ECE 2700 - Final Project

Simple Game: “Google Dino Run”

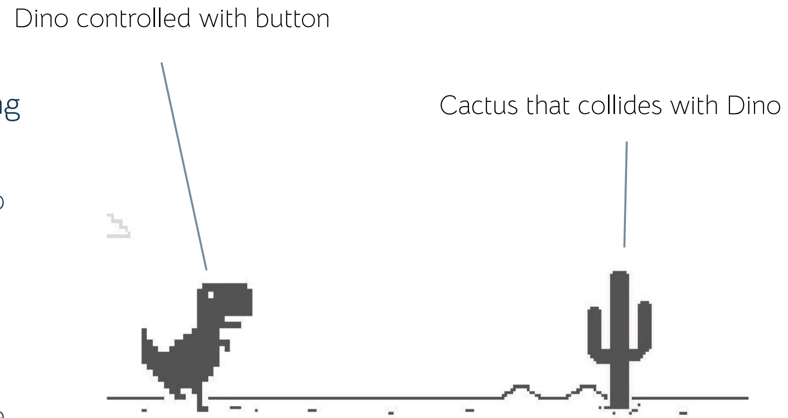
Joey Volcic, Modathir Bougrine, Dylan Wismont, Andrina Toma Toma

Overview

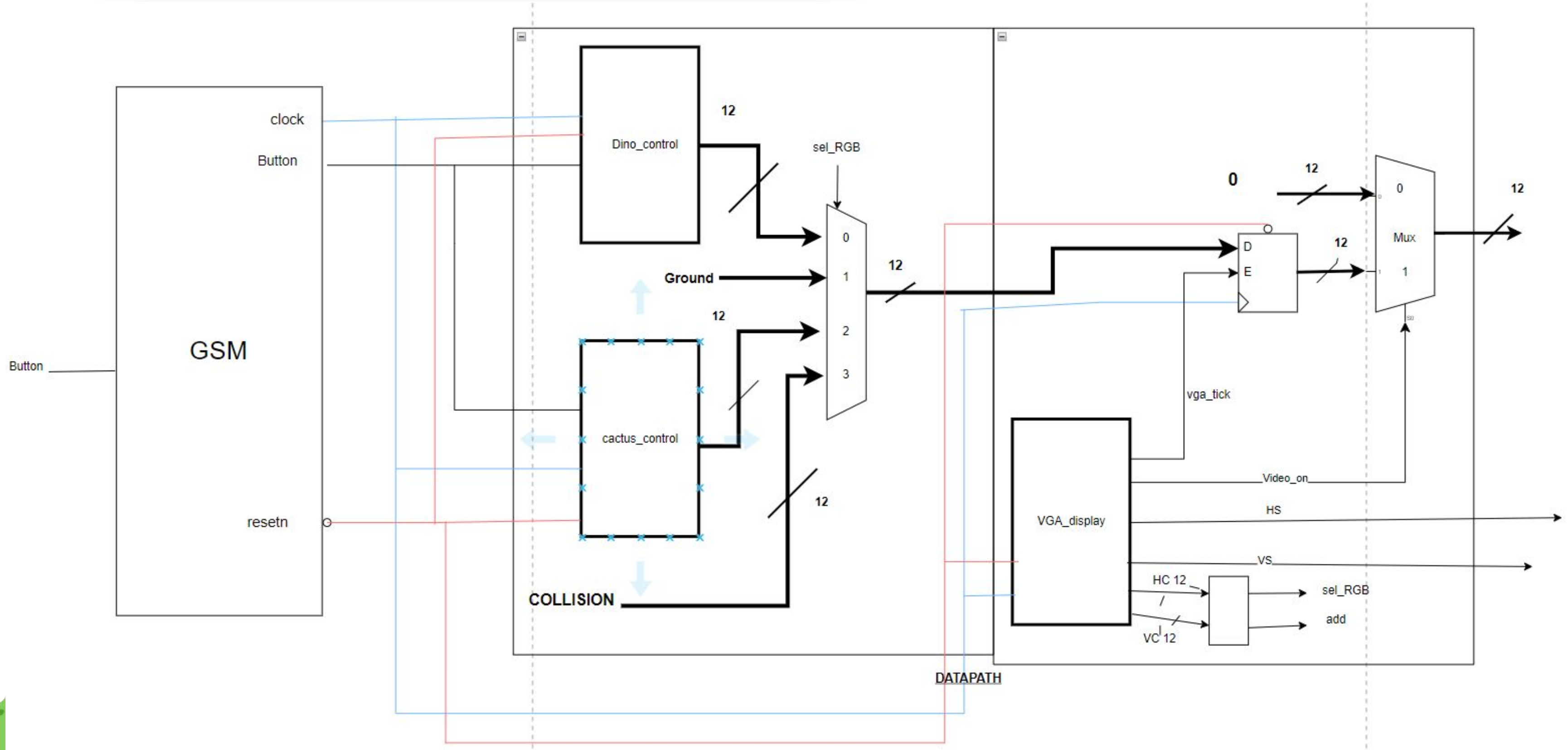
The goal of our project is to recreate the simple “No Internet” game, also known as Google Dino Run, using an FPGA and a VGA display.

We used Vivado’s VHDL development environment to implement this game on a Nexys A-7 board. The components being used from the board are an input button and the VGA port to display the game to the user.

The objective of the game is to control the Dino while avoiding cactus obstacles that are in the path of collision.



Circuit Block Diagram



Game State Machine

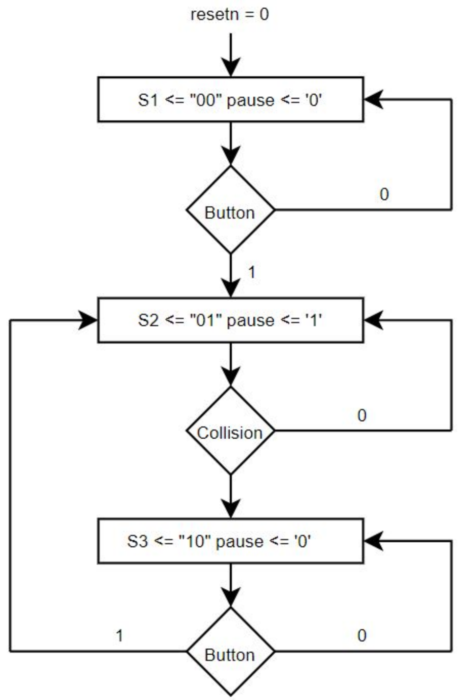
How the Game Works

3 main states of the game: S1, S2, S3

S1 initializes the game

S2 is when the user starts and plays the game

S3 is when the user “loses” the game and collision is detected with the Dino and Cactus. The game state machine goes back to S2 when the user pushes the button again.

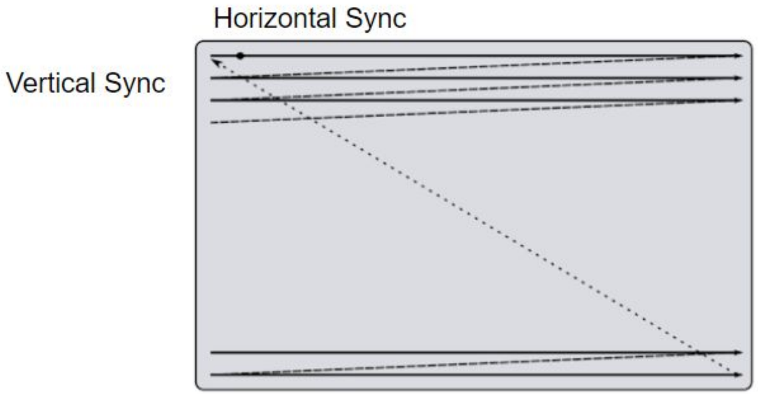


VGA Interface

The main output from the circuit is a VGA signal, a VGA signal requires 14 sub-signals to create visible output.

Twelve of which are used to determine the color of the current pixel, 4 bits are used to represent red, green, and blue. The other 2 signals are horizontal sync (HC), and vertical sync (VC),

The color values are selected based on the horizontal sync, and vertical sync counts. Based on the object we either dynamically generated the colour output or called the colour values from ram.

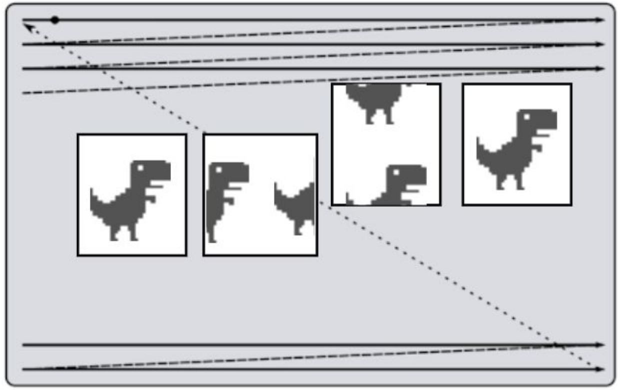


VGA Control

To display images on the vga we need to tell the control module where we want to display the image, and we need to position the image inside the area we want to display.

We used the horizontal sync and vertical sync values to determine where we want to display things on the screen

By adding and subtracting offset values we are able to position the displayed elements



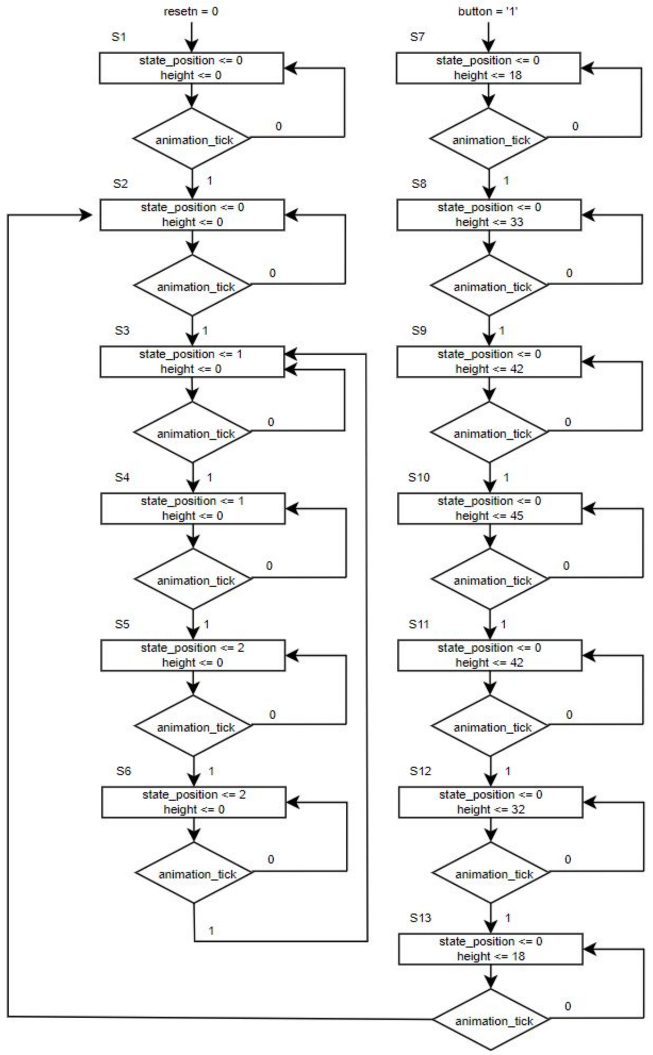
Dino State Machine

The animations are based off of a 0.05s clock

The state machine cycles though states 3 through 6 for the walking animation

The jump action is triggered asynchronously, which removes the feeling of input delay when playing the game

The jump height is non zero in states 7 through 13

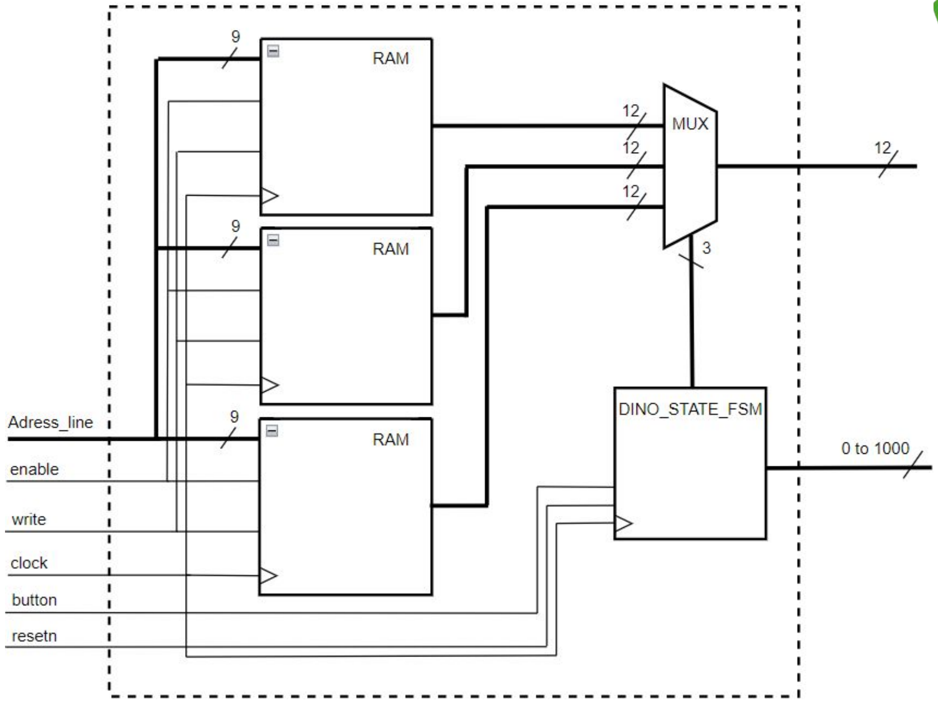


Dino Control

The dino control module loads the images into ram as pixel colours, with a call address

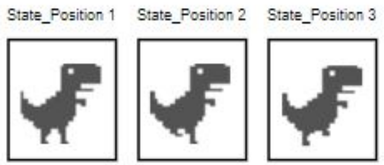
All three ram modules have the same address lines which output a pixel colour for the same screen position

The correct colour is then selected using the dino state machine

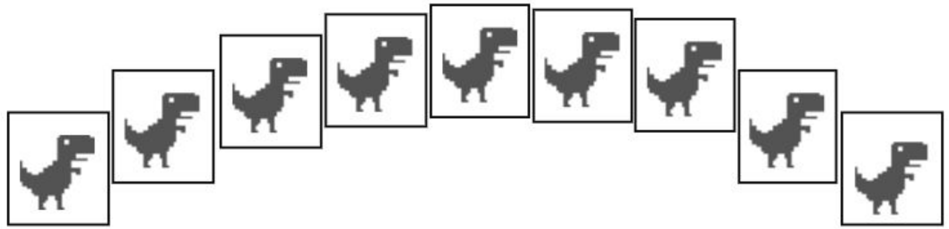


Dino Animations

Dino Animation for Running



Dino Animation for Jumping



CACTUS STATE MACHINE

Cactus State Machine

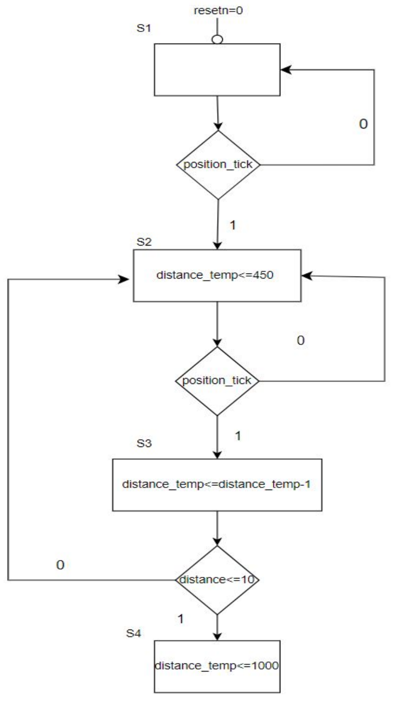
The state machine for the cactus cycles through four different states:

S1 : the loading state.

S2 : the cactus on screen state.

S3 : the cactus motion.

S4: cactus off screen then to reload.



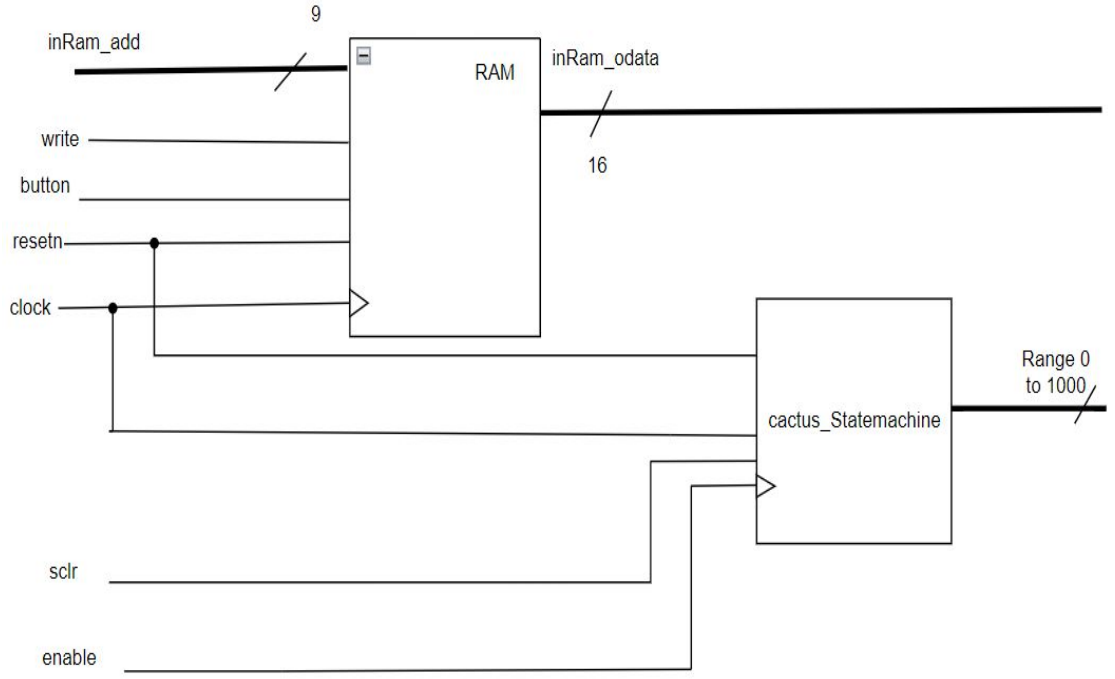
Cactus State Machine

Cactus Control

- The cactus control module loads the images into ram as pixel colours, with a call address.

- The cactus state machine sends that data when needed to be loaded and shown in the screen.

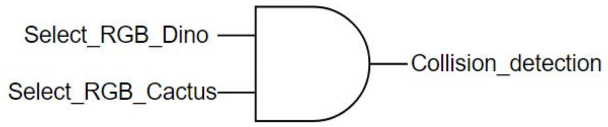
- The state machine also controls the motion of the cactus along the screen.

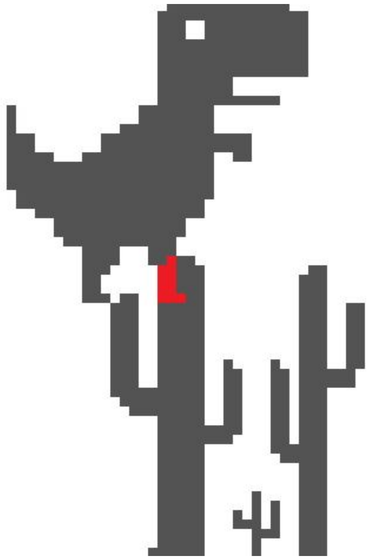


Collision detection

Since we already know where the dinosaur and cactus are located using the select signals generated earlier a simple and statement checking if sel_RGB_dino and sel_RGB_cactus will detect collisions

We then write red to those pixels and stop the game





Demo