**FLAPPY BIRD**

DSD PROJECT

Students: Andronescu Raluca-Antonia

Marinescu George

Moise Mihai-Ioan

Project supervisor: **Ileana Blaj**

**2023**

**Contents**

1. [**Specifications**](#Specifications)**............................................................................................2**
2. [**Design**](#Design)**.......................................................................................................3**
   1. [Black box](#Black)........................................................................................3
   2. [Detail diagram](#detail).................................................................................4
3. [**Structure and functionality**](#structure)**........................................................................4**
   1. [Resources](#resources)........................................................................................6
   2. [Flowchart/fsm](#flowchart).................................................................................10
4. [**Utility and results**](#utility)**......................................................................................11**
5. [**Further development**](#further)**.................................................................................13**
6. [**Technical justifications for the design**](#justifications)**......................................................13**

FLAPPY BIRD

1. **Specifications**

Our project is the game “Flappy Bird”. 🐤

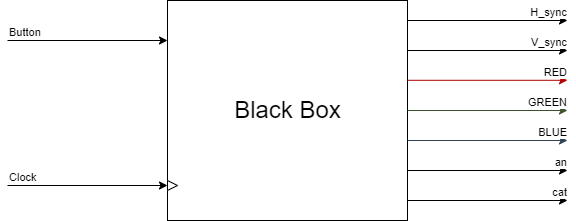
The purpose of the game is to navigate a little square (representing the bird), through a series of gaps between vertical pipes without colliding. The goal is to get a high score (to pass as many pipes as possible).

             Using a button as input, when it is '1' (the button is pressed), we simulate a floating bird, so the square will jump higher, when it is '0' (the button is not pressed), the square is influenced by gravity, causing it to descend.

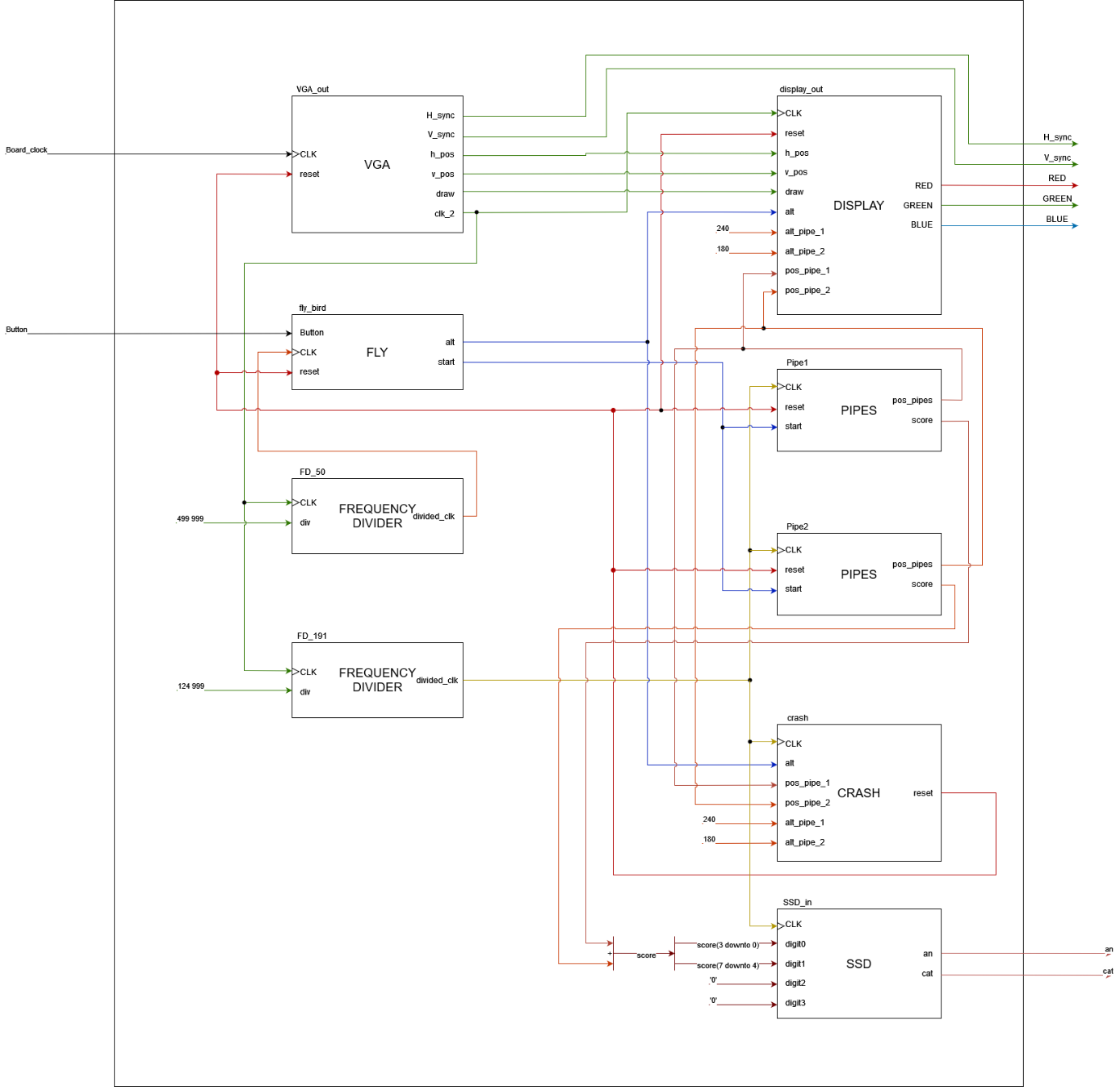
            The pipes that appear on the screen represent the obstacles. The square has to pass through the little gap, if it touches the pipe, the ground or the upper limit, the game is over.

           Points are awarded to the player for successfully passing through each set of pipes.

1. **Design**
   1. **Black box**



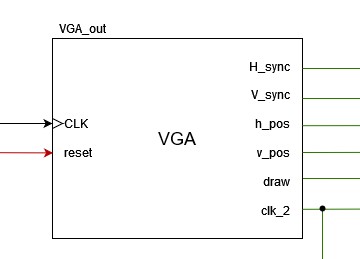
* 1. **Detail diagram**



1. **Structure and functionality**
   1. **Resources**
2. **MAIN**

The main module instantiates these components and connects their inputs and outputs to create the Flappy Bird game. It manages the synchronization, display, movement, scoring, collision detection, and score display of the game.

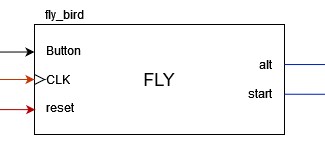
1. **VGA**

****

The VGA resource module generates the necessary synchronization

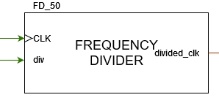
signals (h\_sync and v\_sync), tracks the current position of the display (h\_pos and v\_pos), provides a clock signal for other components (clk\_2), and signals when to draw pixels on the display (draw) based on the specified timing parameters.

1. **FLY**

****

The FLY resource  represents a simple simulation of a flying object controlled by a button. It defines an entity called "fly" with several input and output ports. When the button is pressed, the little square gains altitude, and when the button is released, it gradually descends. The altitude position is clamped within the range of 0 to 480, and the simulation can be reset using the reset signal. The start signal indicates whether the simulation has started, and the alt signal provides the current altitude position of the flying object.

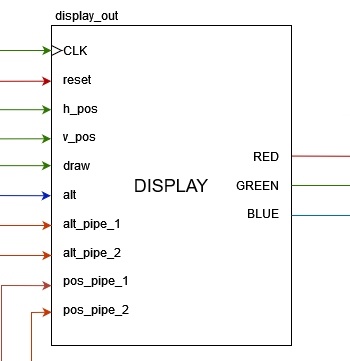
1. **FREQENCY DEVIDER**



The frequency divider resource takes an input clock signal (clk), a division factor (div), and outputs a divided clock signal (divided\_clk).  It counts the number of clock cycles and generates a divided clock signal by inverting the output clock (new\_clk) after the specified number of clock cycles (div) has passed. The divided clock signal (divided\_clk) has a lower frequency than the original clock signal, determined by the division factor.

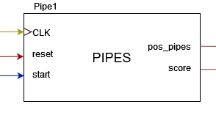
  Two instances of the module are used to generate two different divided clock signals (clk\_50 and clk\_191).

1. **DISPLAY**



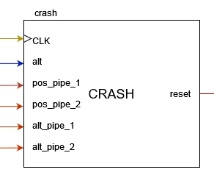
The DISPLAY resource generates color signals (Red, Green, Blue)  based on the position and object detection. It assigns different colors to the bird, pipes, grass, and background depending on the detected objects and the draw signal.

1. **PIPES**

****

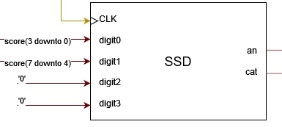
The PIPES resource represents a game with moving pipes and a score system. The position of the pipes and the score are updated based on the clock signal and certain conditions during the game. If the position reaches zero, it is set to a teleport position (pos\_teleport). Two instances of the module are used to generate two different sets of pipes.

1. **CRASH**



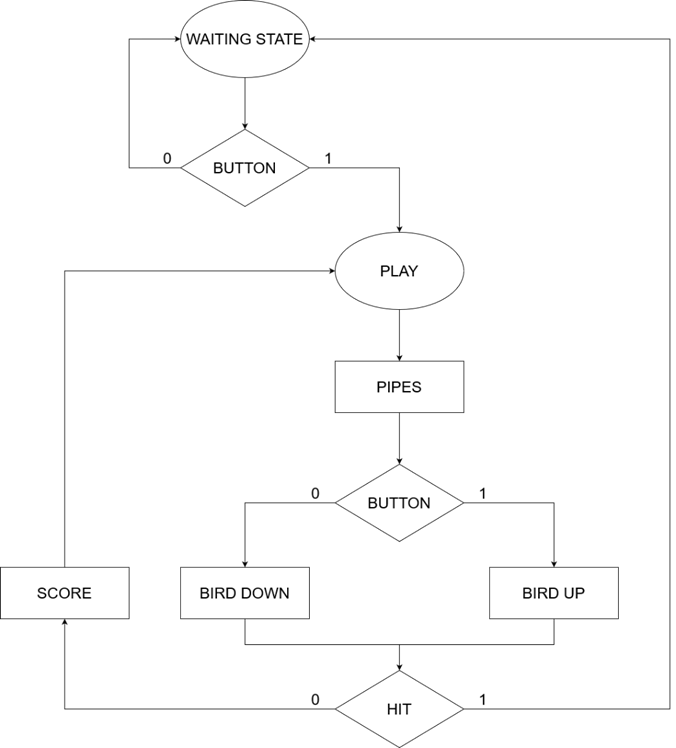
The „CRASH” module checks for collisions between a bird and pipes, and it generates a reset signal to indicate when a collision has occured.

1. **SSD**

****

The SSD module controls the four-digit seven-segment display by sequentially selecting each digit and assigning the appropriate input value and control signals. It will display the score.

* 1. **Flowchart**



A white circle with black letters

Description automatically generated with low confidence

Represents the states. A state represents a moment of time (a period).

A white and black diamond with black text

Description automatically generated with low confidence

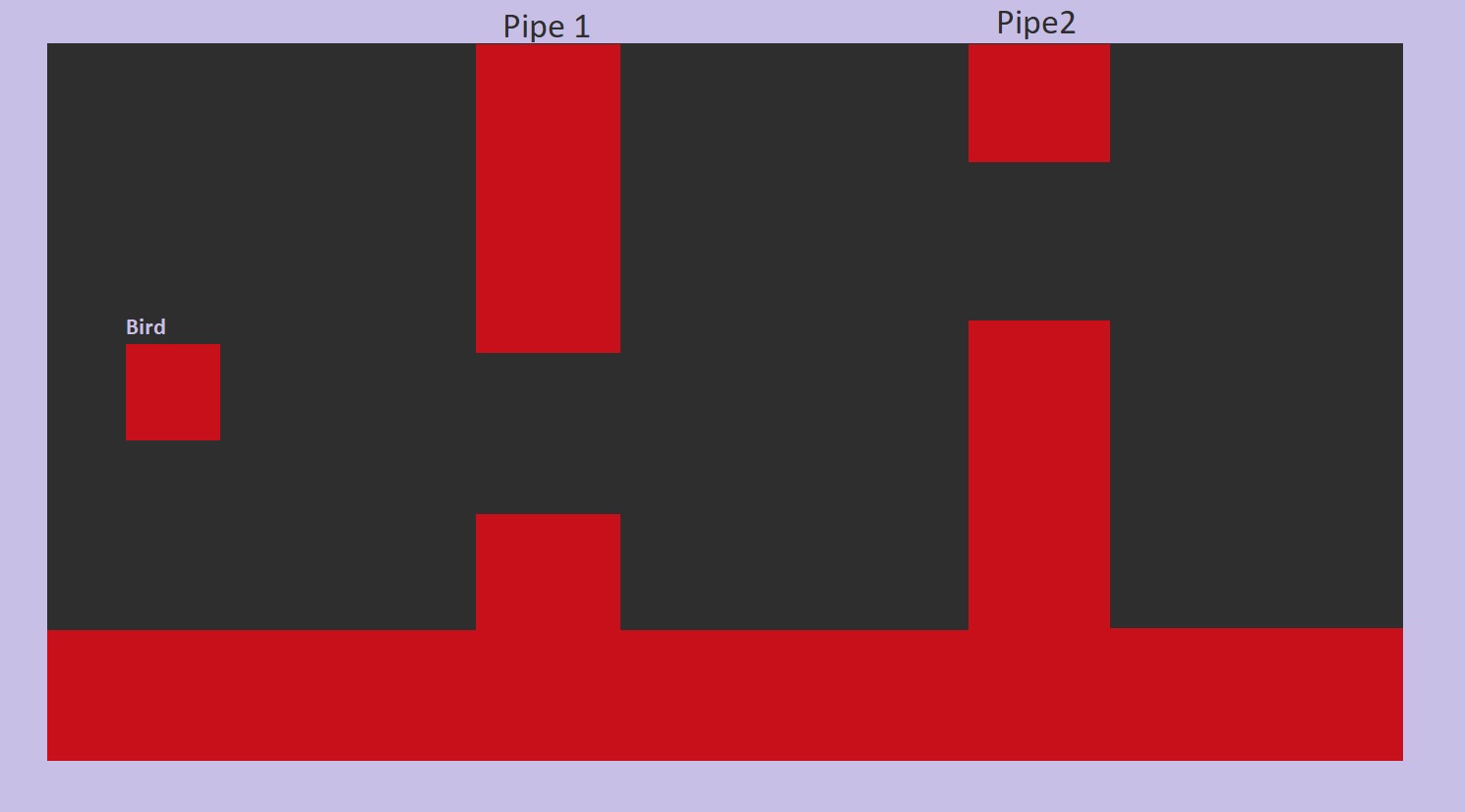
Represents the decisions made in each state.

A black text on a white background

Description automatically generated with medium confidence

Represents the outputs generated in each state

1. **Utility and results**

****

A state diagram, also known as a state machine diagram or statechart diagram, is a graphical representation of the states, transitions, and behavior of a system or process. It is a useful tool for modeling and understanding the dynamic behavior of complex systems.

The initial state is the "WAITING STATE." In this state, the game is waiting for the player to press the button to start the game. The waiting state ensures that the player has time to prepare before the gameplay begins.

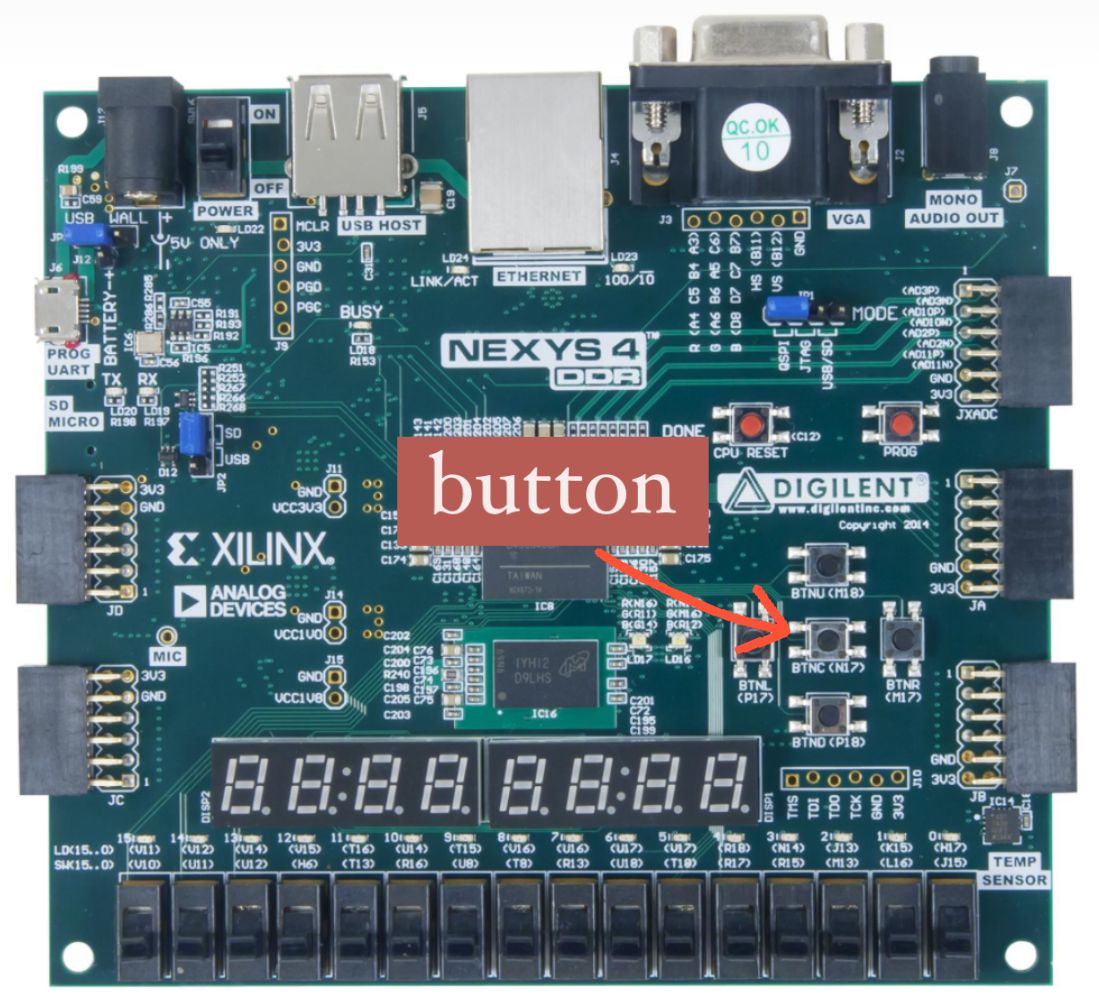
Once the button is pressed, the game transitions to the "PLAY" state. In this state, the pipes start moving from right to left, creating gaps that the player needs to navigate through. The movement of the pipes is typically synchronized with a clock signal to control the speed and timing of their movement.

Simultaneously, the little square representing the bird moves up and down based on the pressing of the button. When the button is pressed, the bird moves upward, and when the button is released, the bird gradually descends due to gravity. The movement of the bird is controlled by the player's interaction with the button, and it provides the means to navigate through the gaps between the pipes.

The gameplay continues in the "PLAY" state until a collision occurs between the bird and a pipe or any other predefined condition that would indicate the end of the game. At that point, the game transitions to a "GAME OVER" state or a similar state to indicate the end of the game and display the player's final score.

Overall, the "PLAY" state in the Flappy Bird game is responsible for controlling the movement of the pipes, as well as the vertical movement of the bird based on the player's interaction with the button. This state ensures an engaging and interactive gameplay experience for the player.

In our case, outputs are positioned after the decision, it means that the output also depends on the internal state and the inputs, so we have a Mealy automaton.

****

1. **Further development**

In order to enhance the quality and appeal of our projects, we aim to elevate them to a higher level by introducing a more vibrant and diverse color palette. Additionally, we plan to go beyond the current simplistic square shape and replace it with a personalized, customized representation of a small bird.

1. **Technical justifications for the design**

Our inspiration for this project was based on the popular game, Flappy Bird, but having a much more minimalist approach for the creation of it. We were looking forward to creating a game that is as close as possible to the one that lots of people love, without removing the details that we thought were most important.

For the creation of the game, and obviously for being able to play it, we decided to have a square, that represents the bird, and two pipes that our square has to go through. For simplicity reasons, the two pipes repeat themselves until our square has crashed into one of them, or into the floor or the sky, case when the game restarts, and the score is set back to 0.

The square is controlled by the central button of the nexys 4 board, going up once with each press of the button, and falling down whenever the button is not pressed. The score is displayed on the SSD display.