Habib University

Dhanani School of Science and Engineering

Digital Logic and Design

EE/CS 172/130 – T2

Project Title:

Chrome Dino Runner

Team Members:

Bilal Ahmed, Daniyal Shadab, Muhammad Hayyan Khan, Hassaan Rashid

Professor: Farhan Khan

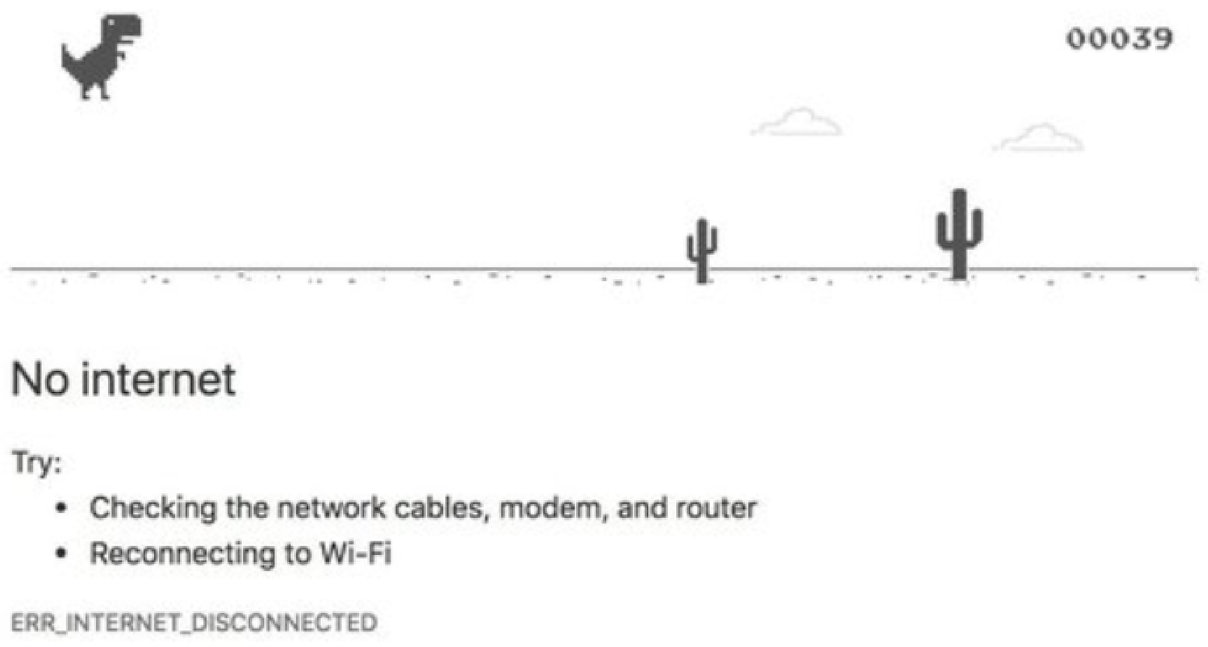
Abstract

Chrome Dino Runner is a captivating endless runner game. The game allows players to control a T-Rex, navigating through a field filled with obstacles. Aimed at both casual gamers and hobbyists, the game is designed to be engaging and thrilling.

The game offers players the option to either use the space key or buttons on the FPGA board, as an input device to make the T-Rex jump to avoid obstacles. The objective is to travel as far as possible, avoiding cactuses, before the T-Rex collides with an obstacle. Another key on the keyboard, the Enter key, as well as another button on the FPGA board offers the players to reset the game, if they want to start again.

Introduction:

Our primary motivation for this game was to develop this legendary game offered by Google Chrome browser from all our childhoods, which was the best way to pass time typically when we lost a stable internet connection.



No internet

Try:

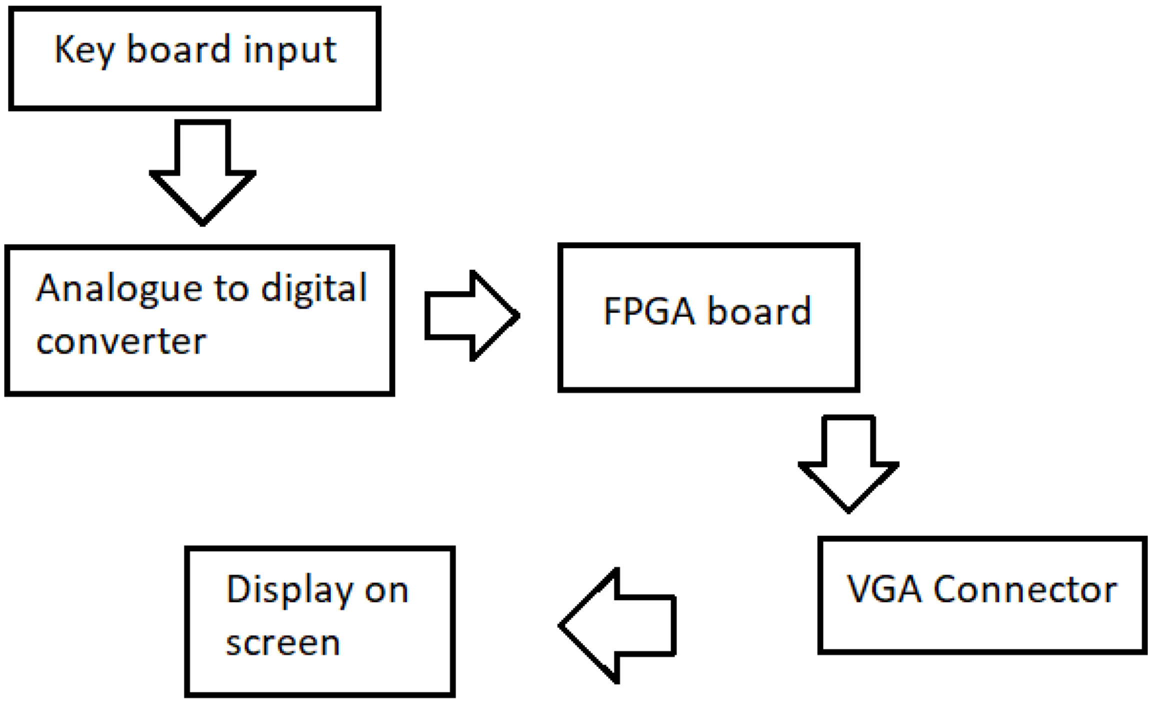
- Checking the network cables, modem, and router
- Reconnecting to Wi-Fi

ERR_INTERNET_DISCONNECTED

Our project is a single player game, where the user uses the space bar to jump over obstacles, and the Enter key to reset the game.

Implementation:

We will have three main components for this project: input, control unit and the game display (imaging block). The components are described in the following sections.

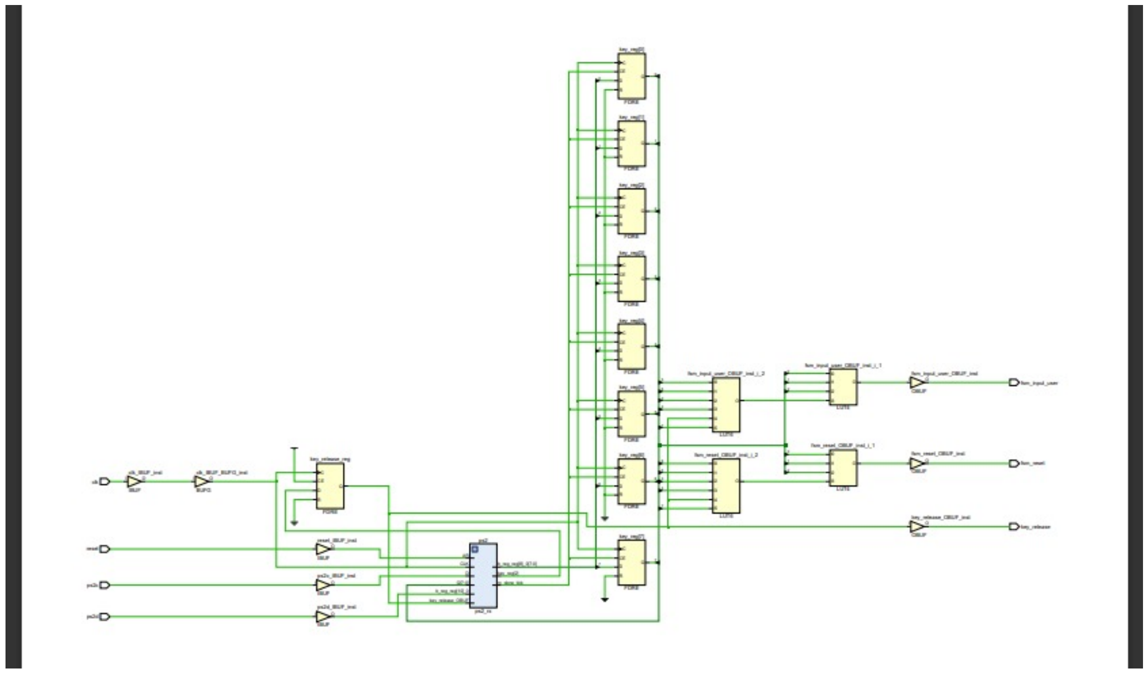


Input Block:

The user will interact with the game using the space bar and the enter button on the keyboard, which will serve as our sources of input. Alternatively, two designated the buttons on the FPGA board can be used as well.

Keyboard:

The space bar is programmed in Verilog to make the dinosaur jump. When the user presses the space bar, a signal is sent to the game logic, triggering the dinosaur to jump over obstacles. The Dino will move vertically at a set speed, up and down to avoid the cactus, the obstacle. On the other hand, the enter key is used to reset the game. When the enter key is pressed, it triggers a reset signal in the Verilog code, which resets the game state, and brings the user back to the start screen, where the user is required to press the space bar to start the game.



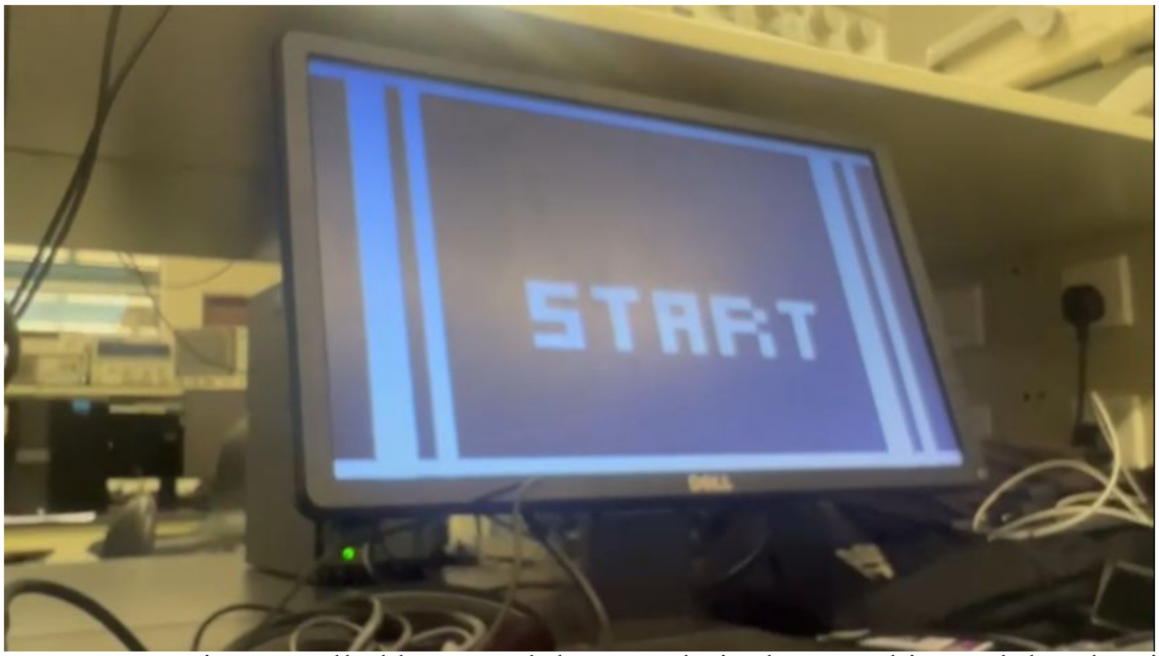
Control Block:

The game consists of three main states which are represented by three different screen displays:

1. Start Screen
2. Game Screen
3. End Screen

Start Screen:

This screen appears as the first screen of the game. It displays the message “Start”.

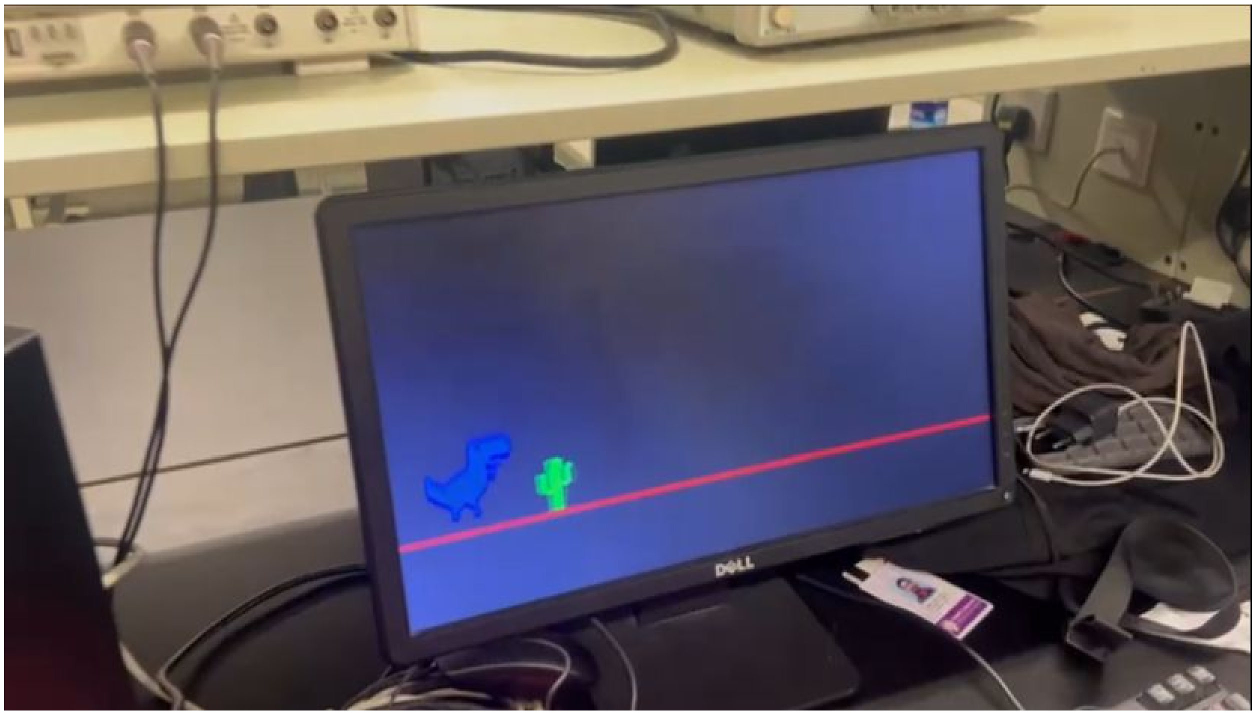


The start screen is controlled by a module named pixel_gen. This module takes in various inputs including pixel coordinates (pixel_x, pixel_y), a clock signal (clk_div1), a video on signal (video_on), and state information. It outputs RGB color values (red, blue, green).

When the state input is 3'b000, the display_start module (p1) is invoked with the pixel coordinates. If display1 is high, the RGB values are set to full intensity (4'hF), displaying the start screen.

The state transition from the start screen to the main game screen is controlled by the state input to the pixel_gen module. When the state changes (possibly triggered by an external event like a switch on the FPGA), the game transitions from the start screen to the main game screen.

Game Screen:



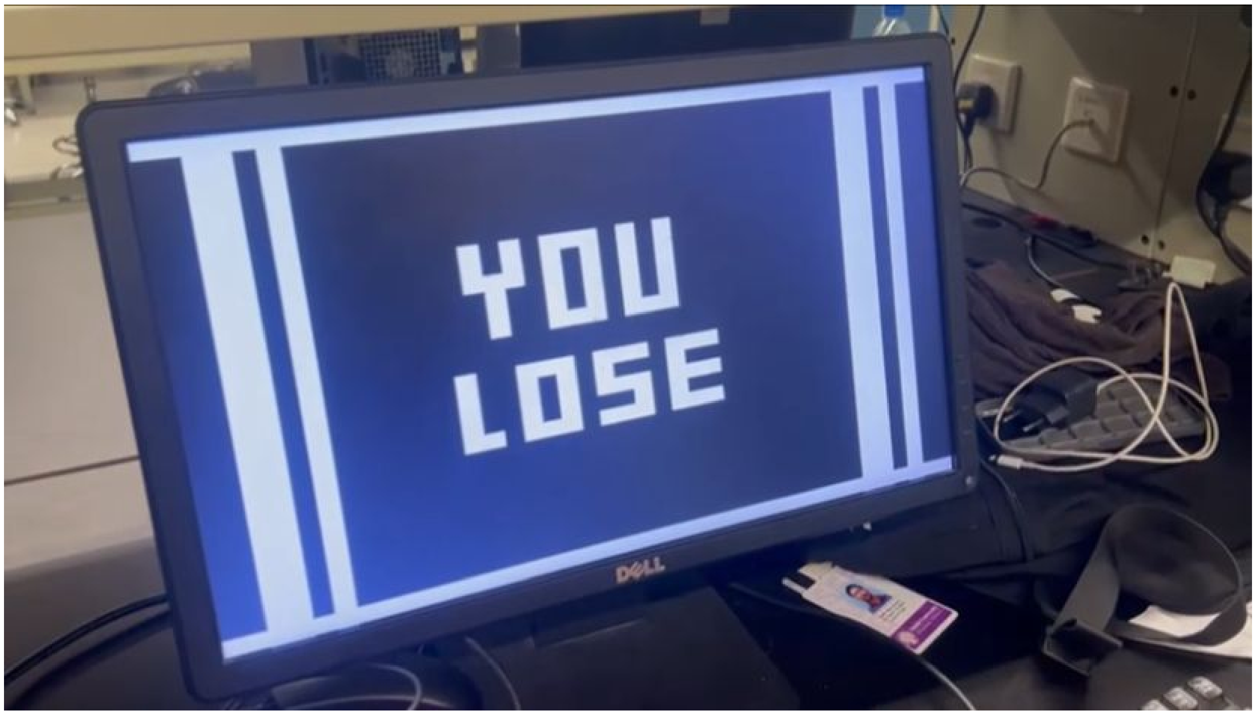
The movement of the Dino (mover_dino variable) is controlled by the pixel_gen module. This module takes in various inputs including pixel coordinates (pixel_x, pixel_y), a clock signal (clk_div1), a video on signal (video_on), and state information.

When the state input is 3'b001 (running state), the RGB values are set based on the pixel's x and y coordinates and the mover_dino variable. The conditions seem to be checking if the pixel is within certain ranges, which might correspond to the shape or position of the dinosaur in the game.

When the state input is 3'b010 (move up state) or 3'b011 (move down state), the RGB values are set based on the pixel’s x and y coordinates and the mover variable. The conditions seem to be checking if the pixel is within certain ranges, which might correspond to the shape or position of an object in the game.

End Screen:

When the Dino collides with the cactus, the game ends, and the state changes and this screen is displayed in the end.



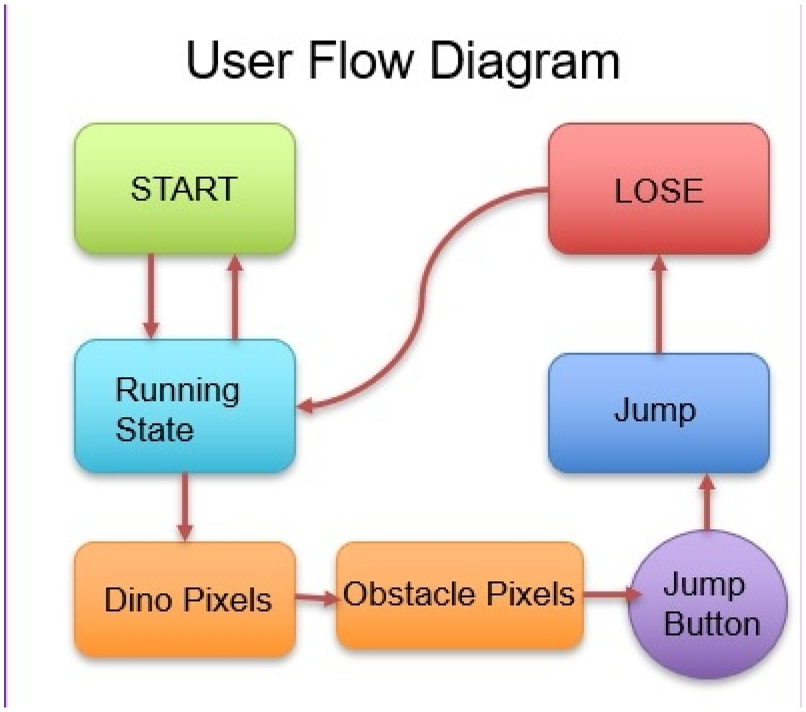
Output Block:

The display screen is generated using the VGA (video graphic array) connector. The display which we are using is standard 640 x 480 pixels.

For imaging on the horizontal axis, we turn the video on from 0 till 639 pixels and for the vertical axis the video is on from 0 till 479 pixels as that is the main display, after that the video is turned off for borders and retracing.

To make the final screen we first used a clock divider to reduce the frequency from 100 MHz to 25 MHz for FPGA to work, then we used h counter to count the pixels of horizontal axis and counter for vertical axis. Then VGA sync was used for turning the video on the required part of the screen and finally pixel gen is used to create our desired screen.

User Flow Chart:



FSM:

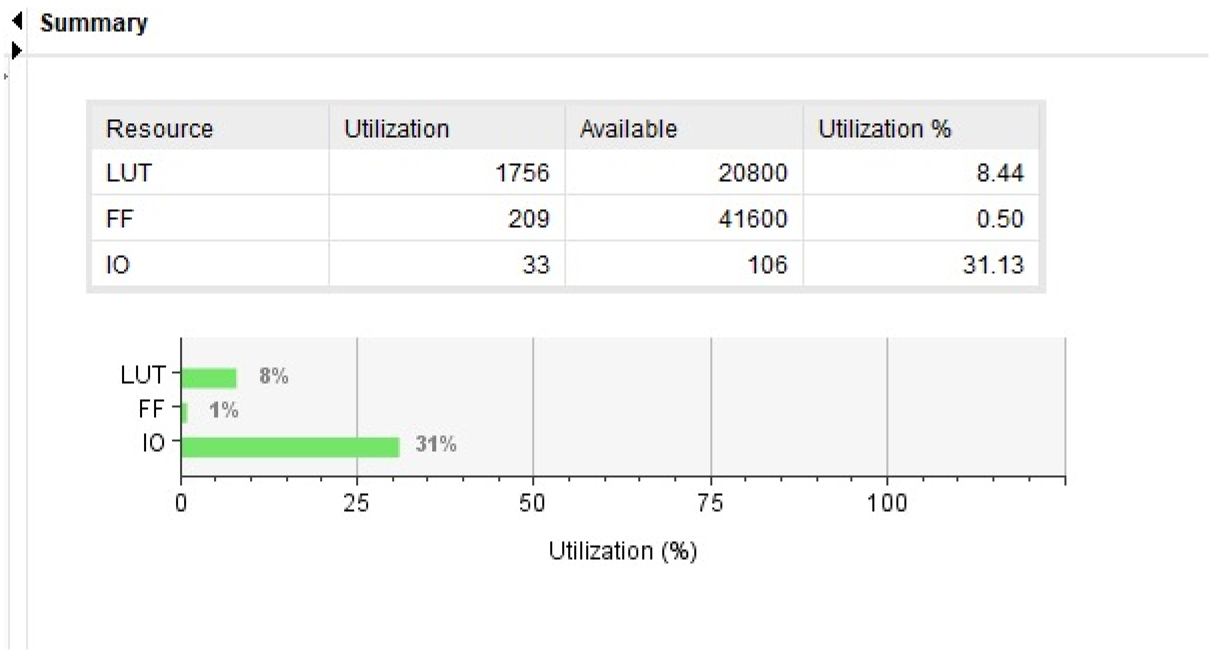
The Finite State Machine (FSM) in the code has three states:

1. Idle: In this state, the T-Rex is not running. This could be the initial state of the game, before the player has started the game.
2. Run: In this state, the T-Rex is running. This is likely the main gameplay state, where the T-Rex is moving and the player is actively playing the game.
3. End: In this state, the T-Rex has collided with an object and is stationary, and the score has stopped counting. This could be the end state of the game, after the player has lost.

The FSM transitions between these states based on certain conditions, which could be inputs from the player or events in the game.

This FSM can be considered a Mealy machine because the output (which could be the RGB values or other game variables) depends on both the current state and the current inputs (like pixel_x, pixel_y, mover, mover_dino). In a Mealy machine, the output can change whenever the input changes, which allows for more immediate responses to changes in input. This can be beneficial in a game setting, where you want the game to respond immediately to player inputs or game events.

Resource Utilization:


Summary

Resource	Utilization	Available	Utilization %
LUT	1756	20800	8.44
FF	209	41600	0.50
IO	33	106	31.13

Major Challenges:

The two major challenges that we faced were:

1. Keyboard inputs: We were initially only relying on the FPGA board button as primary inputs, and then we tried to do this. It took a lot of time, about 9 to 10 hours and a lot of external help, but we eventually figured it out.
2. Integrating the end screen: When the Dino collides with the cactus, the end screen is displayed, with the message “You Lose”. This we found to be particularly challenging, as we kept did not know how to display the message using bits, 1s and 0s.