CSEE 4840
Embedded Systems - Project Proposal

Project Title: Air Hockey

Group Members:

Gerald Zhao - zz3427

Derrick Chen - dc3494

Zening Wang - zw3161

Xuepeng Han - xh2718

Project Concept

The project will be a two-player air hockey game implemented on the FPGA platform. The game will be displayed on a VGA monitor and controlled using a USB keyboard. Two players will each control a paddle and attempt to score by hitting a puck into the opposing goal. The screen will show the table boundaries, the two paddles, the puck, and a score display. The goal is to build an interactive embedded system: the project combines user input, Linux software, a custom FPGA peripheral, and real-time video output, a standard VGA target such as 640×480 at 60 frames per second.

At the user level, the interaction should be very simple and easy to understand. One player can use keys such as W/A/S/D while the other uses the arrow keys. Using a single USB keyboard for both players keeps the system simpler and more reliable than trying to manage two separate USB devices.

System Structure

The project should be divided into three parts: input and game logic in software, a communication path between software and hardware, and a custom hardware graphics peripheral.

The software side will run under Linux on the HPS. It will read keyboard input, keep track of the current game state, and update things such as paddle positions, puck position, score, and reset or start conditions. The hardware side will be a custom FPGA peripheral responsible for the VGA display. Its job will be to generate the video timing and render the current game state on the monitor. In other words, software will decide what the current game state is, and hardware will decide how to turn that state into pixels on the screen.

Between hardware and software will be a memory-mapped hardware/software interface: to the processor, peripherals appear as special memory locations, with status registers that report state and control registers that change the behavior of the peripheral.

If time allows, we may extend the project with features such as improved score display, pause functionality, or moving some additional game logic into hardware.

Milestones

The project will be broken down into multiple phases.

1. Register and memory map: there must be an agreed-upon way that the hardware and software will communicate with each other
2. Software side: building the game engine should include keyboard polling, puck velocity, collision math, etc.
3. Hardware side: responsible for the VGA controller, visual sprites, timing generation, etc
4. Integration: first use simple tests just to confirm that the two systems are communicating with each other. Then tying together the physics state to the VGA output, refining visuals, fixing bugs, polishing game feel and user-experience, and adding any last-minute features

The testing of software and hardware will be done independently first before integration. The software side can "mock" the hardware with *printf* statements and looking at the coordinate values. Similarly, the hardware can "mock" the software by hardcoding temporary inputs.

Anticipated Challenges and Risks

Some specific challenges that may arise or bottlenecks that may be encountered include

1. Collision physics: having the puck bounce off the surrounding walls is very easy to do, but calculating the vector of the puck after it bounces off the circular paddle may be very difficult. One possible simplification could be box-colliders if this calculation proves to be too mathematically intensive.
2. Keyboard inputs: it is very possible to run into the issue of failing to register multiple keypresses at the same time. Since this game is intended to be a two-person game with WASD and arrowkeys, the keyboard may drop inputs. An idea could be to try to use joysticks rather than a keyboard.
3. Synchronization: if the software is trying to update coordinates in-between the VGA's attempt to draw it, the user will experience glitching on the screen. A VSYNC mechanism could be something to consider.

Our overarching strategy is to not be too ambitious and first implement the basic mechanisms for the game. Afterwards, should time allow, we can add details, features, improvements, and make the game more realistic.