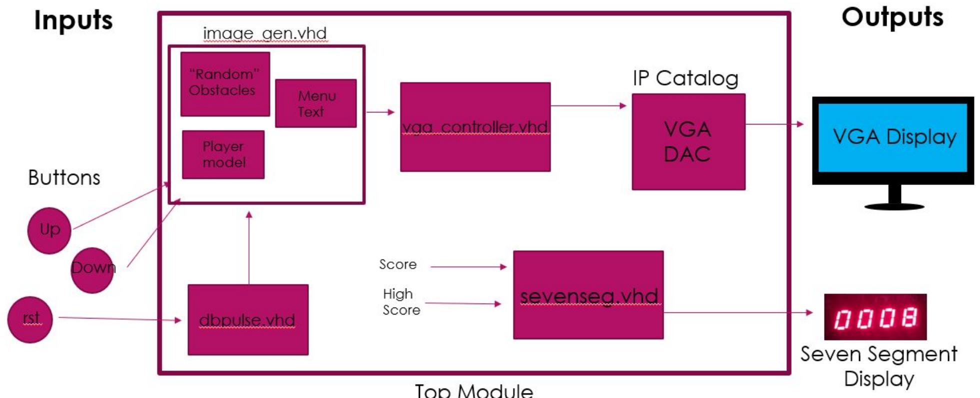
Hexadodge: Mid Semester Review

By: Zach Williams, Robert Grady Williams, and Clay Subert

Overview

- Firstly we began working on a functional VGA module, that drives our display
  - This is a key function in our final production, as it allows us to display our game to the player
  - The functional VGA display will allow the rest of our project to output the data in the form on moving image
- Next, we need to implement the character model, as well as our obstacle generation
  - This will be the base of the game. The player will actively control the player model to avoid obstacles to obtain points
  - These has been the most challenging processes to develop as we having been developing an random number generation of the obstacles to allow for unpredictable gameplay
- Finally, we need to implement the scoring system
  - This will be done through a time based reward system. The goal is to “survive” for a longer period of time to increase score



Zach’s role:

- I worked primarily on the VGA module. I focused on getting the display to have the correct size and display for the game. While implementing the VGA, I made sure that the display could be used with the appropriate ports to output visual data from the Basys3 board
- Additionally, I assisted with the physics for the player model. I helped Clay with the physics for the player model, as the physics was probably our largest undertaking

Grady’s Role:

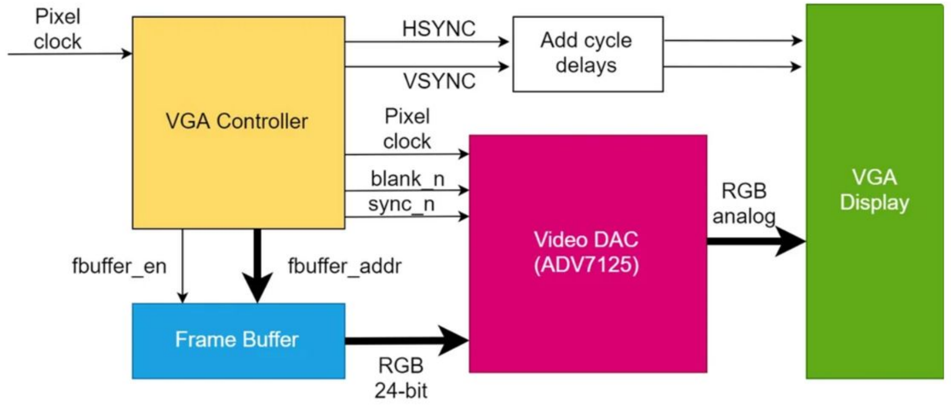
- My primary task was the implementation of the player model and obstacle models. I designed the working models for the display
- I also completed the scoring system, which keeps track of the players active score. When the game ends the score remains on the seven segment display, until the game is restarted
- I also worked the physics for the obstacles with Clay. We created the movement for the obstacles, and how they interact with the player

Clay’s Role:

- I am the physics lead, and have worked primarily on the random number generation for the obstacles.
- Additionally I worked with Zach on the physics and movement for the player model. This includes the vertical controls for the player on the Basys3 board.
- Similarly I worked with Grady on the physics for the obstacles and the motion of the obstacles. This includes the reset state of the obstacles, which will then be regenerated.
- In addition to these tasked, I have worked on the collision of the player and obstacles to end the game.

VGA module:

- We implemented a VGA module that was created by MIT, and is under MIT license
- This drives our display through the vga port, and allows us to display our visual output to the display
- We had to modify the code, to format the appropriate sizing and work under our constraints



Player and Obstacle Models:

- We developed the visual modeling of the player models and obstacle models
- We then utilized theses models to display them to the monitor
- While creating the models, we have gone through several different models, and we have currently been working with a geometric styling, which gives a simplistic style to the game
- We are currently working with a hexagon player model, and a square hitbox
- Our obstacles models are currently horizontal bars that move across the screen from right to left

Scoring:

- We will be scoring the player based on runtime
- We will display in hex the score on the seven segment display based on the number of seconds the game continues to run without a collision
- We will increment the score based on the run clock, and the longer the player last without a collision the score will increase.
- The score will remain until the game is restart

Player and Obstacle Physics:

Player Physics:

- Our player moves vertical up or down at the press of a button
- This means that the user must keep the player between the bounds of the display or the game will end

Obstacle Physics:

- The obstacles will move from the right border to the left border
- When an obstacle collides with the player, then the game will end

Randomization:

- The primary use of randomization is in our obstacle generation
- When obstacles are generated, they should be pseudo-randomly placed along the right border, which will cause the obstacles to vary between each game, and differ the patterns greatly
- We have experimented with couple different implementations, and the one that we are attempting currently is to place the obstacle with a clock based offset vertically when it is reset to the right border
- This may change if we cannot produce satisfactory results with this method

Current Procedure Updates:

- We began using GitHub to share our code, however, this seemed to not be all that efficient as we could not easily implement the files with Vivado.
- Rather we have been sharing our files directly, as this has been easier to work with
- We still plan on keeping all the files on GitHub as we finish our project to store the project in a repository, however, we just won’t use the code sharing aspect
- Another modification we have made as a team is we have started segmenting our tasks
- We began working together on many of the strategies that we came up with, however, it has been much more beneficial to delegate the tasks to work collaboratively on specific aspects of the code

What next?

- We are working to finish a functioning implementation of the game as a whole. We want to test our implementation of the game.
- This currently includes finishing the physics for the game(primarily the RNG, and collision)
- We also need to finish some minor visual bugs
- Assuming we are able to finish all of these current issues, we want to implement a UI.
- This will primarily be a Game Over screen, that alerts you to the end of the game(assuming time permits)

References:

https://github.com/alisemi/fpga-projects/tree/master/Don'tHitTheBars

https://forum.digikey.com/t/vga-controller-vhdl/12794

https://github.com/Derek-X-Wang/VGA-Text-Generator