Final Lab Report

ECE 643 – Computer Design Lab

Maze Game

Prepared by:

Andy Freshnock, Aidan Harries, and Andrew Sonner

Course Instructor:

Dr. Gruenbacher

12/15/2023

Introduction

For our final project in ECE 643, our team has developed a digital version of the traditional ball bearing maze game. The project required us to integrate both hardware and software components to achieve this goal. Our approach was to replicate the physical experience of the maze game in a digital environment, using the tools and knowledge we have acquired in this course. The outcome of our work is a functional, digital maze game that combines the use of an accelerometer for navigation, a VGA display for visual output, and additional features like a stopwatch and pause switch to enhance the user experience.

Figure 1 – Ball Bearing Maze Game

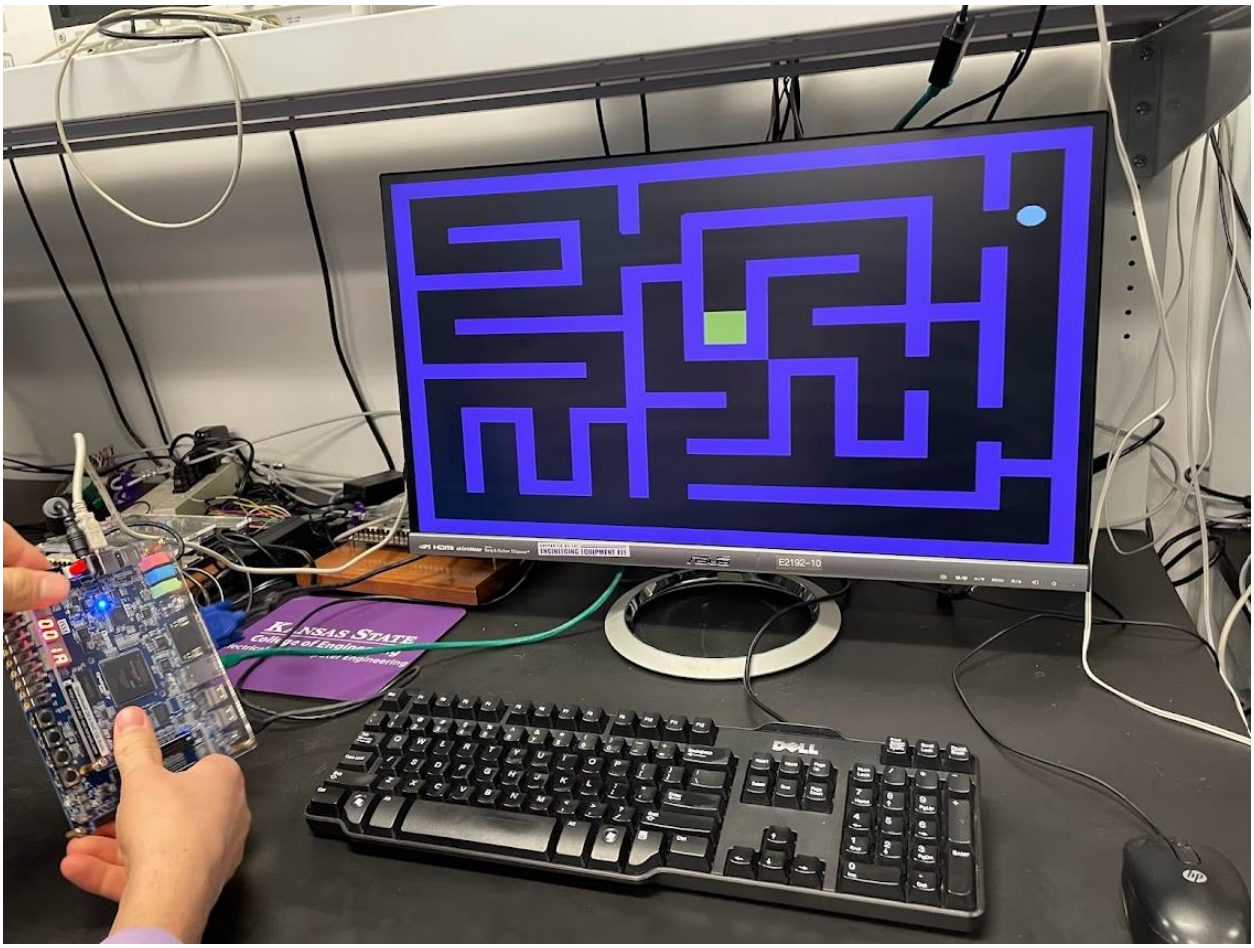

Figure 2 – Finished Game

Overall Design Concept

The primary focus of our design was to translate the ball bearing maze game into a digital format. In doing so, we used an accelerometer to simulate the physical act of tilting the maze, allowing players to navigate the digital ball through the virtual maze displayed on a VGA screen. The ball also had logic built into its movement to simulate a similar feel to the typical movement physics of a rolling ball. This approach ensured that the fundamental challenge and appeal of the original game were preserved in our design.

To further enhance the gaming experience, we integrated additional features such as a stopwatch for time tracking and a pause switch for gameplay control. These elements were designed to complement the core gameplay, maintaining the simplicity of the original while leveraging modern digital interfaces.

System Diagrams

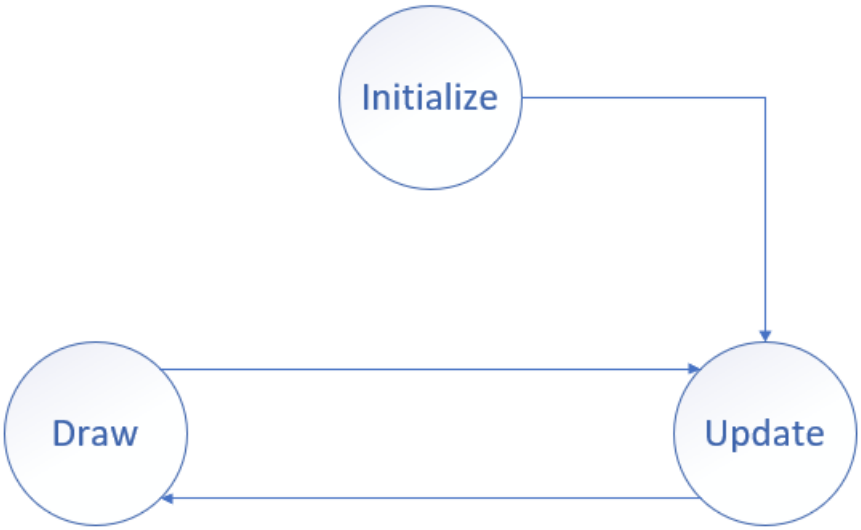

Figure 3 – Basic State Machine for Arm CPU

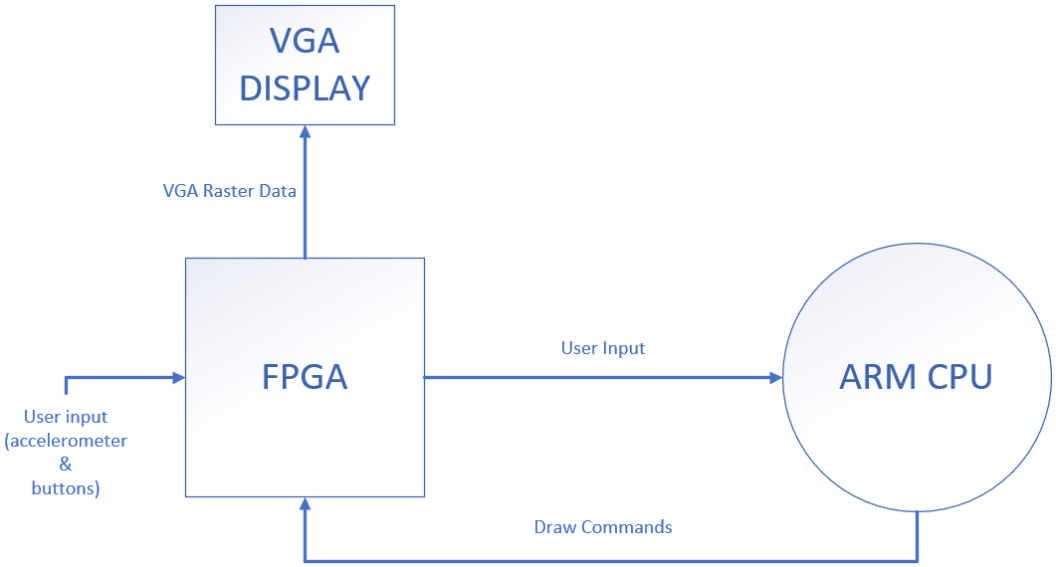

Figure 4 - System Diagram

Desired Outcomes / Goals

Minimum Goals:

- **Ball Control:** Smooth movement of ball on VGA display via accelerometer.
- **Maze Display:** Clearly present maze layout on VGA screen.
- **Collision Mechanics:** Implement basic collision detection with maze walls.
- **Scoring and Timing:** Incorporate system for tracking scores and game duration
- **Reset Button:** Add reset button for game restarts and timer reset
- **Graphic Quality:** Use double buffering for VGA graphics.

Stretch Goals:

- **Hazards:** Introduce obstacles to increase difficulty.
- **Interactive Menu:** Develop an in-game menu.
- **Advanced Collisions:** Implement elastic and diagonal collision physics.
- **Complex Mazes:** Design mazes with diagonal paths or curves.
- **Multiple Levels:** Create more maze levels for progressive gameplay.
- **Animations:** Enhance game with animations.
- **Collectables:** Add items withing maze for players to collect.
- **Level Selection:** Implement level select and save feature using SD card.
- **Realistic Physics:** Add inertia to ball movement.
- **Audio Elements:** Integrate sound effects and background music.
- **Score Records:** Keep track of high scores and previous attempts.

Design Implementation

Hardware:

We handled the following aspects of our design within the hardware:

- Timer utilizing the hex display.
- LEDs
- Reset push button
- Pause game slider

The timer was created using a built in 50 MHz clock that was divided down to a single Hz. This value was then driven to the hex digit display (only the first 4). We also implemented a push button that resets the clock, as well as the entire game/board. The slider switch has similar functionality, as it can pause the hardware timer and suspend functionality of the entire game. The LEDs remained off the entirety of the gameplay until the game logic detected the ball was in the winning square, after which they were all turned on.

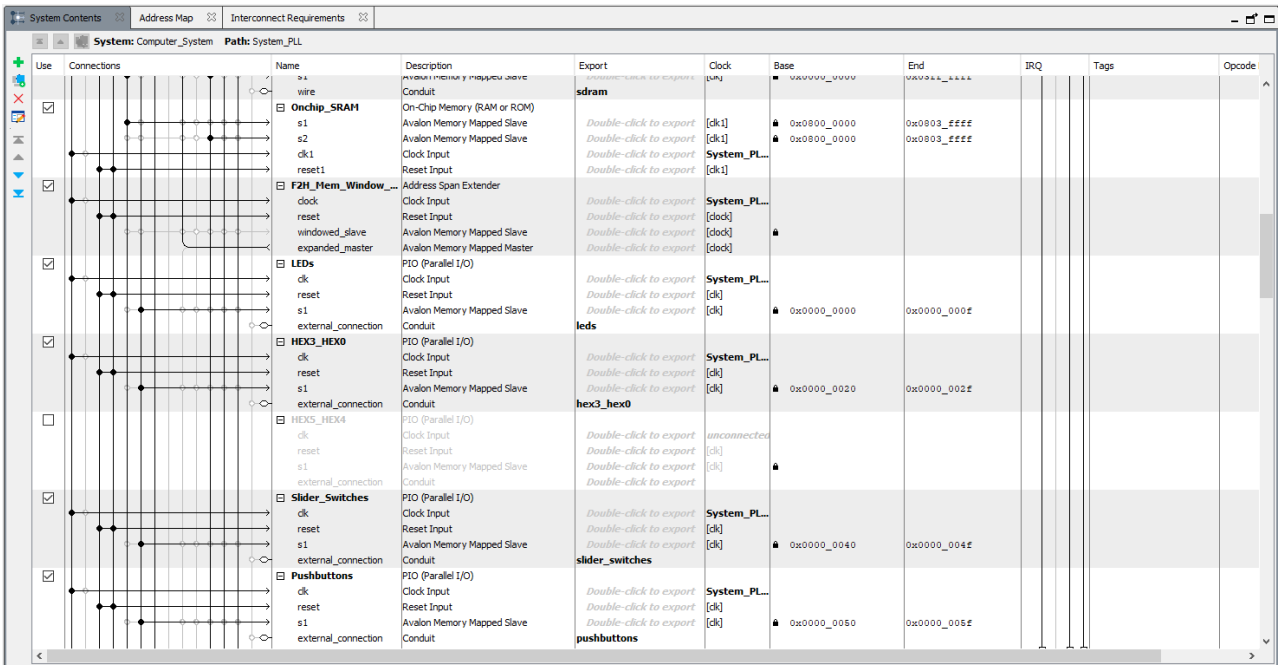

Figure 5 - Hardware Implementation

Software:

The core of the game is a tile map. This allows us to partition the screen into different tiles that have different properties associated with them. This was critical as it allows us to control the interactions between the user-controlled ball and the game world. Each tile is assigned a number which controls the color of the tile and its properties. This design philosophy allows us to add more tiles that have different interactions with the user. The tiles that we were able to implement were the wall tile and the goal tile. The rest of the software tackles interactions with FPGA. This includes monitoring the accelerometer and the other buttons on the FPGA. The VGA drawing code is also included.

For the base addresses we utilized PIOs that were already declared within the University Computer.

Button Base: 0xFF200050

Slider Base: 0xFF200040

Validation and Testing

Minimum Goals:

- **Control Ball with Accelerometer:** Accomplished
- **Display Maze Layout on VGA:** Accomplished
- **Basic Collision Detection**: Accomplished
- **Scoring and Timing:** Accomplished
- **Reset Button:** Accomplished
- **Double Buffering for VGA Graphics:** This goal was not met. Our team faced challenges in understanding and implementing the buffering mechanics. Specifically, writing to each buffer and executing buffer swaps proved to be more complex than anticipated.

Stretch Goals:

- **Hazards:** Not met.
- **Interactive Menu:** Not met.
- **Advanced Collisions:** Not met.
- **Complex Mazes:** Not met.
- **Multiple Levels:** Not met.
- **Animations:** Not met.
- **Collectables:** Not met.
- **Level Selection:** Not met.
- **Add Inertia to Ball Physics:** Accomplished
- **Audio Elements:** Not met.
- **Score Records:** Not met.
- **Pause Switch for Game and Timer:** Accomplished

Summary:

We successfully implemented core features like accelerometer-based ball control, maze display, collision detection, score/time tracking, and a reset button. However, we faced challenges with double buffering for VGA graphics, which proved more complex than anticipated and remained unachieved. Focusing more on perfecting the basics, we managed to add realistic ball physics and a pause switch, but other stretch goals like advanced hazards and multiple maze levels were not developed within our timeline.

Individual Contributions

Throughout our project, each team member played a vital role in bringing different elements of the game to life:

- **Andy:** Developed the visual interface of the game, including the maze display and tile map. Additionally, Set up our GitHub repository for collaboration and version control.

- **Aidan:** Focused on the general game logic, working on the foundational mechanics that drive the gameplay and contributed to the development of the collision logic and visual interfaces.

- **Andrew:** Was responsible for integrating the score counter and other FPGA logic and making sure that the hardware interfaced well with our software.

- **Team Efforts:** Several components of the project were worked on collaboratively. As a team, we worked on:

    - Drawing the ball on the VGA display
    - The overall architecture of the game
    - The collision logic

Reflections and Improvements

Looking back on our project, we realized that our approach to time and resource management could have been better optimized from the start. A significant challenge we faced was the screen flickering issue. Our initial solution was to implement double buffering, but this turned out to be more complex and time-consuming than we had anticipated. In hindsight, categorizing double buffering as a stretch goal rather than a primary objective might have been a wiser choice. This would have allowed us to channel our efforts more effectively towards other critical aspects of the project, like refining the collision logic, which also proved to be a tricky area for us.

Summary and Conclusions

As a team we were successful in implementing and achieving our minimum goals. We all learned a lot about debugging on the FPGA and how to problem solve with the constraints of the board. As most of our project was in C code, we also learned how navigate the challenges of developing a game from scratch. The university computer system Verilog code gave us access to most of the FPGA peripherals that we needed. This was critical to our success as it allowed us to focus on developing the game instead of having to implement controls in Verilog. We implemented a hardware timer and display to keep track of times for the player.

The project is designed with the implementation of more features in mind. This would allow us to continue developing the project if we wanted. As most of the project is developed in C we could also switch the way that we display the project. As most of the drawing code for the project is separate from the game update code it would be feasible to move the project into a different development environment. Because the core of the game logic is based on a tile map it is very flexible. We can easily change the size of the tiles and map to make the game more complex and larger. Overall, we would have been more successful if we did not hit as many dead ends by trying to implement double buffering and more complex collision logic.

Appendices

(PLEASE SEE ATTACHED ZIP FILE FOR C CODE)

Timer Code/Pause switch code within the University Computer program:

```
HexDigit Digit0(HEX0, Hz[3:0]);
HexDigit Digit1(HEX1, Hz[7:4]);
HexDigit Digit2(HEX2, Hz[11:8]);
HexDigit Digit3(HEX3, Hz[15:12]);
reg [26:0] c;
reg [15:0] Hz;
always @ (posedge CLOCK_50 or negedge ar)
begin
	if(~ar)
	 begin
		c = 0;
		Hz = 0;
	 end
	else if(~SW[0] || LEDR[0])
	 begin
	 c = c;
	 Hz = Hz;
	 end
	else
	 begin
		if(c < 49999999)
		 begin
		  c = c + 1;
		 end
		else
		 begin
		  c = 0;
		  Hz = Hz + 1;
		 end
	 end
end
```

//