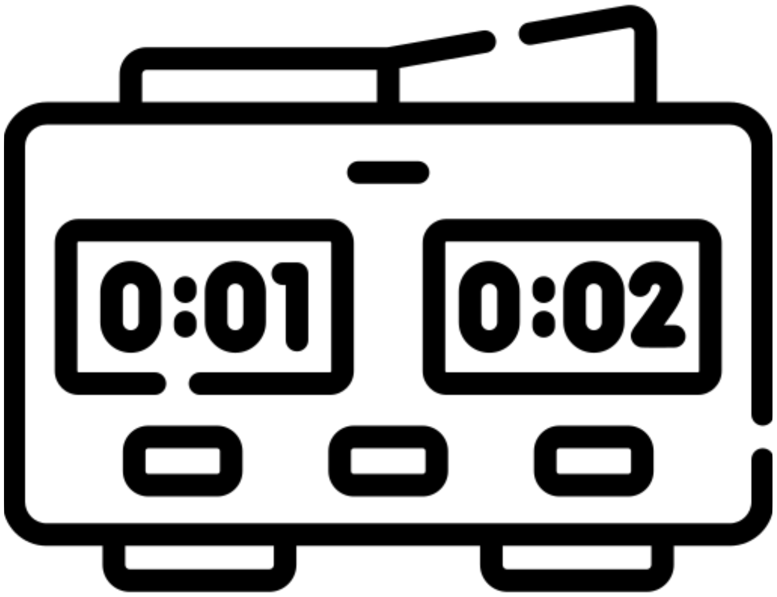
The Chess Clock



Team members:

1. 221CS239, Abhyuday Rayala, rayalaabhyuday.221cs239@nitk.edu.in , 7013831726

2. 221CS235, Pramod Chaitanya Dandu, pramodchaitanya.221cs235@nitk.edu.in , 6305451581

3. 221CS230, Manohar Rohit Vijay, rohitvijaymanohar.221cs230@nitk.edu.in , 8767014074

Abstract:

Development of a Digital Chess Clock System

Chess, a timeless strategy game, has long relied on analog chess clocks for timed play. As the game evolves in the digital age, there is a growing need for a modern, versatile, and user-friendly digital chess clock system.

The aim of this mini project, "Development of a Digital Chess Clock System," is to address the shortcomings of traditional chess clocks and provide a solution that enhances the chess playing experience. With the advent of technology, we seek to offer a convenient and feature-rich alternative for chess enthusiasts and tournament organizers.

This project introduces a state-of-the-art digital chess clock system that combines precision, usability, and flexibility. The key contributions include:

User-Friendly Interface: Our system features an intuitive LCD interface with easily accessible control buttons, ensuring straightforward operation for players of all levels.

Customizable Time Controls: Chess players can tailor time settings to suit their preferences, incorporating delay and increment options for added tactical depth.

Dual-Clock Mode: The system supports competitive play with dual clocks, making it ideal for tournament use.

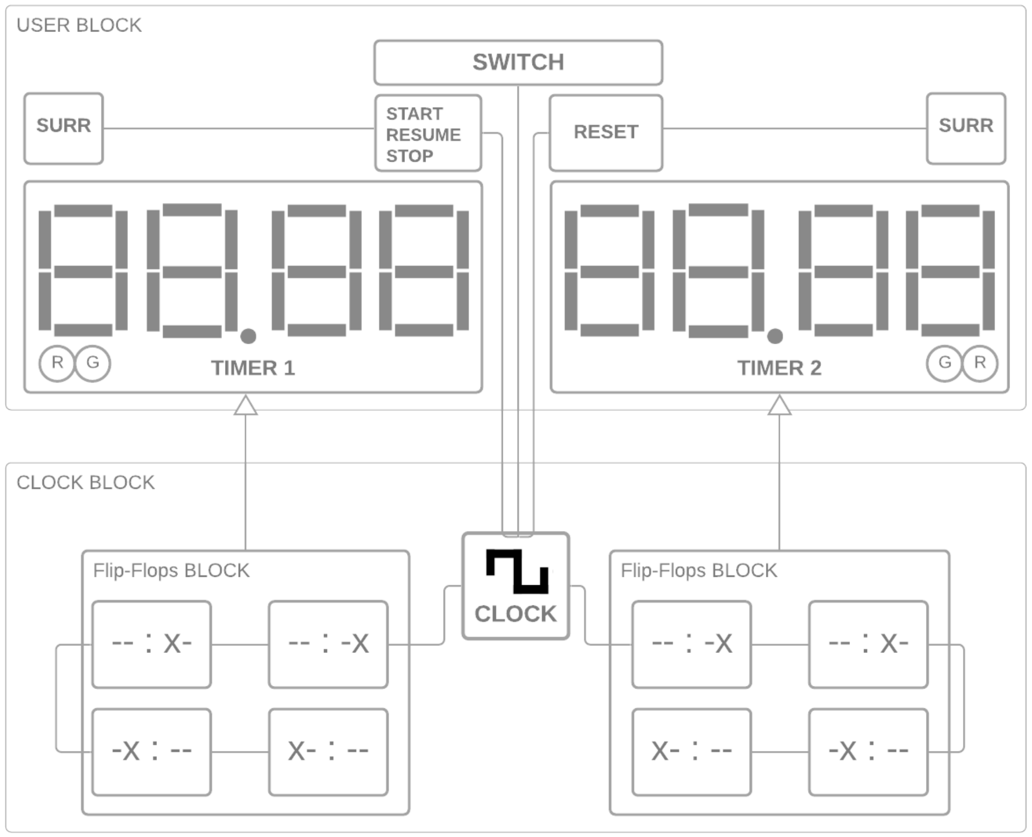
Accessibility Features: To promote inclusivity, we have integrated audio cues, making the system accessible to visually impaired players.

Win probability : The system predicts the winning probability based on the time gap between two players. Adding onto this a time difference is created for players who has lesser rating, so that weaker one gets extra time to think. For this to decide, Individuals Rating is taken into account.Our project not only modernizes the traditional chess clock but also serves as a potential solution for other strategy-based board games. It offers a comprehensive and innovative tool that not only caters to the evolving needs of the chess community but also paves the way for the integration of digital technology into traditional board gaming

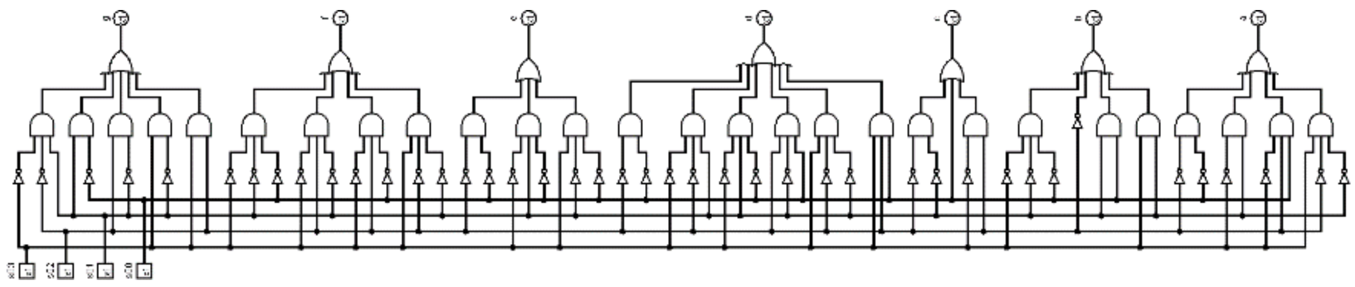
Description:

A digital clock designed for chess games. It mainly consists of two timers which cannot count at the same time. The source clock is alternatively transmitted between the two down counters using a switcher

The following is a schematic diagram of the idea behind the implementation of this clock



7 segment IC



Working And Features:

The Digital Chess clock consists of the following features

1. **Game Start**: When the 'start' button is pressed, the clock transitions to player 1's turn. Their timer starts counting down, and the other player's timer is paused.
2. **Player Turn**: During each player's turn, their timer counts down while the opponent's timer remains paused. Players can switch the timers by pressing the 'switch' button. This feature is useful in games where players take turns alternately.
3. **Surrender**: If a player decides to surrender by pressing the 'surrender' button, their timer stops, and their opponent is declared the winner. The losing player's timer displays a red light, while the winning player's timer displays a green light, signalling the game outcome.
4. **Reset** : This option resets the timer of both the players. To start a new match or to end up the match, this can be used.

This digital chess clock ensures fair play by limiting the time available for each move and provides a visual indication of the game's progress, making it a valuable tool for competitive chess matches.

This clock basically work under various components that includes counters, 7-seg Display's, basic input gates , bulbs etc.

Operation Mode:

	Stop/ Start	Nex t	Res et	0 (zero)	Surrend er
Start the game	1	x	0	0	0
Stop the game	0	x	0	0	0
Resume the game	1	x	0	0	0
Bob's turn	1	0	0	0	0
Alice's turn	1	1	0	0	0
Surrender – Finish the game	x	x	x	x	1
Reset the timers	1	x	1	1	0

Functioning:

Clock Logic

- The main clock feeds both timers with pulses periodically.
- It is designed in such a way that if any one of the player surrenders, then it stops giving pulse to the timers.

Reset Logic

- Moving to the next state by passing only one pulse Must push on 0 first

Stop/Start Logic

Stop

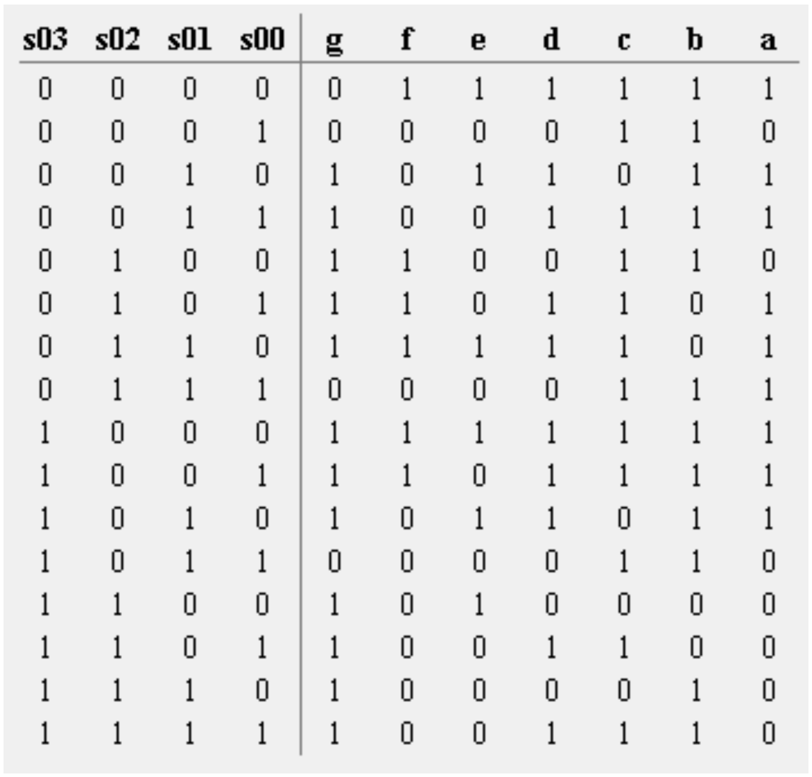
When Stop/Start pin is set to 0, Which is connected to surrender and clock to simulate the required timer

Start

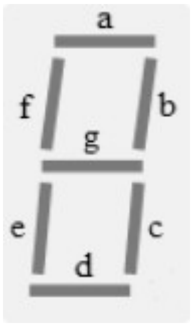
When Stop/Start pin is set to 1, Clock gate get supplied by null voltage, hence activation of one of thetimers.

Timer Logic

7-seg Display



s03	s02	s01	s00	g	f	e	d	c	b	a
0	0	0	0	0	1	1	1	1	1	1
0	0	0	1	0	0	0	0	1	1	0
0	0	1	0	1	0	1	1	0	1	1
0	0	1	1	1	0	0	1	1	1	1
0	1	0	0	1	1	0	0	1	1	0
0	1	0	1	1	1	0	1	1	0	1
0	1	1	0	1	1	1	1	1	0	1
0	1	1	1	0	0	0	0	1	1	1
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	0	1	1	1	1
1	0	1	0	1	0	1	1	0	1	1
1	0	1	1	0	0	0	0	1	1	0
1	1	0	0	1	0	1	0	0	0	0
1	1	0	1	1	0	0	1	1	0	0
1	1	1	0	1	0	0	0	0	1	0
1	1	1	1	1	0	0	1	1	1	0



Surrender Logic

When surrender pin is set to 1, It is in connection with start button, finally resulting in stoppage of the timer and declaring the winner with green light.

Verilog Code

```
module chess_clock(
   input wire clk, // Clock input
   input wire reset, // Reset signal for timers
   input wire start, // Start signal for timers
   input wire surrender_player1, // Surrender button for player 1
   input wire surrender_player2, // Surrender button for player 2
   input wire switch_turn, // Switch turn signal
   output reg [6:0] seg_player1_min1, // 7-segment display for player 1 minutes (tens)
   output reg [6:0] seg_player1_min0, // 7-segment display for player 1 minutes (units)
```

```
    output reg [6:0] seg_player1_sec1, // 7-segment display for player 1 seconds (tens)

    output reg [6:0] seg_player1_sec0, // 7-segment display for player 1 seconds (units)

    output reg [6:0] seg_player2_min1, // 7-segment display for player 2 minutes (tens)

    output reg [6:0] seg_player2_min0, // 7-segment display for player 2 minutes (units)

    output reg [6:0] seg_player2_sec1, // 7-segment display for player 2 seconds (tens)

    output reg [6:0] seg_player2_sec0, // 7-segment display for player 2 seconds (units)

    output reg player1_green_led, // Green LED indicating player 1's victory

    output reg player2_green_led, // Green LED indicating player 2's victory

    output reg player1_red_led, // Red LED indicating player 1's loss

    output reg player2_red_led // Red LED indicating player 2's loss

);


    reg [3:0] min_ones_player1; // 4-bit register for player 1 minutes (units)

    reg [3:0] min_tens_player1; // 4-bit register for player 1 minutes (tens)

    reg [3:0] sec_ones_player1; // 4-bit register for player 1 seconds (units)

    reg [3:0] sec_tens_player1; // 4-bit register for player 1 seconds (tens)


    reg [3:0] min_ones_player2; // 4-bit register for player 2 minutes (units)

    reg [3:0] min_tens_player2; // 4-bit register for player 2 minutes (tens)

    reg [3:0] sec_ones_player2; // 4-bit register for player 2 seconds (units)

    reg [3:0] sec_tens_player2; // 4-bit register for player 2 seconds (tens)


    reg [1:0] player_turn; // 2-bit counter to track player turns (00: Player 1, 01: Player 2)


    always @(posedge clk or posedge reset) begin

        if (reset) begin

            min_ones_player1 <= 4'd0;

            min_tens_player1 <= 4'd0;

            sec_ones_player1 <= 4'd0;

            sec_tens_player1 <= 4'd0;
        min_ones_player2 <= 4'd0;
        min_tens_player2 <= 4'd0;
        sec_ones_player2 <= 4'd0;
        sec_tens_player2 <= 4'd0;

        player_turn <= 2'b00;

        seg_player1_min1 <= 7'b0000000; // Initialize 7-segment displays to 0
        seg_player1_min0 <= 7'b0000000;
        seg_player1_sec1 <= 7'b0000000;
        seg_player1_sec0 <= 7'b0000000;

        seg_player2_min1 <= 7'b0000000;
        seg_player2_min0 <= 7'b0000000;
        seg_player2_sec1 <= 7'b0000000;
        seg_player2_sec0 <= 7'b0000000;

        player1_green_led <= 1'b0;
        player2_green_led <= 1'b0;
        player1_red_led <= 1'b0;
        player2_red_led <= 1'b0;
    end
    else if (start) begin
        // Timer logic for player 1
        if (player_turn == 2'b00) begin
            if (sec_ones_player1 < 4'd9) begin
                sec_ones_player1 <= sec_ones_player1 + 1;
            end
            else if (sec_tens_player1 < 4'd5) begin
                sec_ones_player1 <= 4'd0;
                sec_tens_player1 <= sec_tens_player1 + 1;
            end
    else if (min_ones_player1 < 4'd9) begin
        sec_tens_player1 <= 4'd0;
        sec_ones_player1 <= 4'd0;
        min_ones_player1 <= min_ones_player1 + 1;
    end
    else if (min_tens_player1 < 4'd5) begin
        sec_tens_player1 <= 4'd0;
        sec_ones_player1 <= 4'd0;
        min_ones_player1 <= 4'd0;
        min_tens_player1 <= min_tens_player1 + 1;
    end
    else begin
        // Player 1 has won, set LEDs accordingly
        player1_green_led <= 1'b1;
        player2_red_led <= 1'b1;
    end
end
// Timer logic for player 2
else if (player_turn == 2'b01) begin
    if (sec_ones_player2 < 4'd9) begin
        sec_ones_player2 <= sec_ones_player2 + 1;
    end
    else if (sec_tens_player2 < 4'd5) begin
        sec_ones_player2 <= 4'd0;
        sec_tens_player2 <= sec_tens_player2 + 1;
    end
    else if (min_ones_player2 < 4'd9) begin
        sec_tens_player2 <= 4'd0;
        sec_ones_player2 <= 4'd0;
        min_ones_player2 <= min_ones_player2 + 1;
    end
    else if (min_tens_player2 < 4'd5) begin
```

```
            sec_tens_player2 <= 4'd0;
            sec_ones_player2 <= 4'd0;
            min_ones_player2 <= 4'd0;
            min_tens_player2 <= min_tens_player2 + 1;
          end
          else begin
            // Player 2 has won, set LEDs accordingly
            player1_red_led <= 1'b1;
            player2_green_led <= 1'b1
  endmodule
```

test bench:

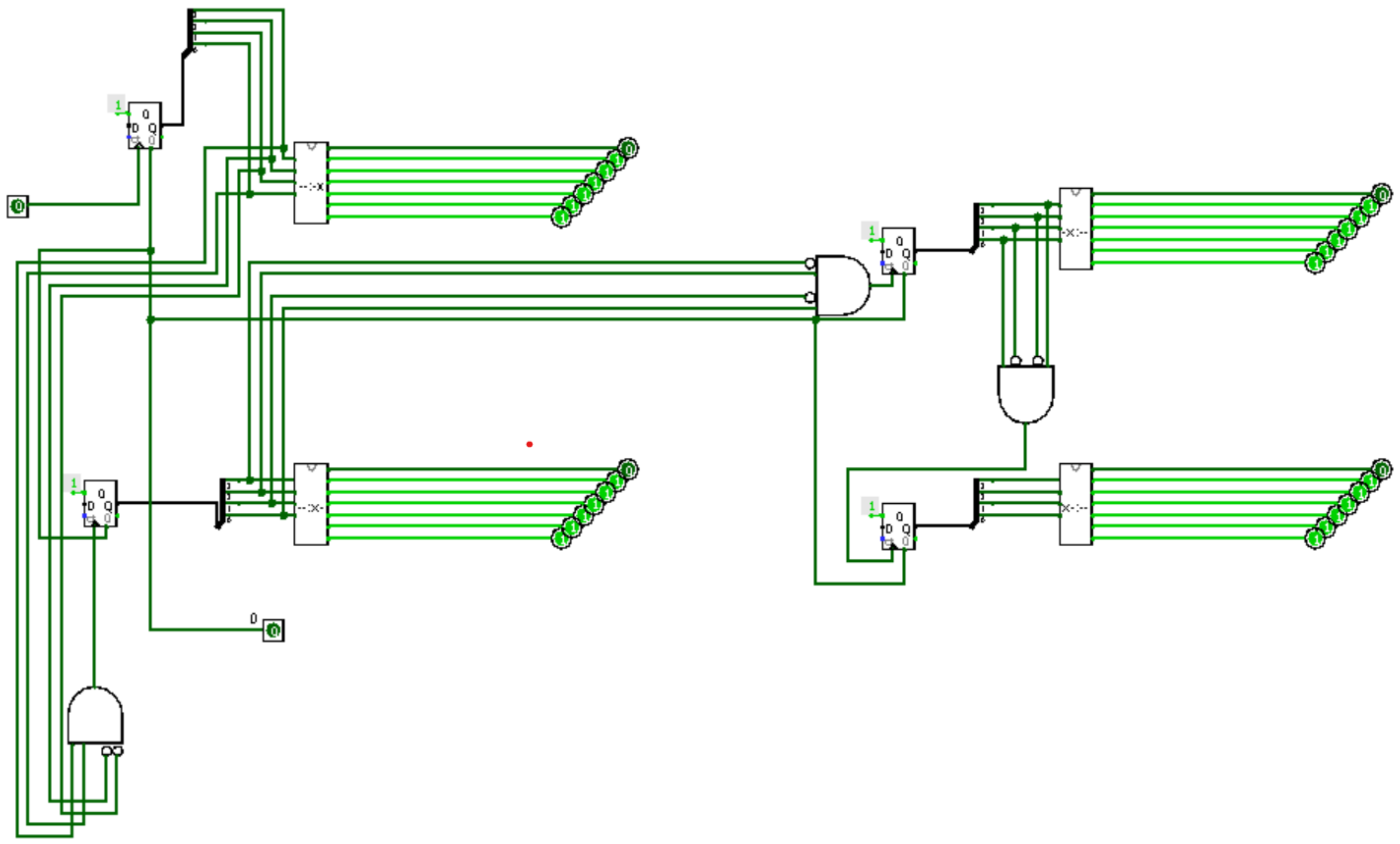
```
module chess_clock_tb();
   reg clk;
   reg reset;
   reg start;
   reg surrender_player1;
   reg surrender_player2;
   reg switch_turn;
   wire [6:0] seg_player1_min1;
   wire [6:0] seg_player1_min0;
   wire [6:0] seg_player1_sec1;
   wire [6:0] seg_player1_sec0;
   wire [6:0] seg_player2_min1;
   wire [6:0] seg_player2_min0;
   wire [6:0] seg_player2_sec1;
   wire [6:0] seg_player2_sec0;
   wire player1_green_led;
   wire player2_green_led;
   wire player1_red_led;
wire player2_red_led;

// Instantiate the chess_clock module
chess_clock clock_inst (
   .clk(clk),
   .reset(reset),
   .start(start),
   .surrender_player1(surrender_player1),
   .surrender_player2(surrender_player2),
   .switch_turn(switch_turn),
   .seg_player1_min1(seg_player1_min1),
   .seg_player1_min0(seg_player1_min0),
   .seg_player1_sec1(seg_player1_sec1),
   .seg_player1_sec0(seg_player1_sec0),
   .seg_player2_min1(seg_player2_min1),
   .seg_player2_min0(seg_player2_min0),
   .seg_player2_sec1(seg_player2_sec1),
   .seg_player2_sec0(seg_player2_sec0),
   .player1_green_led(player1_green_led),
   .player2_green_led(player2_green_led),
   .player1_red_led(player1_red_led),
   .player2_red_led(player2_red_led)
);

// Clock generation
always begin
   #5 clk = ~clk;
end
```

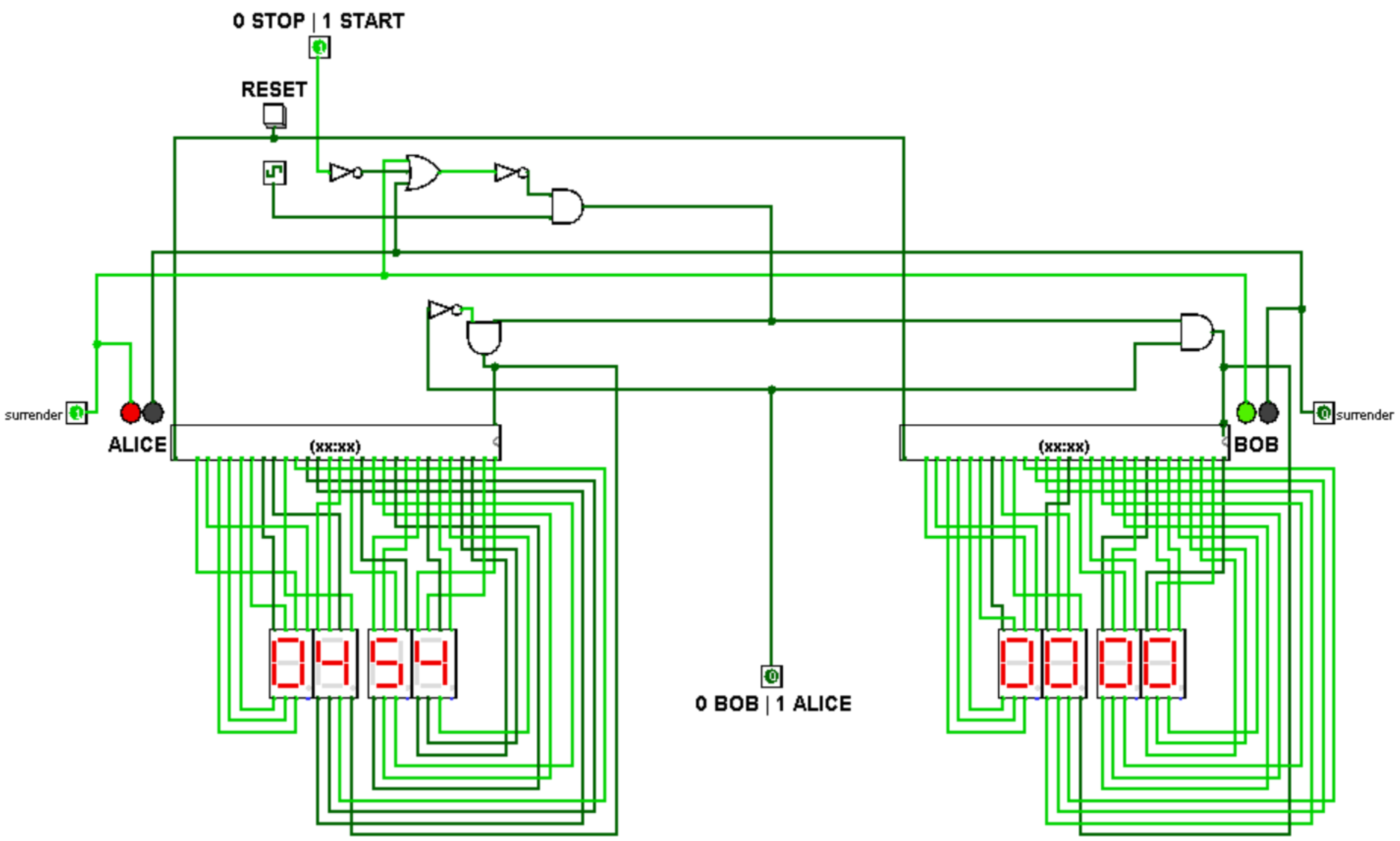
```
// Stimulus generation

initial begin

   clk = 0;

   reset = 1;

   start = 0;

   surrender_player1 = 0;

   surrender_player2 = 0;

   switch_turn = 0;


   // Apply reset and wait for a few clock cycles

   #10 reset = 0;

   #100;


   // Start the timer

   #10 start = 1;


   // Simulate player turns and timer countdown

   #100 surrender_player1 = 1;

   #100 surrender_player1 = 0;

   #100 surrender_player2 = 1;

   #100 surrender_player2 = 0;


   #100 switch_turn = 1;

   #100 switch_turn =0

   // Add more test scenarios as needed


   // End simulation
```

```
        #1000 $finish;

    end


endmodule
```

Logisim Part:

Clock IC





References:

https://sweetcode.io/logisim-software-digital-clock/

https://en.wikipedia.org/wiki/Chess_clock

https://digitalgametechnology.com/products/chess-clocks/dgt2010-official-fide-chess-clock

https://ro.farnell.com/

https://github.com/Ruben304/LogicDesign/blob/main/FINAL/digital_clk_12hr.v
https://github.com/cadewey/Simple-Chess-Clock/tree/master/simplechessclock