Whack A Mole

Special Assignment and Lab Project Report

Course: 2EC201CC23: Digital Logic Design

Submitted by:

24BEC522	*24BEC524*
Rohit Vadhiya	Poojan Mehta

Department of Electronics and Communication Engineering,
Institute of Technology,
Nirma University,
Ahmedabad 382 481

[Nov-2024]

Abstract

This project aims to develop a stack module using Verilog, specifically designed for simulation in an FPGA environment This Whack-a-Mole project explores interactive game design through an FPGA-based setup, integrating hardware and software components to simulate a classic arcade experience. The system utilizes LEDs and push buttons to represent "moles" appearing at random intervals. When an LED lights up, players must press the corresponding button to "whack" the mole, with each successful press increasing their score. The game includes adjustable difficulty settings, enhancing player engagement and reaction challenges. This project demonstrates key concepts in embedded systems, including timing control, randomization, and user input handling, while providing an entertaining application that emphasizes practical skills in circuit design and coding.

Contents

1. Introduction

Introduction of the project : The Whack-a-Mole project is a classic arcade game reimagined as a digital logic design project. It involves designing a circuit that simulates the game's core functionality, including randomly lighting up "moles" (LEDs) and detecting player hits (button presses). This project provides a hands-on opportunity to apply digital logic concepts, such as combinational and sequential logic, to create a real-world application.

Key Components and Functionality

- Random Number Generator: A circuit that generates random numbers to determine which "mole" should pop up next.
- LED Display: A series of LEDs representing the "moles" that can be illuminated.
- Button Inputs: Buttons that simulate the player's whacks.
- Scoring and Timing: A circuit to keep track of the player's score and time remaining.
- Game Over Detection: A mechanism to determine when the game is over, based on time or score.

Applications : The Whack-a-Mole game concept is a useful application in digital logic design projects due to its straightforward structure and flexibility for incorporating various digital logic principles

1. Sequential Logic and Timing Control

- **Application**: The timing and random appearance of the "moles" rely on sequential logic circuits or timers that control the intervals between LED activations.
- **Benefit**: Students learn to design circuits using flip-flops, counters, and clock dividers, enabling them to manage timing sequences accurately.

2. State Machines

- **Application**: A Whack-a-Mole game can be built around a finite state machine (FSM) where each state represents the status of the game (e.g., idle, mole-on, score update).
- **Benefit**: Designing FSMs for this game helps students practice creating well-defined states and managing transitions between them, enhancing their understanding of state-based design.

2. Literature Survey/State of the Art Technology Available

The Whack-a-Mole game concept has served as a foundation for interactive game design and a learning tool in both digital logic and embedded systems. This section surveys the state-of-the-art technologies, techniques, and research advancements relevant to interactive game systems, focusing on hardware design, embedded systems, user interaction, and educational applications.

1. Embedded Systems in Gaming

- **ICs**: Modern Whack-a-Mole games in embedded design often use ICs such as the 555 Timer IC & 4017 decade Counter IC for control, timing, and input/output management.
- **FPGAs** : FPGAs offer flexibility in digital game design by allowing the integration of complex logic circuits and high-speed response systems. They are particularly suitable for applications requiring parallel processing, such as handling multiple LEDs and buttons in a Whack-a-Mole game.

2. Sensor and Display Technologies

- **LED Displays and Segmented Displays**: For displaying scores and active targets, high-efficiency LEDs and seven-segment displays are standard due to their low cost, ease of interfacing, and visibility. Research in display technology has introduced addressable LEDs (like WS2812B) that allow each LED to be individually controlled, offering complex lighting effects and real-time feedback in gaming systems.

3. Timing Control in Digital Logic Design

- **Timing and Interrupt Control**: Timing control is critical for pacing in interactive games. Advances in interrupt-based handling in ICs allow for responsive button inputs, ensuring player actions are registered in real time. Research on embedded timers and low-power modes in ICs also aids in optimizing system response while maintaining power efficiency.

4. User Interaction and Haptic Feedback

- **Button and Haptic Feedback Integration**: User interaction through button presses has evolved with haptic feedback mechanisms. Modern Whack-a-Mole projects incorporate vibration motors or haptic drivers to enhance the user experience by adding physical feedback upon hitting a target. Haptic feedback has been shown to improve immersion in gaming, as it provides real-time feedback to the user.

5. Educational Applications and Gamification

- **Game-Based Learning (GBL) Frameworks**: Whack-a-Mole has been used as a learning tool in digital logic and embedded systems courses, often serving as an engaging introduction to circuit design, state machines, and programming. Studies indicate that game-based learning improves engagement and retention, making the Whack-a-Mole game a popular choice in educational kits.
- **Gamification in Learning Embedded Systems**: Recent research highlights the benefits of gamification in technical education. By structuring digital design tasks as games, students find it easier to understand and apply fundamental principles. Whack-a-Mole projects exemplify the integration of gamification in engineering education, aligning with trends in experiential learning.

The Whack-a-Mole game’s simplicity and adaptability make it ideal for exploring a range of state-of-the-art technologies in interactive systems and digital logic design. By leveraging advancements in microcontrollers, sensors, display technology, randomization, and user interaction, this project serves as a hands-on application for understanding modern embedded systems and digital circuit design.

3. Proposed Solution/Methodology

In this methodology, the Whack-a-Mole game will be built entirely using combinational and sequential digital ICs(555 counter,7014), such as timers, counters, shift registers, and decoders, to handle game logic, timing, and display. The project emphasizes classic digital logic design principles and avoids programmable microcontrollers, providing a foundation in digital circuit design.

Game Timing and Mole Activation:

- **555 Timer ICs**: 555 timers configured in astable mode will generate periodic pulses to control the mole appearance intervals. The timing frequency can be adjusted with resistors and capacitors, allowing the game's difficulty to be modified.
- **4017 Counter IC** : It be used to cycle through different "mole" positions. Each pulse from the timer increments the counter, selecting which LED (mole) will turn on.

LED Display and Button Matrix:

- **LED Driver ICs**: Using a 3-to-8 or 4-to-16 decoder, each output can correspond to an LED, representing a mole. When the LFSR or counter selects an output line, the corresponding LED lights up.
- **Button Matrix**: A 3x3 matrix of push buttons corresponding to the LEDs will allow the player to "whack" the mole. When a button is pressed, it generates an input signal that is checked against the active mole position.

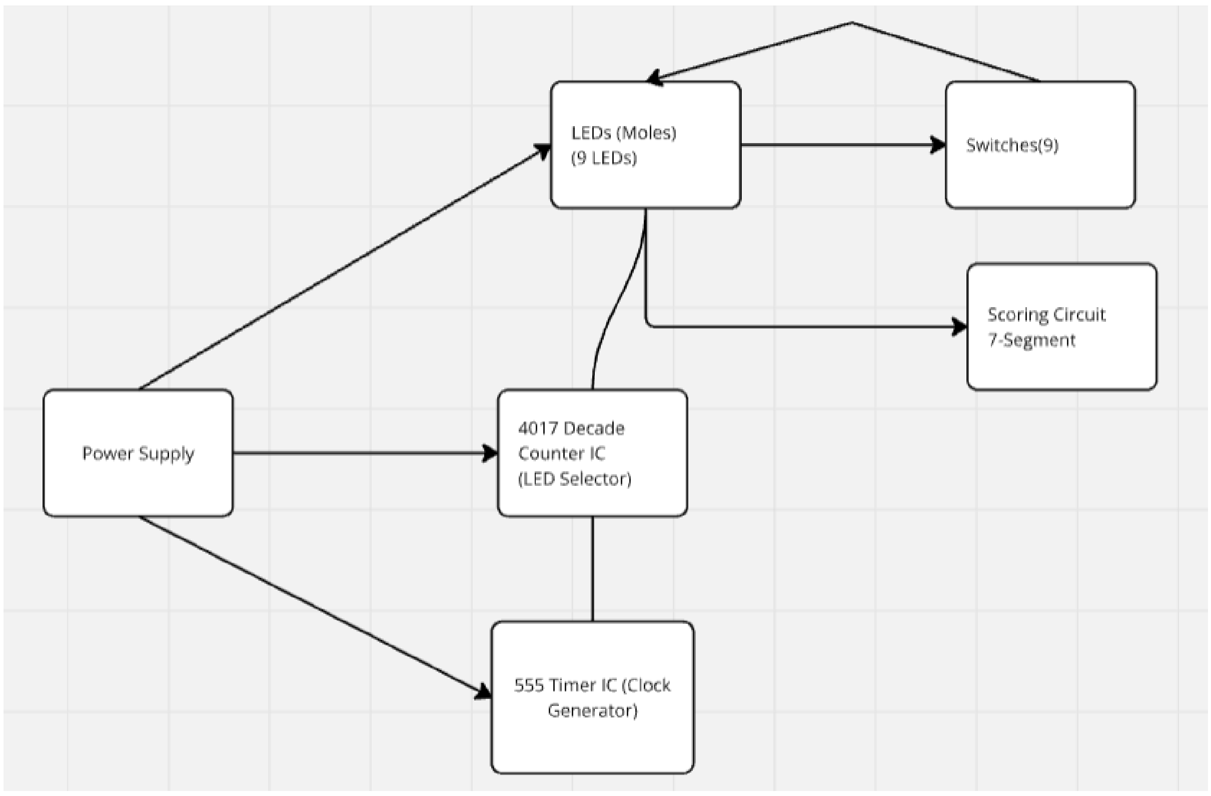
Score Counting and Display:

- **Binary Counter** : A counter IC will increment each time a player correctly hits a mole, keeping track of the score. The counter output is connected to a 7-segment decoder for display.
- **7-Segment Decoder** : The score counter's binary output will feed into a BCD-to-7-segment decoder, driving a 7-segment display to show the current score.

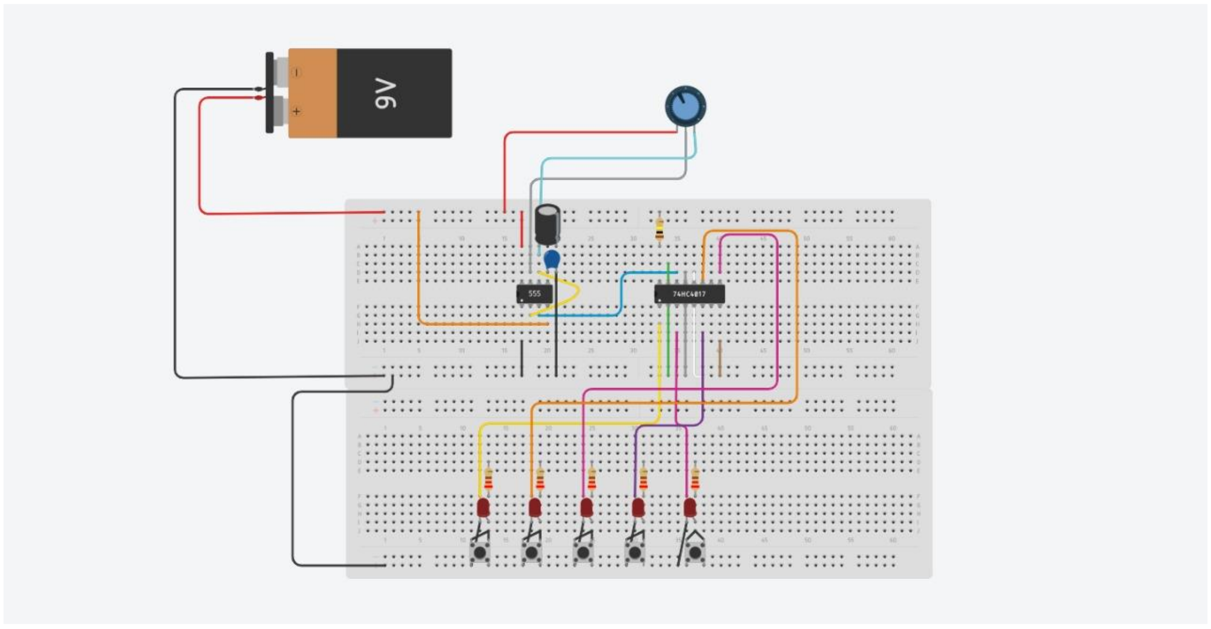
Game Over Indicator:

- **Comparator IC**: A comparator can be used to detect when the score reaches a pre-defined value (e.g., 9), at which point the game-over indicator LED lights up, and the system locks to prevent further scoring.

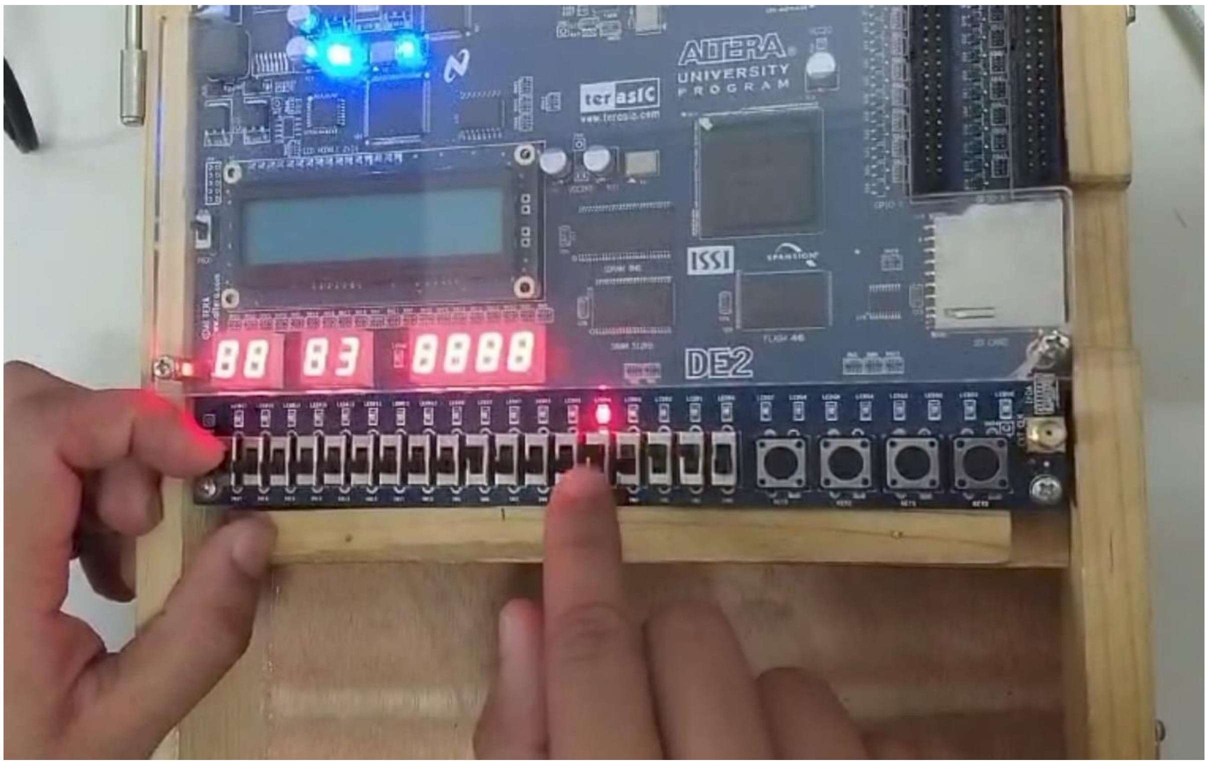
4. Block Diagram :



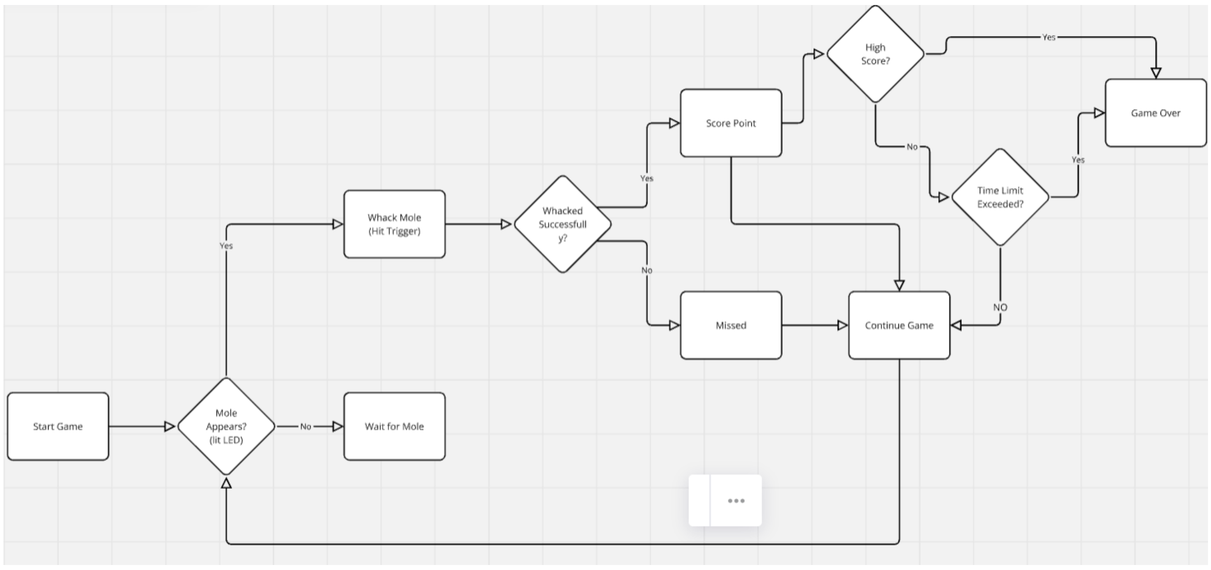
Circuit Diagram:



FPGA implementation :



Design(Flowchart)

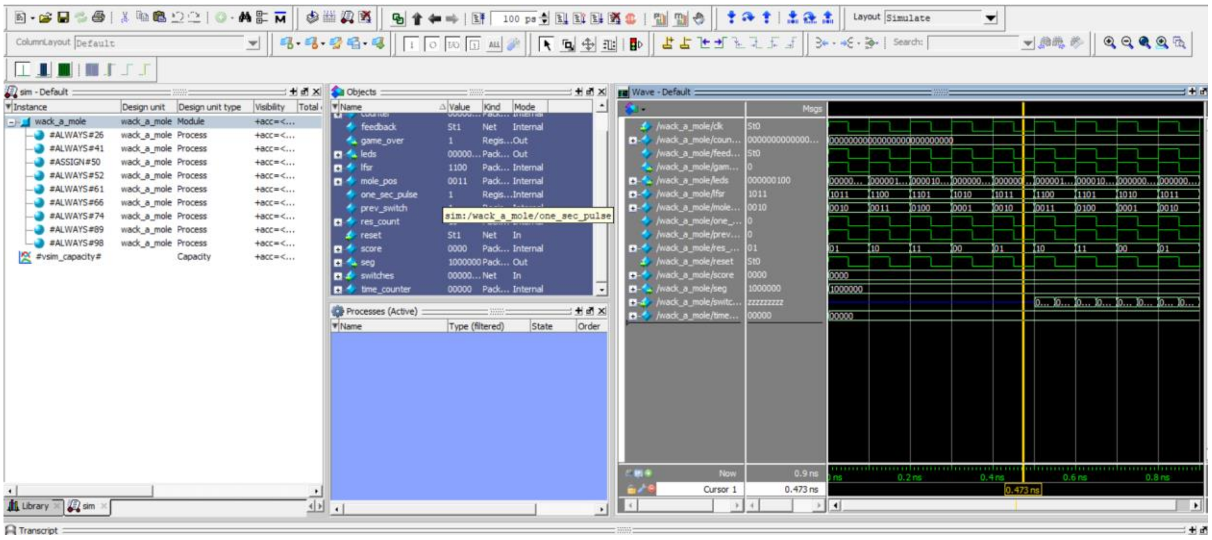


Details of All Hardware Components Used and Its Justification

- 555 counter IC :Configure the 555 timer in astable mode to generate continuous clock pulses.
- CD4017 IC **:** is a decade counter IC commonly used in digital logic projects to count and control outputs sequentially
- 10kΩ Resistor
- 1kΩ Resistor
- 10 µF Capacitor
- Power Supply ((9V)

Simulation Results :

Modelsim Simulation:



Instance	Design unit	Design unit type	Visibility
wack_a_mole	wack_a_mole	Module	+acc=<...
#ALWAYS#26	wack_a_mole	Process	+acc=<...
#ALWAYS#41	wack_a_mole	Process	+acc=<...
#ASSIGN#50	wack_a_mole	Process	+acc=<...
#ALWAYS#52	wack_a_mole	Process	+acc=<...
#ALWAYS#61	wack_a_mole	Process	+acc=<...
#ALWAYS#66	wack_a_mole	Process	+acc=<...
#ALWAYS#74	wack_a_mole	Process	+acc=<...
#ALWAYS#89	wack_a_mole	Process	+acc=<...
#ALWAYS#98	wack_a_mole	Process	+acc=<...
#vsim_capacity#		Capacity	+acc=<...

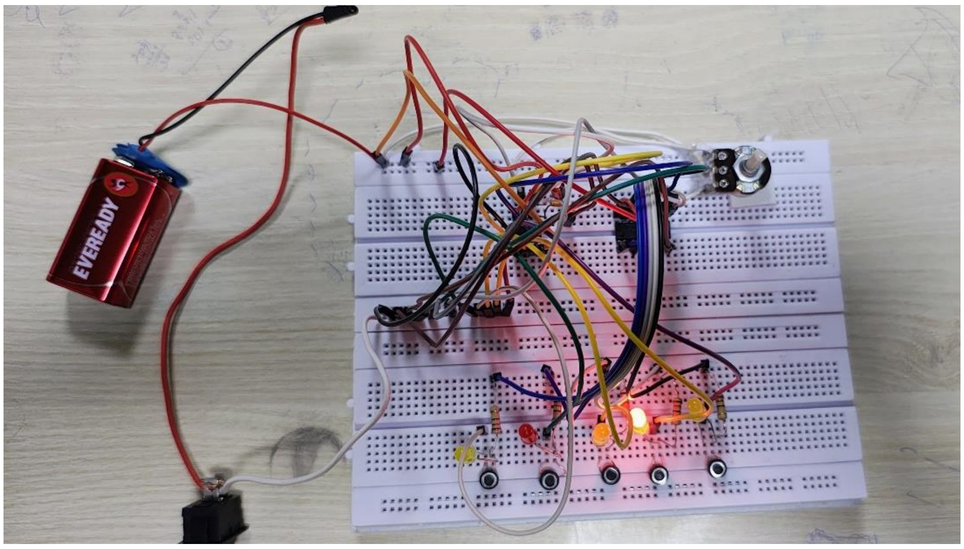
Name	Value	Kind	Mode
feedback	St1	Net	Internal
game_over	1	Regis...	Out
leds	00000...	Pack...	Out
lfsr	1100	Pack...	Internal
mole_pos	0011	Pack...	Internal
one_sec_pulse	1	Regis...	Internal
prev_switch			
res_count			
reset	St1	Net	In
score	0000	Pack...	Internal
seg	1000000	Pack...	Out
switches	00000...	Net	In
time_counter	00000	Pack...	Internal

sim:/wack_a_mole/one_sec_pulse

	Msgs
/wack_a_mole/clk	St0
/wack_a_mole/coun...	0000000000000...
/wack_a_mole/feed...	St0
/wack_a_mole/gam...	0
/wack_a_mole/leds	000000100
/wack_a_mole/lfsr	1011
/wack_a_mole/mole...	0010
/wack_a_mole/one_...	0
/wack_a_mole/prev...	0
/wack_a_mole/res_...	01
/wack_a_mole/reset	St0
/wack_a_mole/score	0000
/wack_a_mole/seg	1000000
/wack_a_mole/switc...	zzzzzzzzz
/wack_a_mole/time...	00000

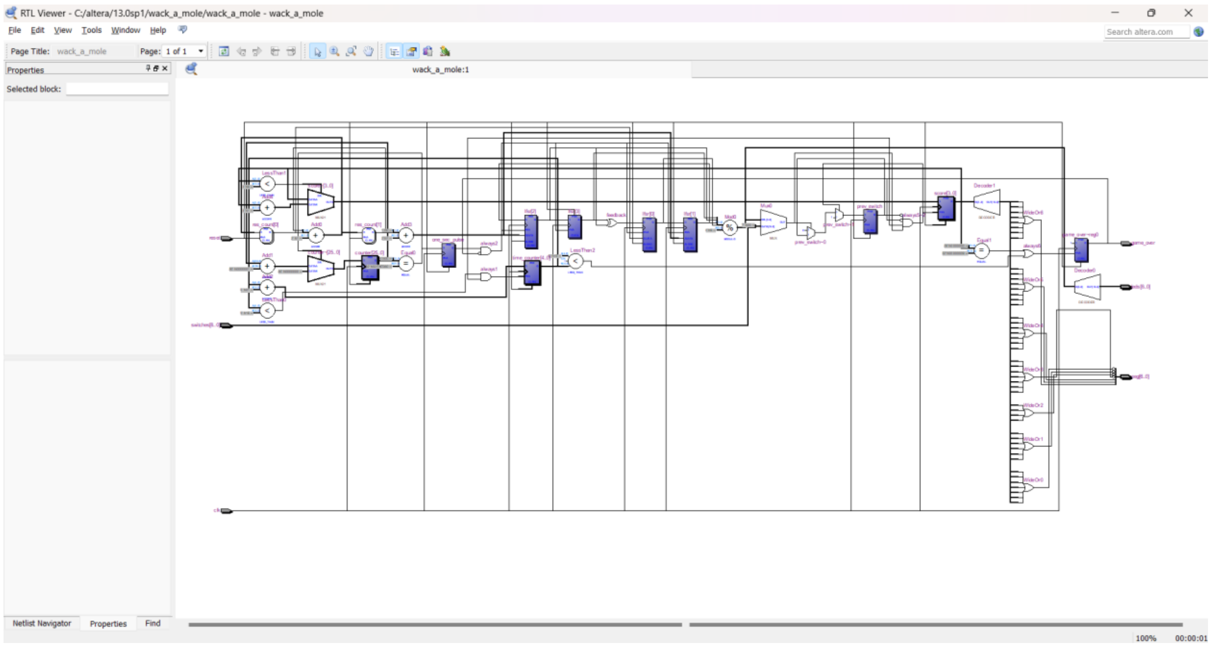
Now 0.9 ns
Cursor 1 0.473 ns

Circuit Simulation :

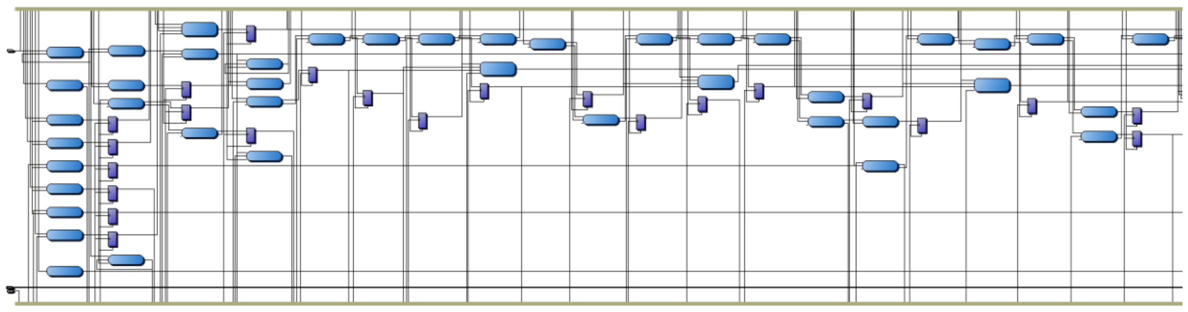


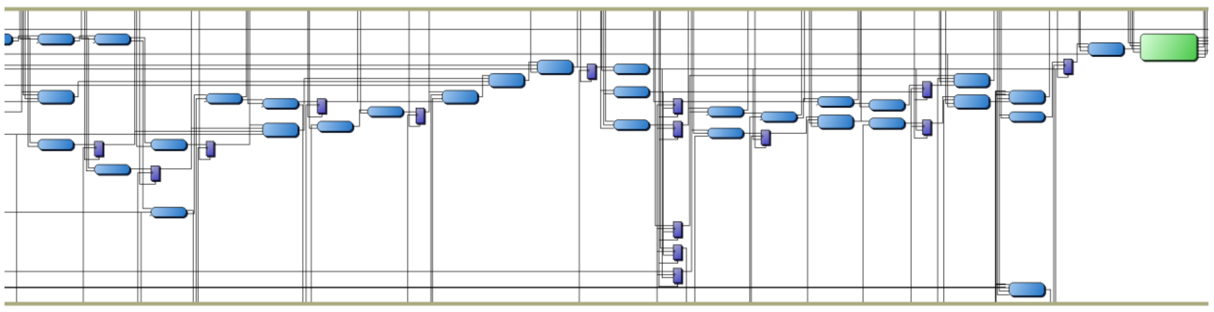
Results and Measurements

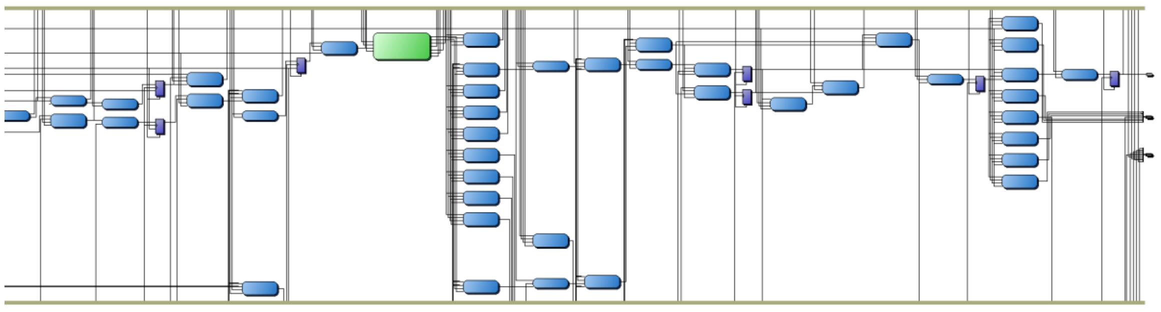
RTL View:



Map View:







Conclusion:

- Successfully implemented a Whack-a-Mole game using the 555 and 4017 ICs.
- Demonstrated understanding of timing and counting circuits in electronics.
- Possibly integrated a scoring system or difficulty levels based on the player's performance.
- 555 and 4017 are widely used, easy-to-understand ICs that provide fundamental experience in timing and counting, enabling creativity in a wide range of electronic projects.

Future Scope:

- **Adding More Features**: multiple levels of difficulty, or even a high score tracker.
- **User Interface**: Improve the user interface for a more enjoyable experience.
- **Component Optimization**: Experiment with different components or configurations to enhance performance or reduce complexity.

References

- **Online sources :**
  https://chat.openai.com/

Appendix

```
module wack_a_mole(
   input wire clk,             // Clock input
   input wire reset,           // Reset switch
   input wire [8:0] switches,  // 9 switches for whacking moles
   output reg [8:0] leds,      // 9 LEDs representing holes
   output reg [6:0] seg,       // 7-segment display for score
   output reg game_over        // Signal to indicate the game is paused
);


   // Internal signals
   reg [25:0] counter;        // 26-bit counter for clock divider (1 second)
   reg one_sec_pulse;         // 1-second pulse signal
   reg [3:0] mole_pos;        // Mole position (0 to 8)
   reg [3:0] score;           // Player score (0 to 9)
   reg [4:0] time_counter;    // Time counter to track 30 seconds
           reg [1:0] res_count;
   // LFSR-based random number generator for mole position
   reg [3:0] lfsr;            // 4-bit LFSR
   wire feedback;             // Feedback bit for LFSR
           initial begin
           res_count = 2'd00;
```

```
          end
// Clock Divider to generate 1-second pulse
always @(posedge clk or posedge reset) begin
    if (reset) begin
        counter <= 0;
        one_sec_pulse <= 0;
                                        res_count = res_count + 1;
    end else if (counter == 50000000) begin // Assuming a 50MHz clock
        one_sec_pulse <= 1;
        counter <= 0;
    end else begin
        one_sec_pulse <= 0;
        counter <= counter + 1;
    end
end

// Time Counter (Track time for 30 seconds)
always @(posedge clk or posedge reset) begin
    if (reset) begin
        time_counter <= 0;  // Reset time
    end else if (one_sec_pulse && time_counter < 30) begin
        time_counter <= time_counter + 1;
    end
end

// LFSR logic for random mole position generation
assign feedback = lfsr[3] ^ lfsr[2]; // Feedback function using XOR

always @(posedge clk or posedge reset) begin
    if (reset) begin
        lfsr <= 4'b1010 + res_count;  // Seed the LFSR with a non-zero value
    end else if (one_sec_pulse && !game_over) begin
        lfsr <= {lfsr[2:0], feedback};  // Shift the LFSR
    end
end

// Map the LFSR value to mole positions (0 to 8)
always @(*) begin
    mole_pos = lfsr % 9;  // Map LFSR output to range 0 to 8
end
```

```
// LED Control (One LED on based on mole position)
always @(*) begin
    leds = 9'b000000000;     // Turn off all LEDs by default
    leds[mole_pos] = 1;      // Turn on the LED corresponding to the mole position
end

// Score Counter (Check if the correct switch is pressed once per second)
reg prev_switch;           // Stores the previous state of the switch for debounce

always @(posedge clk or posedge reset) begin
    if (reset) begin
        score <= 0;          // Reset score
        prev_switch <= 0;
    end else if (one_sec_pulse && switches[mole_pos] && !game_over && !prev_switch) begin
        // Increment score once per second when the correct switch is pressed and game is not over
        if (score < 9)
            score <= score + 1;
        prev_switch <= 1;     // Set prev_switch to prevent repeated scoring within the same second
    end else if (!switches[mole_pos]) begin
        prev_switch <= 0;     // Reset prev_switch if the correct switch is released
    end
end

// Game Over Logic (Stop game when score reaches 9 or time exceeds 30 seconds)
always @(posedge clk or posedge reset) begin
    if (reset) begin
        game_over <= 0;       // Reset game state
    end else if (score == 9 || time_counter >= 30) begin
        game_over <= 1;       // Pause game if score reaches 9 or time exceeds 30 seconds
    end
end

// 7-Segment Display for Score
always @(*) begin
    case(score)
        4'd0: seg = 7'b1000000;  // Display 0
        4'd1: seg = 7'b1111001;  // Display 1
        4'd2: seg = 7'b0100100;  // Display 2
        4'd3: seg = 7'b0110000;  // Display 3
```

```
            4'd4: seg = 7'b0011001;  // Display 4
            4'd5: seg = 7'b0010010;  // Display 5
            4'd6: seg = 7'b0000010;  // Display 6
            4'd7: seg = 7'b1111000;  // Display 7
            4'd8: seg = 7'b0000000;  // Display 8
            4'd9: seg = 7'b0010000;  // Display 9
            default: seg = 7'b1111111; // Blank display for invalid numbers
        endcase
    end

endmodule
```