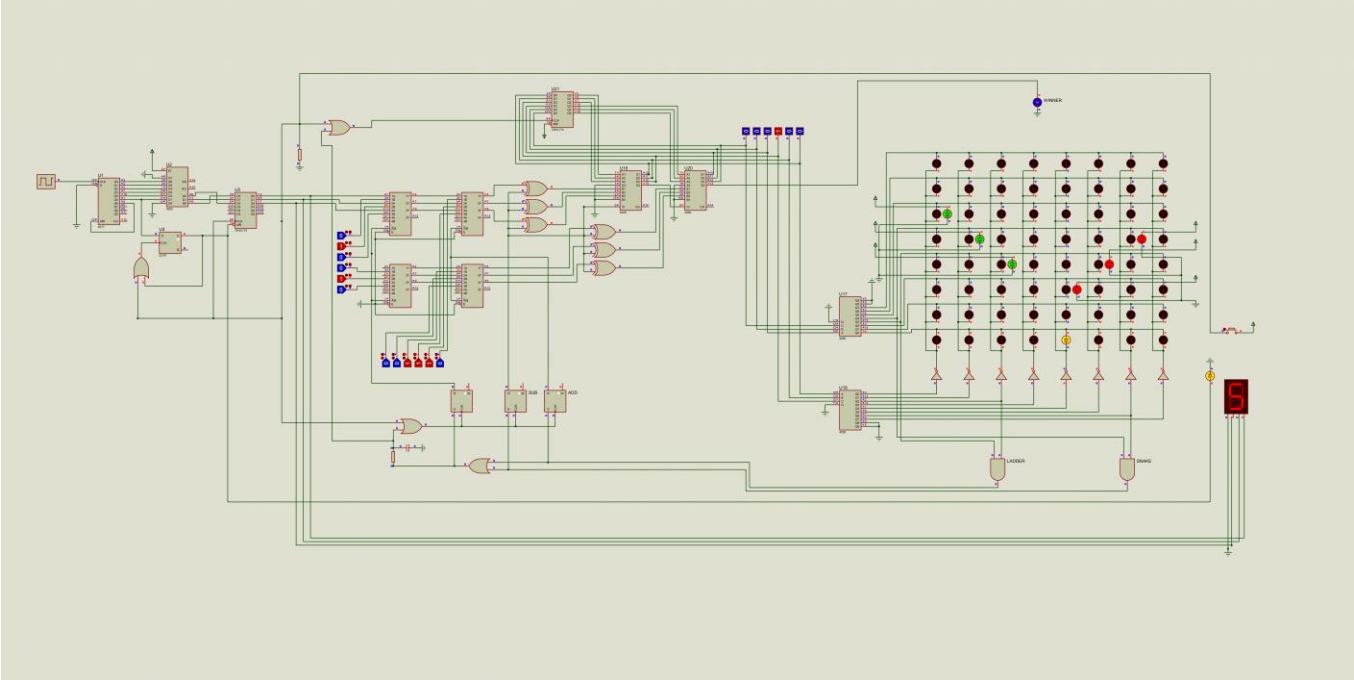


Figure 4.1: Full circuit implementation of the project in Proteus

4.1 Generating the value of the dice

The very first segment of the game includes deciding the value of the dice. Here we have designed a system that generates an arbitrary value of the dice from 1 to 6 every time the play button is pushed.

Here we used a Johnson Decade Counter, 8-bit Priority Encoder and D flip-flops to determine the value of the dice. The input is generated from the signal generator, passed through the ICs and gives an output. However, the DTFF IC ensures that the player must generate 1 for the game to begin. The detailed circuit is presented in Figure 4.2.

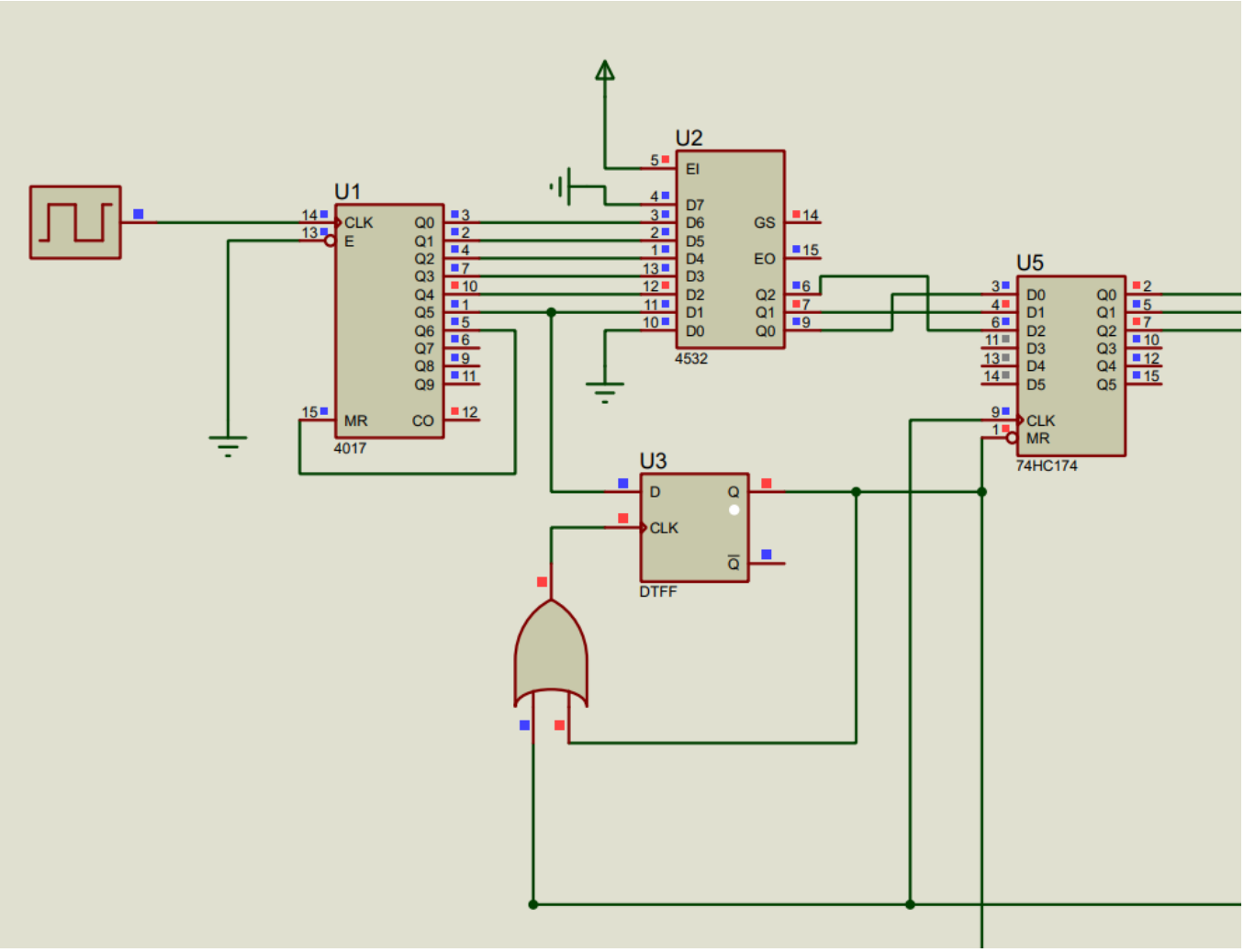


Figure 4.1: Generating the value of dice

4.2 Detecting Snake or Ladder

The detection of snake and ladder is the most important part of the game. If the player encounters a snake or a ladder on the game board the AND gate provides an output 1. The D flip-flops used in the circuit determines whether it is a snake or a ladder and send the output to the multiplexers.

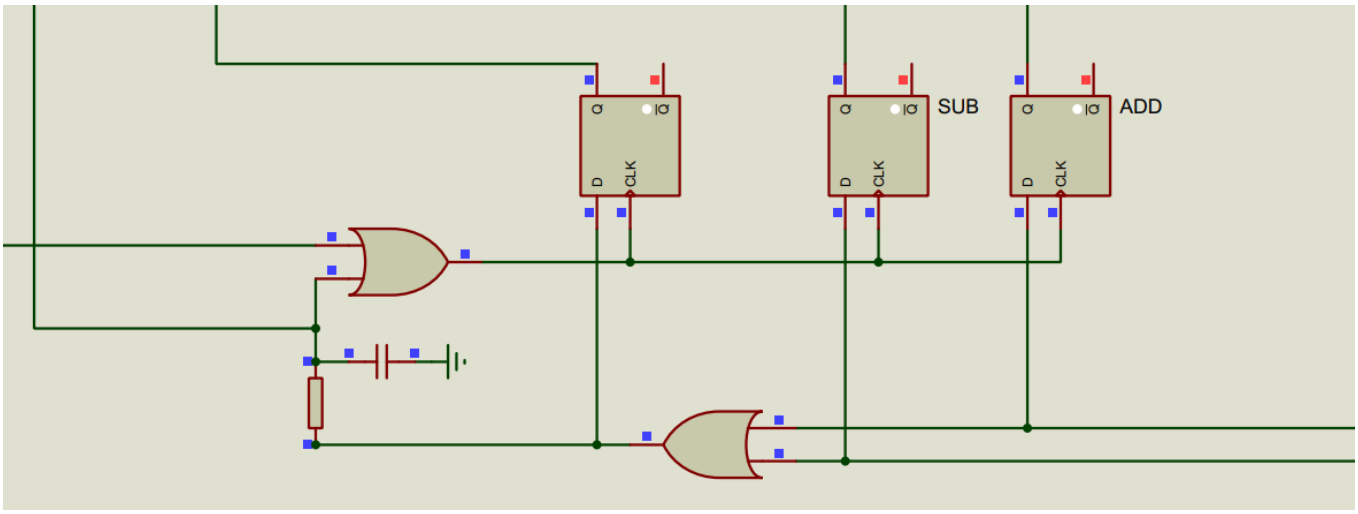


Figure 4.2: Determining snake or ladder

Here we have used Quadruple 2-1 multiplexers and XOR gates to calculate the updated position of the player. If the output of the previous D flip-flops detect Ladder, the multiplexer will provide the output of the current new position on the game-board. Eventually, the updated position will be added with the dice value and the player will go up the ladder.

On the other hand, if the previous portion of the circuit detects Snake, the input of the multiplexer will be the new-found position at the bottom of the snake. As a result, the output will be converted into 2's complement through the XOR gates and eventually be added (originally subtracted from) with the current dice value. As a result the player will go down the game board.

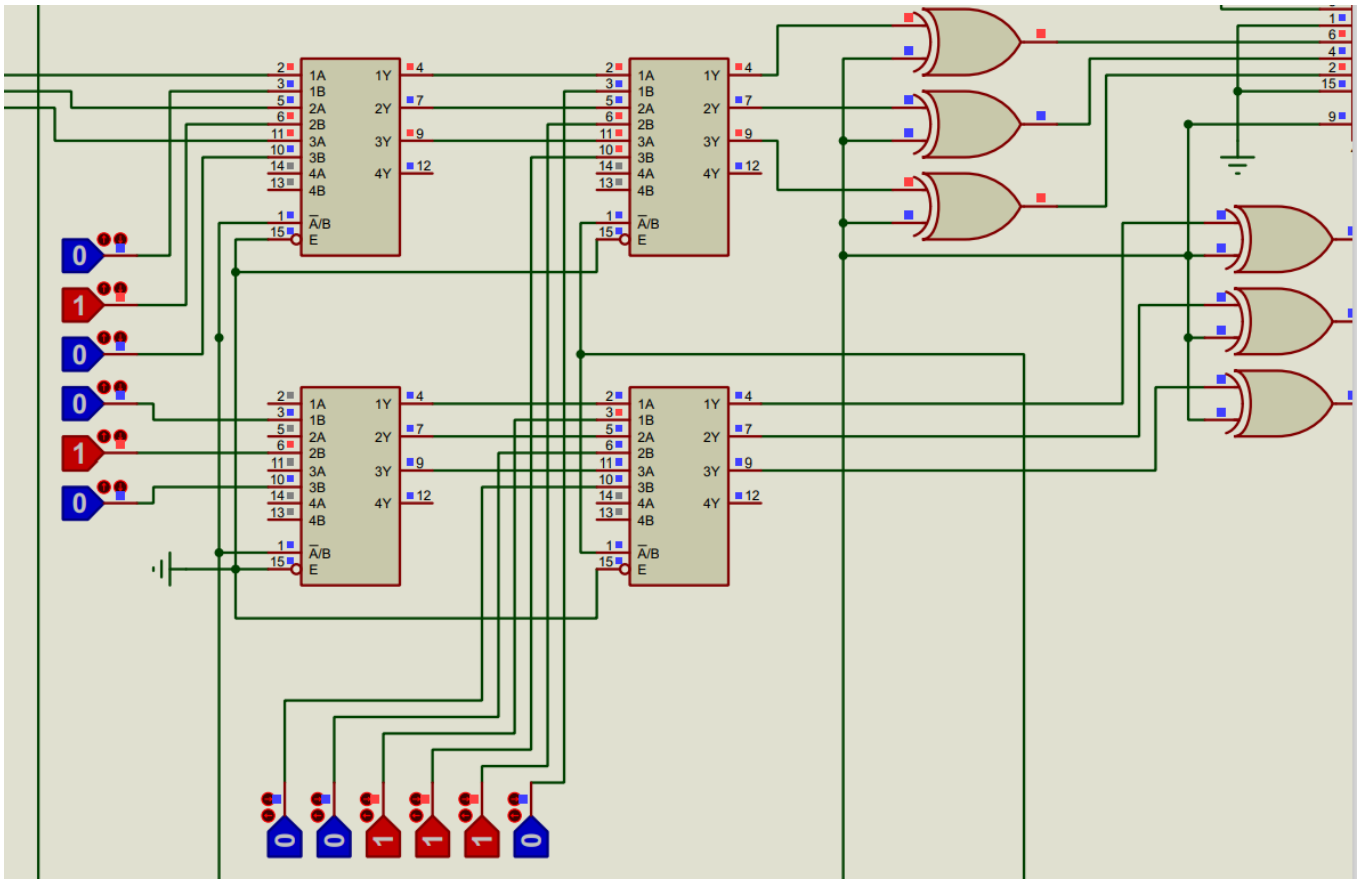


Figure 4.3: Addiction or Subtraction due to ladder or snake

4.3 Final Position of the Player

Four bit full adder and D flip-flops are used for calculation the final position of the player. The input of the adder will be the current value of the dice and the previous output. In a snake and ladders game, the position of the player is upgraded from the previous position with each turn. That is why the D flip-flop is used to store the previous output position in order to use it as the input of the adder on the next turn.

Updated position = Previous position + Current Value of Dice

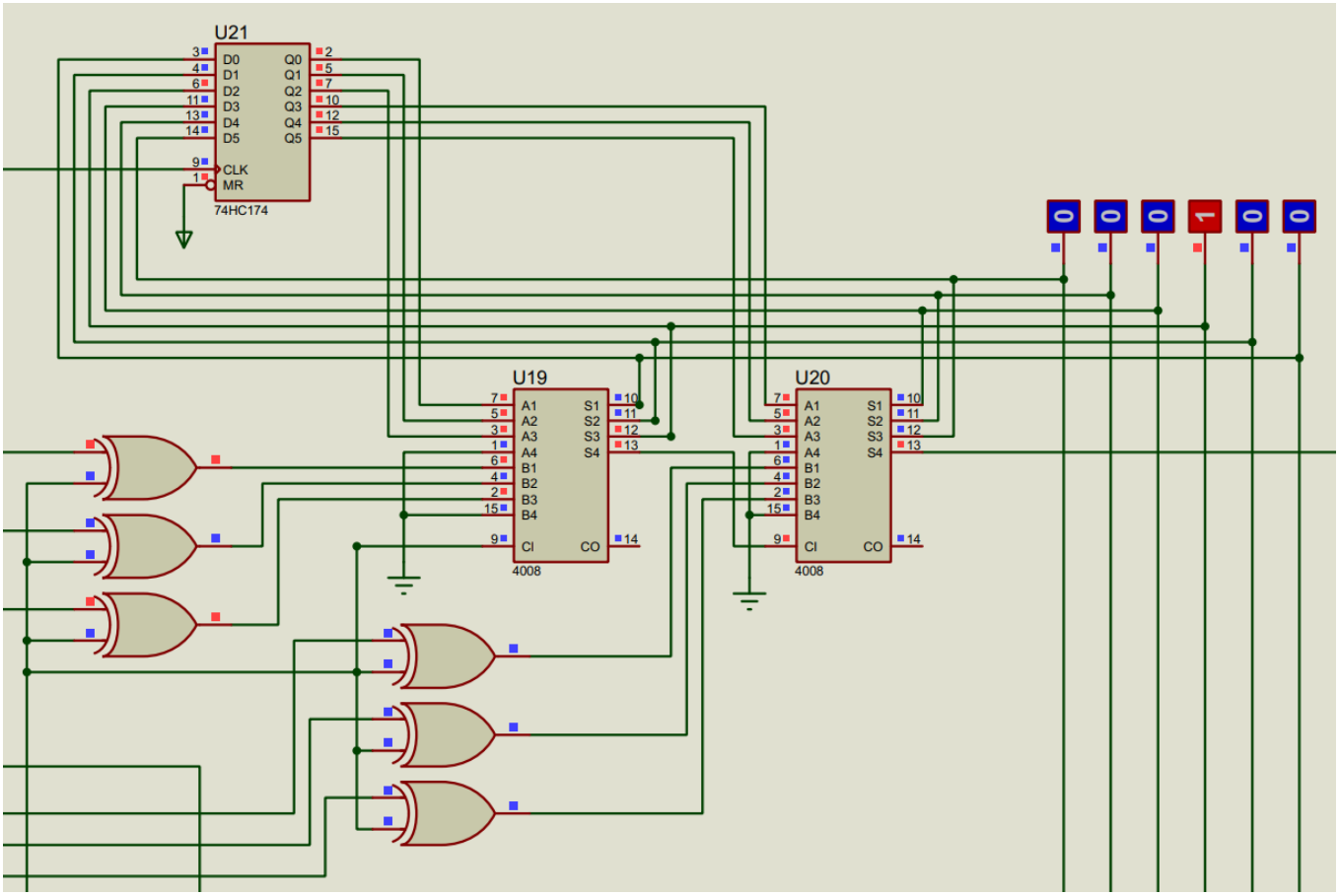


Figure 4.4: Calculating the final position

Once the final position is calculated, it undergoes the decoders and determine the exact row and column of the newfound place of the player in the game board. Here, the game board is designed in a manner that it has 64 square grids. Therefore, the number of bits that has been handled throughout the entire process is 6. If the player reaches the top right corner of the board, he is declared as the winner.

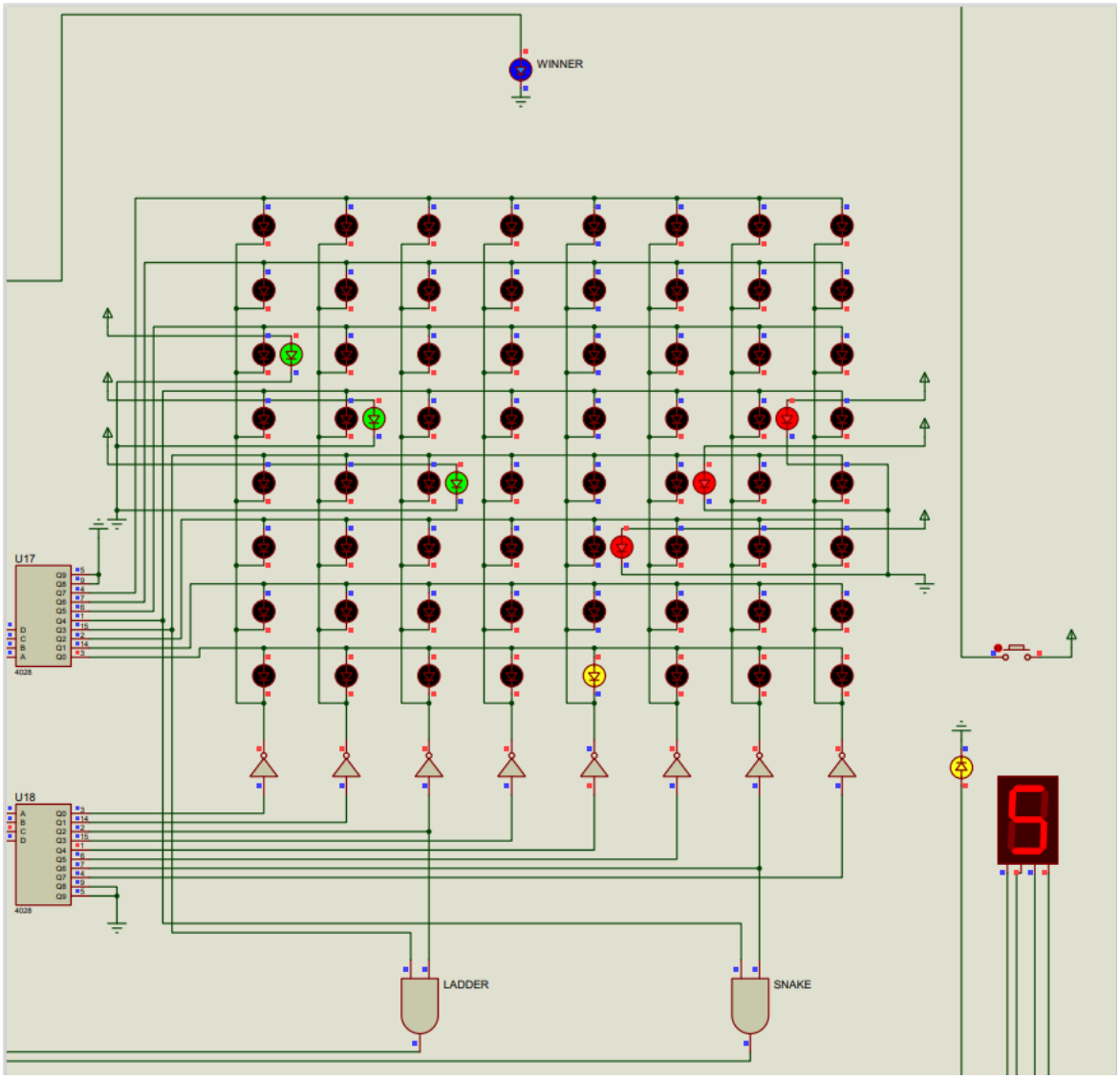


Figure 4.5: The game board

SECTION 5

Hardware Implementation for Multiplayer

We have also designed a version of the game for three players in the Proteus environment. It also happens to be divided into different modules to perform different tasks. Each module is explained in details in the upcoming sections.

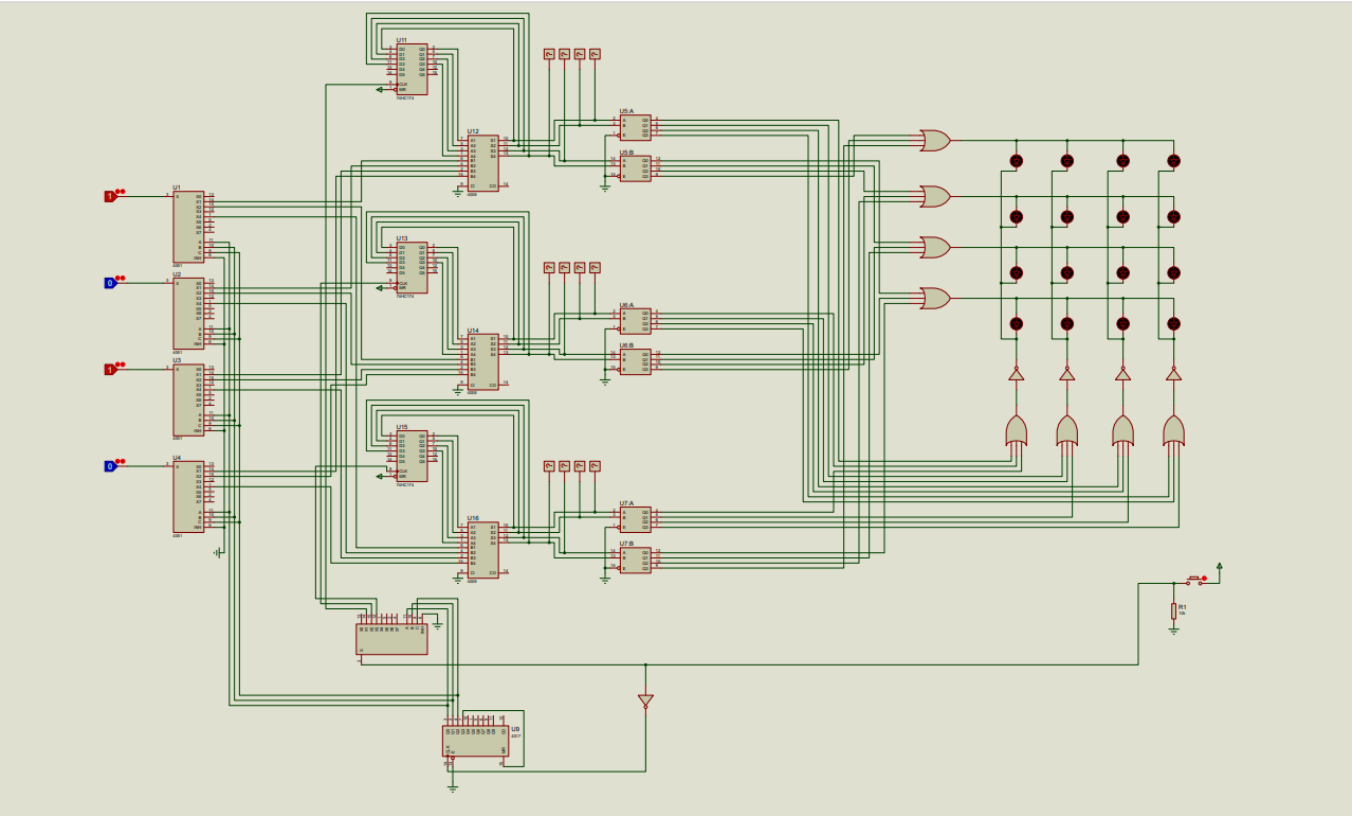


Figure 5.1: Hardware implementation for multiplayer

5.1 Determining the Dice Value and Calculating the Positions

For this part, we will be updating the value of the dice manually. The calculation of the position of each player is similar to that of the circuit for single player. The output dice value will be the input of the 4-bit full adder. The other input of the adders will be the previous position of the player that is already stored using the D Flip-flops. There are three sets of adders, D flip-flops and decoders in order to accommodate three players.

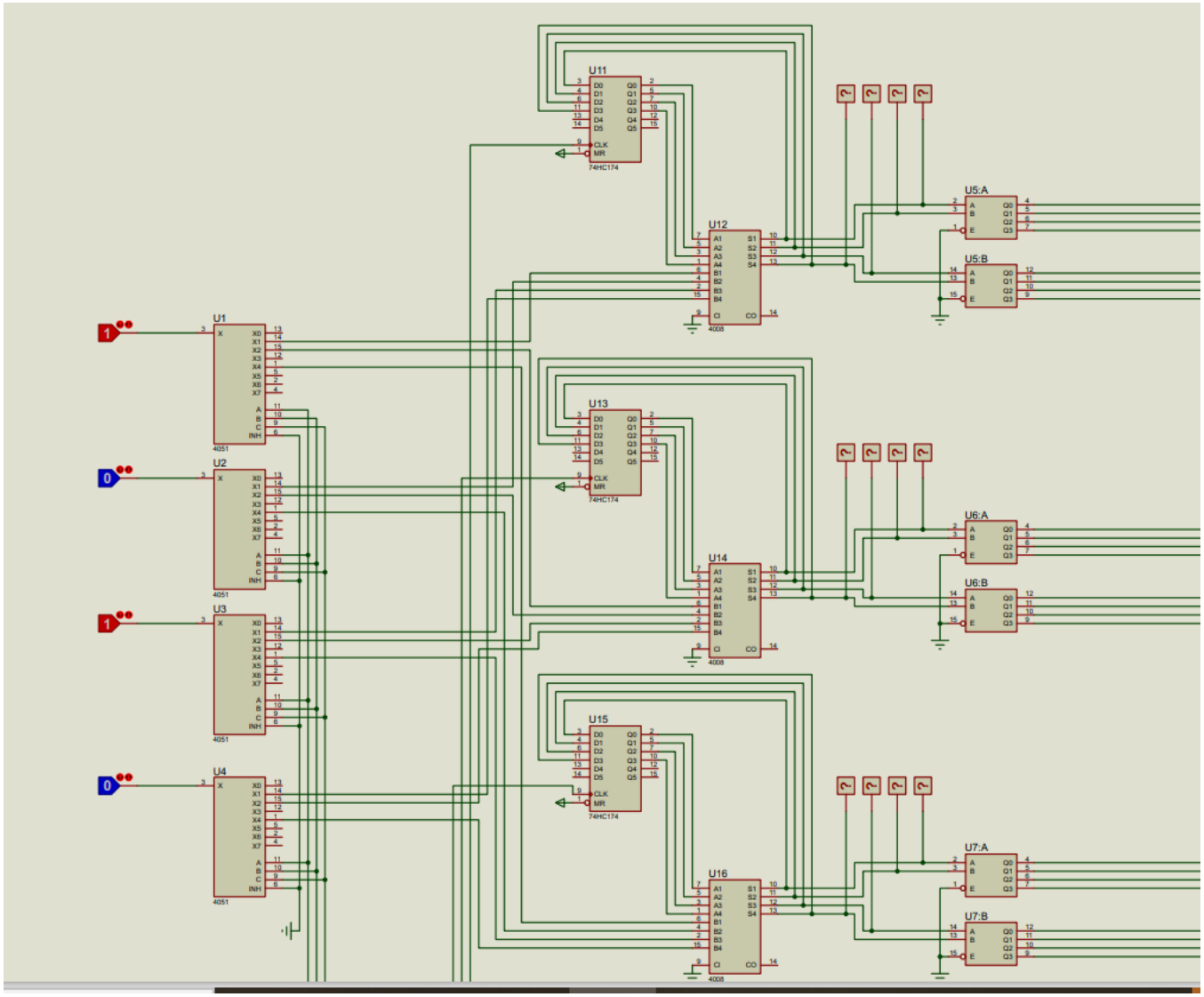
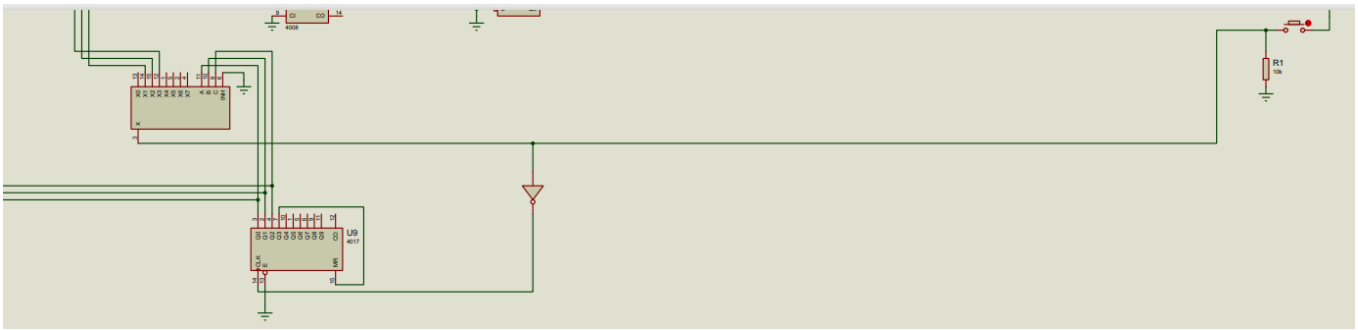


Figure 5.2: Determining the position of each player

5.2 Changing the Players

We have designed this circuit to accommodate three different players where the sequence of turns will be player 1 > player 2 > player 3. When, a player pushes the play button the output of the Johnson decade shifter determines which player's turn it is.



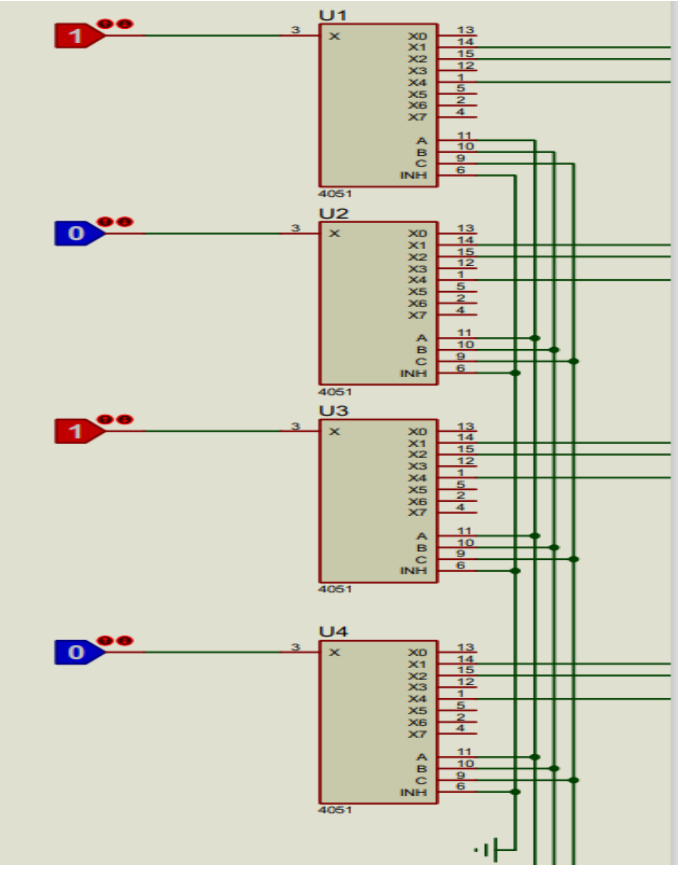


Figure 5.3: Determining the turn of each player

The output of the Ring shifter will determine which select pin of the multiplexers will receive the input 1. As a result, the output (the bits of the dice value) will be directed to the respective player's segment of calculation of the current position (5.1).

SECTION 6

Performance

This game was developed both in Verilog and Proteus environment and it was simulated and tested separately in both these environments as well.

We have used edaplayground.com for the convenience of the display. Due to the Testbench part of the code the values were randomized and automatic that saved us the hassle of manually putting the value every time. From the figure shown in 6.1, each player takes a turn one after another and is assigned a random dice value that changes their current positon.

```
Current positions are   x   x   x   x.
Current positions are   0   0   0   0.
Player 1's turn.
Advance  3 blocks.
Your are currently at   3.
Current positions are   3   0   0   0.
Player 2's turn.
Advance  4 blocks.
Your are currently at   4.
Current positions are   3   4   0   0.
Player 3's turn.
Advance  2 blocks.
Your are currently at   2.
Current positions are   3   4   2   0.
Player 4's turn.
Advance  6 blocks.
Your are currently at   6.
Current positions are   3   4   2   6.
```

Figure 6.1: Updating players' position

If a player come across a snake or ladder, the output is shown in figure 6.2.

```
Player 1's turn.
Advance  4 blocks.
Your are currently at  15.
You found a ladder, you will be promoted to position  97.
Current positions are  97  12   9  17.
```

Figure 6.2: Snake or ladder calculation

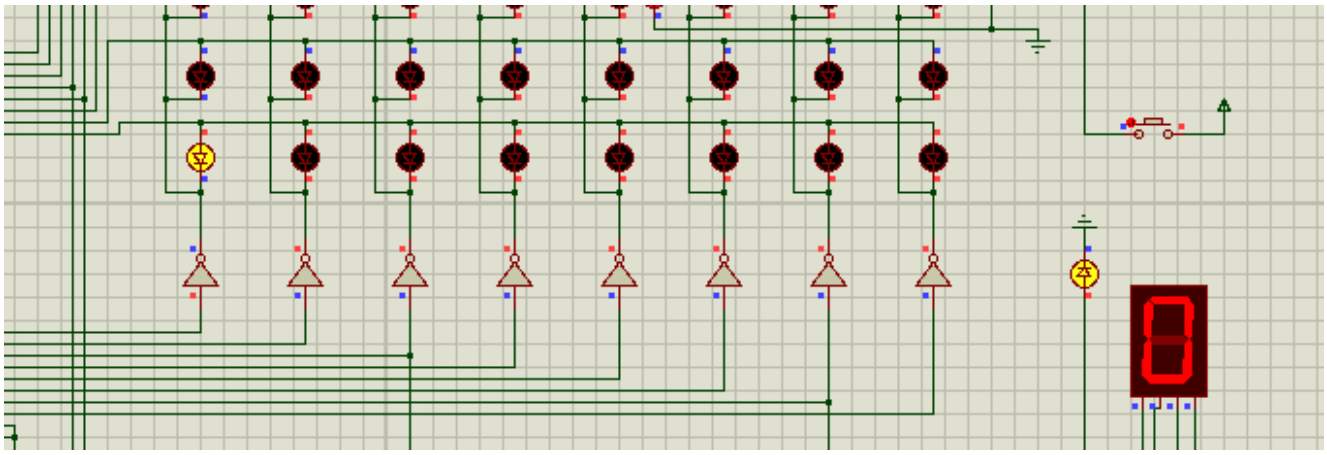
Eventually, when one of the player reaches the end of the game board and is declared the winner, the game ends.

```
Player 3's turn.
Advance  2 blocks.
Your are currently at 100.
Congrats Player 3, you are the winner!!!
Other players are at p1  16,p2  26,p4  29.
Done
```

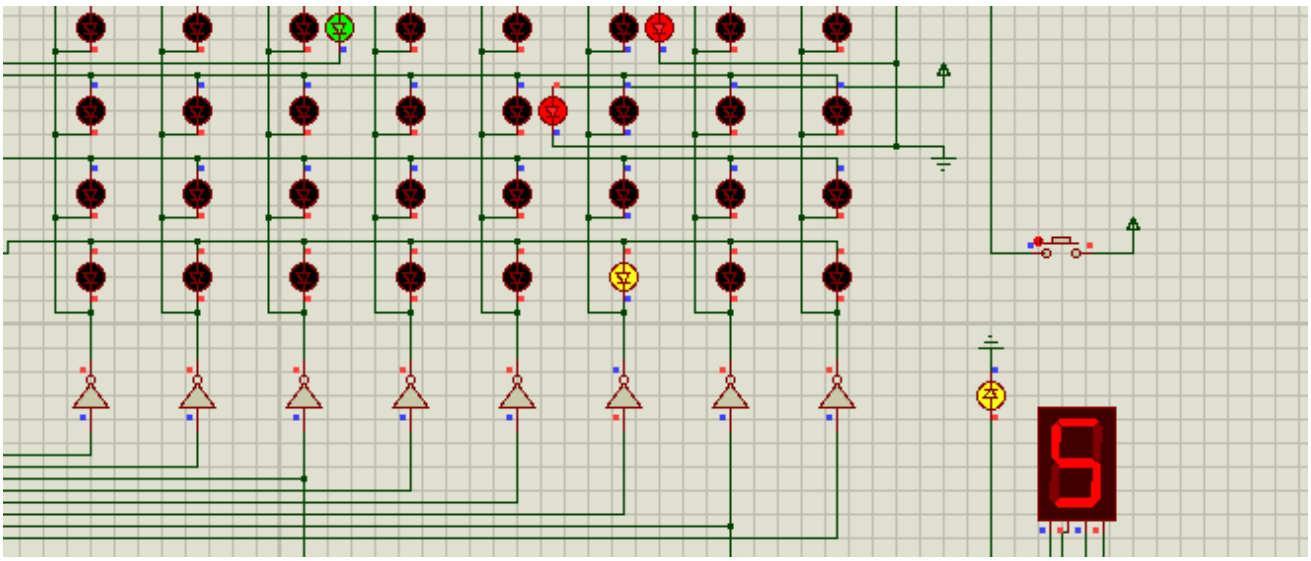
Figure 6.3: Winner declaration

For the hardware implementation, we simulated the circuit developed in the previous section and played a game. The game started after a couple of tries as the initial value of the dice required to be 1. Once the dice output became 1, the game ran smoothly till the end.

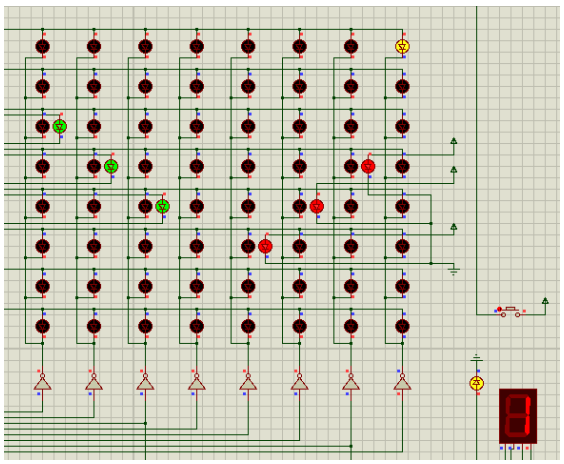
1) The following picture indicates that the value of dice is 1. Therefore, the player is in the bottom left position.

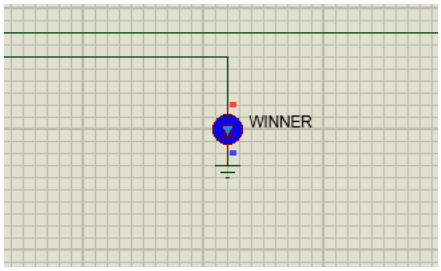


2) On the next turn, the dice value was 5. Therefore, the player's position shifted five times to the right.



3) If we keep playing in this manner, eventually the player will reach the ending of the game board and be declared a winner.





SECTION 7

Limitations

Even though the game performed accurately in both the setup there are still some factors in the designed circuit, that limits the user experience and there are room for improvement. In the software part, the requirement of throwing 1 at the very beginning was not specified. As a result, the game will start at any random value that violates one of the fundamental rules of the game. Improvement of that segment can be made with further approach.

In the first hardware implementation, the game is designed for only one player. If there is more than one player, the system cannot perform. A multiplayer segment of that portion can be designed to keep the original competitiveness of the game intact.

On the multiplayer segment, only the position change of each player was shown. However, no snake or ladder has been added to the game board or the system yet. Combining the snake-ladder identification module from the single player system with the multiplayer system will result in a competitive and accurate Snake and Ladders game.

SECTION 8

Conclusion

This project was developed with the aim of bringing a game played board of token and dice to the digital platform. We have been able to successfully implement the project using discrete IC components and as well as in Verilog. The circuit components and the coding modules in the project can be scaled up to accommodate more players and play for larger game board with more complicated chains of snakes and ladders. We believe the ideas and methods used in this project can be exploited further to make other projects with real life impacts a reality.

SECTION 9

Contributions:

Contributions by 1606099: Implemented the snake and ladder detection and reward/penalty system and the multiplayer demonstration in hardware.
Contributions by 1606100: Wrote the main game code and the testbench code for Verilog code implementation.
Contributions by 1606102: Implemented the dice value generating and player position update systems and the LED matrix in hardware and prepared the report.