Task #5: Project Proposal

Snake Game:
Total I/O Available: 38
Pmod VGA: 16 I/O pins (24 left)
Controls: 4 minimum (Up, Down, Left, Right) (20 left)

This project will be split between three parts:

1) VGA Controller

In terms of flexibility on scale, the VGA controller is certainly the least flexible, considering each of the pins needs to be driven by something. Implementing this first will most likely set the course for how complex I can make the other components. I will most likely utilize the Pmod VGA: Video Graphics Array from Digilent (https://digilent.com/shop/pmod-vga-video-graphics-array/). This seems to be a pretty common choice for integrating a VGA port for FPGA projects. I've also seen a version made for TinyTapeout, but considering I am not constrained I/O wise, this seems like the best choice.
VGA uses separate wires to transmit the Red, Green, and Blue component signals as well as vertical and horizontal synchronization signals and requires no fancy protocols like HDMI or DP. The job of the VGA controller is to take in information about the game state from the Game State Controller and properly display it on screen.
Inputs:

- Game state information from the Game State Controller.

Outputs:

- RGB Color Signals
- Horizontal Sync Signal
- Vertical Sync Signal

2) Game State Controller

This will control and keep the state of the game as it plays. This is the most scalable and most flexible part of the design. Depending on how much space (or time) I have, features can be scaled back or added.
Things the game will need to keep track of:

- Snake position
- Snake length
- Food placement
- Collisions between food and walls (determining game over)
- Grid layout (walls and food)
- Color?

Due to the relative simplicity of snake, the game state can be thought of as an array, where each element can represent either: air, snake, food, or a wall. The more involved part is being able to quickly make decisions on what the next board state will be based on the inputs and collisions between objects on the game board. This game state will need to be communicated with the VGA controller to accurately display the game as it is changing.
Inputs:

- Button controls (control the direction the snake is advancing towards)
- random number(s)

- Clock
- Reset

Outputs:

- Either the grid state (array of the board that needs to be displayed) or individual specific characteristics (position of snake, food, walls, etc… that will be combined to display the game)

3) LFSR Unit

Utilize LFSR to create pseudo-random numbers to make each playthrough of the game more interesting.
Opportunities for randomness:

1. Food location
2. Snake spawn location
3. Optional: Game board layout? (Maybe a selection of certain stages with different obstacles?)

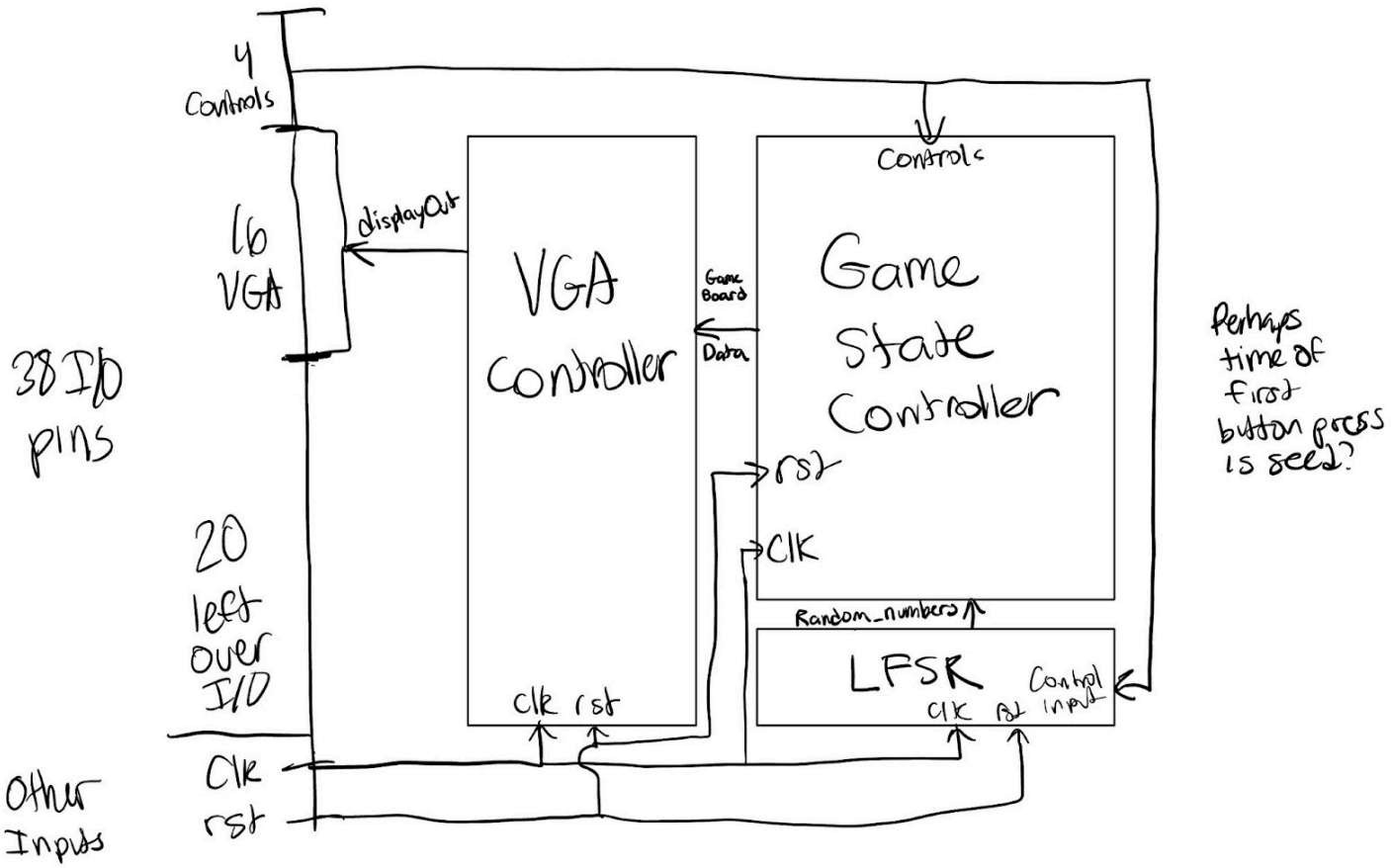
LFSR only generates one random bit per clock. For an N-bit random number, then I could either sample LFSR every N cycles, or have N LFSRs in parallel with different seeds to generate the random number. If I recall correctly, I believe a NES video game used the number of frames that the user was on the title screen as the seed. I could use a similar approach and implement a counter that counts every clock and when the user first touches a button, that count is provided to the LFSRs.
Inputs:

- Button controls?
- Clock
- Reset

Outputs:

- A N bit random number

The following is a high level diagram of the design.


4
Controls
16
VGA
displayOut
VGA
Controller
Game
Board
Data
Game
State
Controller
Controls
Perhaps time of first button press is seed?
38 I/O pins
rst
Clk
20
left
over
I/O
Random_numbers
LFSR
Control
Clk
rst
Input
Clk
rst
Other
Inputs