Flappy Bird Interim Report

Darren Ho, Luke Ryan, Thomas Yeh

Game Strategy

The game Flappy Bird involves the player surviving as many obstacles as possible until they die. The goal is to aim to score a higher score through each playthrough. The simple motion of the bird (player) is dictated by the clicking of a mouse, which moves the bird vertically (and slightly horizontally) through the sky as oncoming pipes (obstacles) approach. Each time the player passes between pipes, their score increases, resulting in one point per past obstacle, until the player ultimately hits an obstacle (pipes, ground, sky) and dies, ending the playthrough. The player's goal is to survive as long as possible and gain as many points as possible.

The version of the Flappy Bird game we will make will consist of two modes: training and single-player mode. Training mode will allow the player five lives to practice at the lowest difficulty level in training mode until the bird dies. In single-player mode, the game will usually play, where the player will only have three lives, and the difficulty will increase the longer the player survives (based on time or points acquired). A player will lose a life or die when they collide with either the pipes, ground or sky. The single-player difficulty increase will be shown by the rate at which the pipes come towards the player, getting faster as the game progresses. This will challenge the players as they play, but to help the players, there are gifts to aid them. The player can pick up a randomly generated health pack, refilling one life slot (if there is one to fill) or a randomly generated star (star power-up is subject to change), giving them invincibility and allowing them to pass through any oncoming pipes (but not ground or sky). Power-ups will be activated in both modes, and the mode itself will be determined by one of the DIP switches and push buttons.

Design Specifications

The game will be implemented in VHDL and with a DE0-CV board. The player will control the game via a PS/2 mouse, DIP switches, and push buttons on the DE0-CV board. The game will be displayed on a VGA screen at a 25Mhz pixel rate with 640 x 480 resolution. The screen will be able to produce 60 frames per second (to avoid flickering).

Planning

We plan to create a randomised pipe generator that consistently brings pipes from right to left of the screen as the player plays. This will be done with an LFSR component. We will make a component to control the bird's movement, detection of collision, or pickup of power ups based on data sent from the PS/2 mouse and FPGA. For scoring, we plan to use the 7-segment display to show a real-time counter of how long the player survives. The VGA can also display an in-game counter for how many pipes the player passes through. To switch between the states of the game, we will implement an FSM machine, which will be a Moore machine to handle the different states of the game.

Finite State Machine

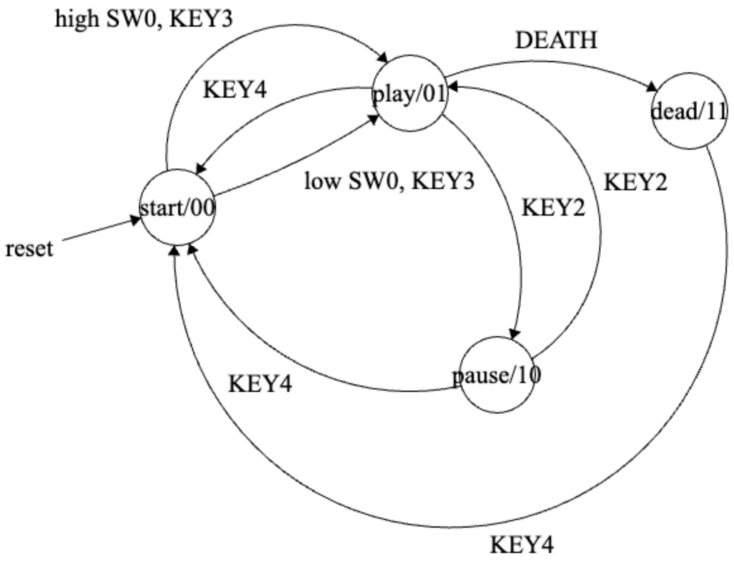
We have decided to implement a Moore state machine. The inputs will be "SW0" of the DIP switches, "KEY2", "KEY3", "KEY4" (reset) of the push buttons and "DEATH" (this will indicate that the player has hit an obstacle). The output of the Moore state machine will be a two-bit signal called "state_out", representing the game's current state.

- When state_out = "00"

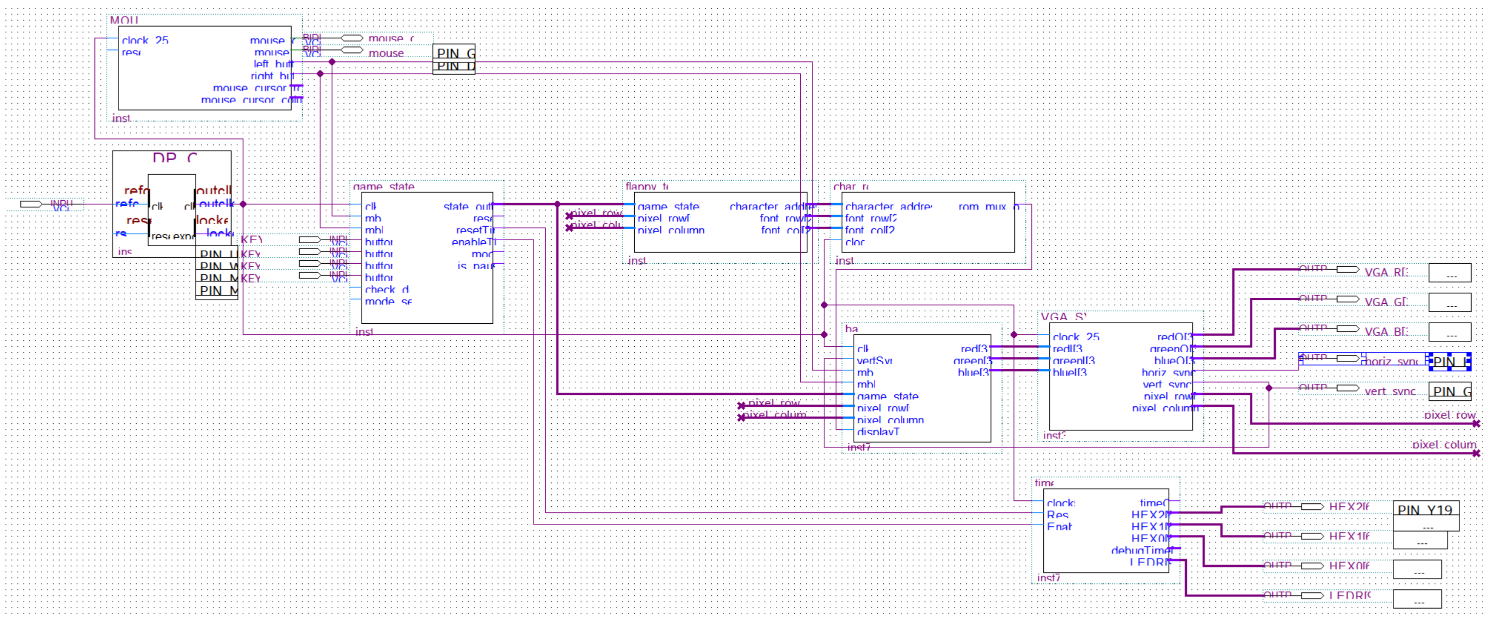
- The game will be in a “start” state, displaying the player with a home screen where a title and the two game modes will be shown to the player. From here, the player can switch SW0 high and then press KEY3 to go to “training” mode, or SW0 low and then press KEY3 to go to “single-player” mode.

- When state_out = “01”
  - The game will be in a “play” state, either of the two selected game modes from the home screen: training or single-player mode. In this mode, the player can control the bird with the PS/2 mouse, moving the bird vertically as the pipes come from right to left of the screen. The player can press the KEY2 push button to transition to the “pause” state.
- When state_out = “10”
  - The game will be in a “pause” state, where the player and the pipes will be suspended until the KEY2 push button is pressed again to transition back to the “play” state (will need to make sure there is a checking function, so no overlap occurs when pressing KEY2)
- When state_out = “11”
  - The game will be in a “dead” state, displaying a game over the screen to the player and showing their score (amount of pipes passed and time survived). From here, the player can press KEY4, bringing them back to the “start” state to begin again.

High-Level State Machine



main_dd.bdf



game_state
state_out
VGA_S
clock_25
redO
greenO
blueO
horiz_sync
vert_sync
pixel_row
pixel_column
HEX2
HEX1
HEX0
LEDR