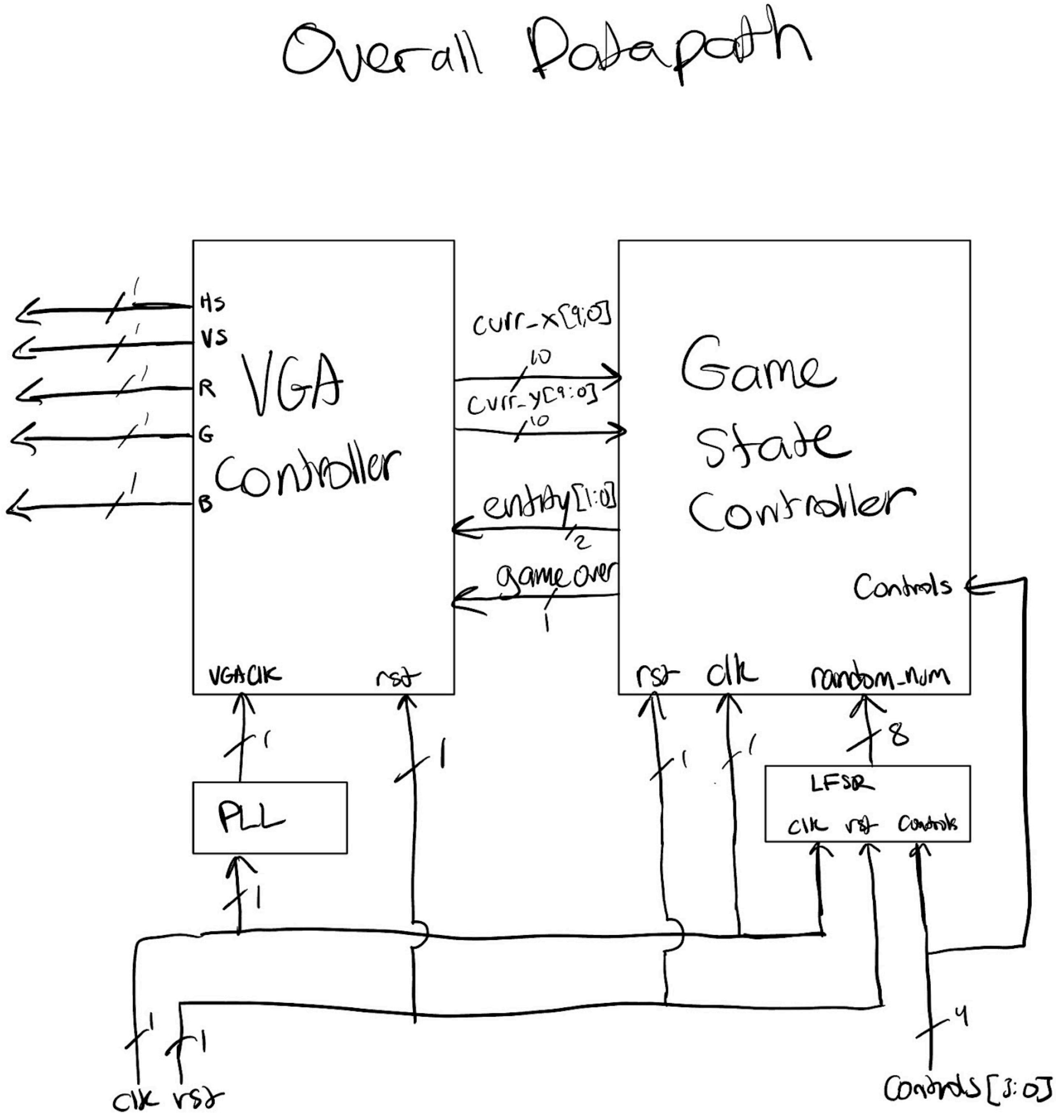
Nick Ward (npward)
18-624 - Intro to Open-Source FPGA and ASIC Chip Design
3/25/25

EX8: Become STA: The Trilogy

Task #4: Project Milestone

1. **Datapath schematics of core modules in your design. These should convey the way your modules function and are connected to each other.**



Compared to my last diagram, instead of having to send the entire game board over to the VGA controller in order to display it, I instead will be passing smaller values that represent different parts of the game state.

Communication between VGA controller and Game State Controller:

- As the VGA controller iterates through pixels on the screen, that X,Y value is sent to the game state controller. This returns an entity value back to the VGA controller. All entities in the game can fit inside a 2 bit value:

00 - Air
01 - Snake
10 - Fruit
11 - Wall

- Based on this value, the VGA controller will change what RGB value is set at that pixel.

Wall: Blue
Snake: Green
Fruit: Red
Air: White or Black

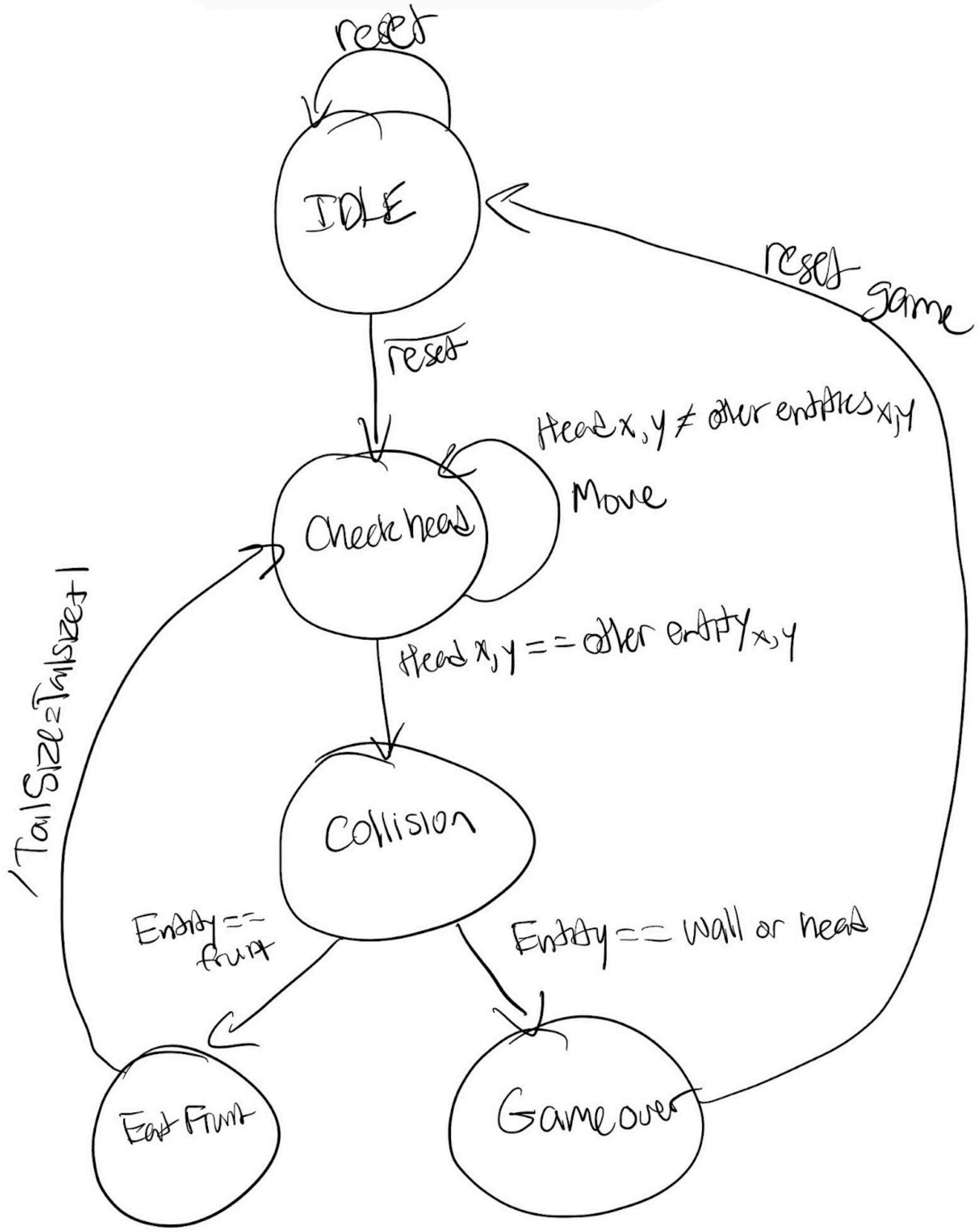
- The gameOver is also sent to the VGA controller so that a game over screen can be displayed, letting the player know that the game is finished.

A PLL will be used to change the clock from 50MHz to the 25MHz that the VGA controller expects. If we were actually fabbing out a chip, I could make a digital frequency divider for this functionality instead, but this can be a good opportunity to implement a PLL in my design which I have not done before.

The LFSR will be used to generate a random position for a new fruit after the snake consumes the previous one. The time between control inputs could be the seed in order to make things a bit more random!

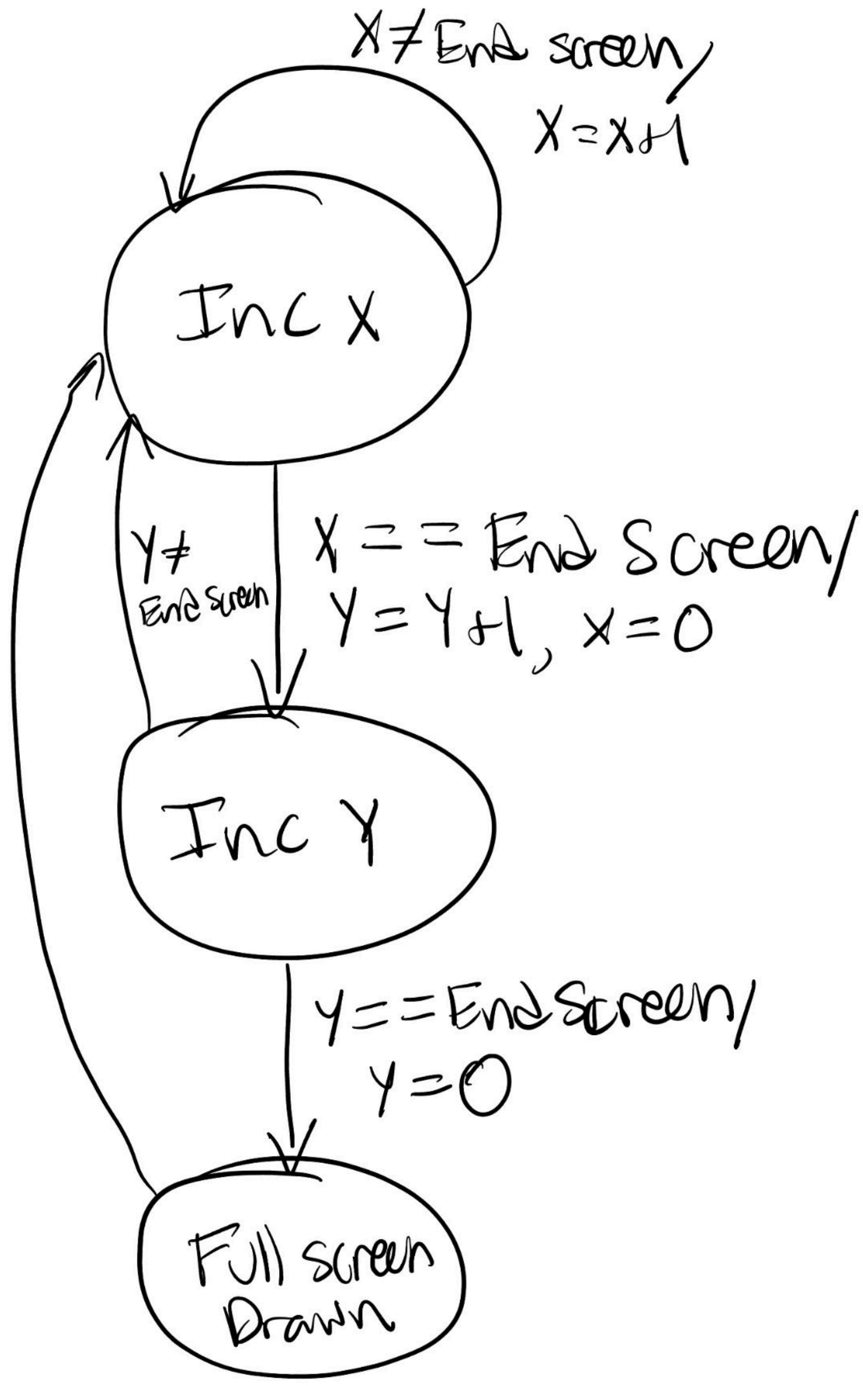
2. **FSM state transition diagrams for any nontrivial processes in your design. Do not worry about details relating to status and control points.**

Basic Game State FSM



This is the major FSM that will control the game state. The head will either be colliding with nothing (air) or will be colliding with another entity (its tail, a wall, or a fruit). Depending on what the collision is, this will do different things as discussed above.

VGA Drawing



This is the FSM for iterating through the pixels on the screen in the VGA controller. It basically works that when X hits the end of the screen, it will reset back to 0, but the Y will increase by 1. This will repeat until Y hits the end of the screen, and then Y gets reset back to 0, and a new frame is drawn.

I would include more FSMs, but I can't really think of any other big FSMs that will be necessary other than if-else statements in other blocks.

I apologize in advance for the lack of detail on some of these diagrams. It would be very helpful feedback to hear if you think I am overlooking something critical or oversimplifying a complex challenge.

3. **Some thoughts about testbench strategies and specific tests relating to your design**

One way to do testing is to use the CocoTB and a drawing function with python to be able to display what would be visible on the screen using the data from the VGA controller. This may be a bit challenging, but could be worth it to not have to reupload the design to the FPGA every time that I want to visually verify how the game is running as expected. This can also be used as a way to unit test the VGA controller as well without connecting it to the other components.

There are two ways of testing game play. The first is to set up a sample game where all interactions are tested: moving in all directions, eating a fruit, hitting a wall, and hitting itself. Another way is to test smaller parts of the design independently by forcing values into the RTL and reading output values. The second option is a bit easier to test the design more granularly, whereas the first will be harder to implement due to how controls are very time dependent, and may require limiting randomization or making a certain "test game" in order to make for a comprehensive test that won't always work in the same way.

However, these don't have to be mutually exclusive. I will most likely make small unit tests for each of the blocks as I create them, and once the game is complete, I will make an overall gameplay test that activates all of the different conditions in the game and checks their values within the game state.