ECE 385

Final Project Proposal: I wanna game on FPGA

Spring 2024

Tianyang Wu / tw42

Yue Shi / yueshi6

TA: Jinghua Wang

1. Idea and Overview

Our project aims to develop and showcase a fully operational version of the "I wanna" game, implemented on an AMD FPGA board utilizing the Microblaze processor. The original gameplay is shown as below as a reference.

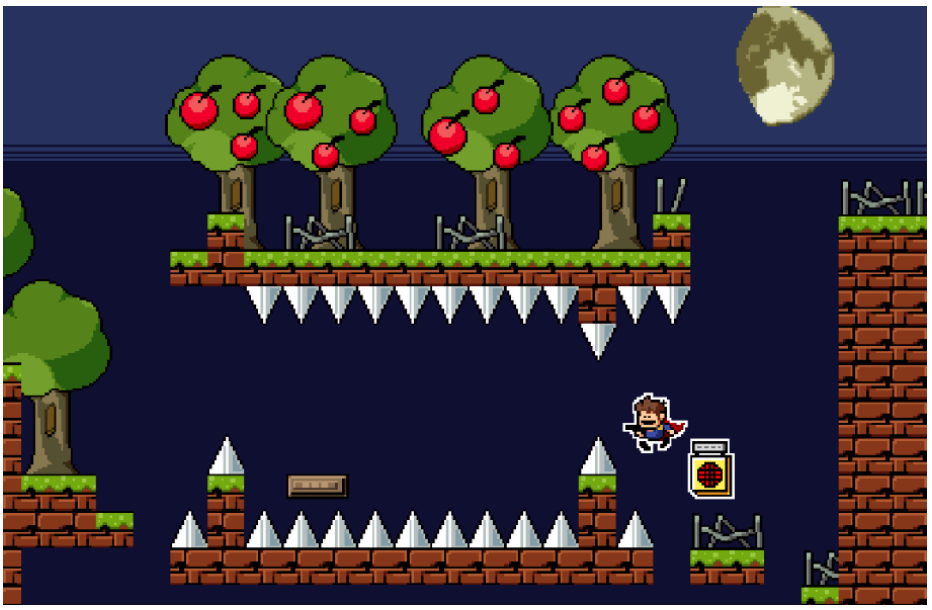
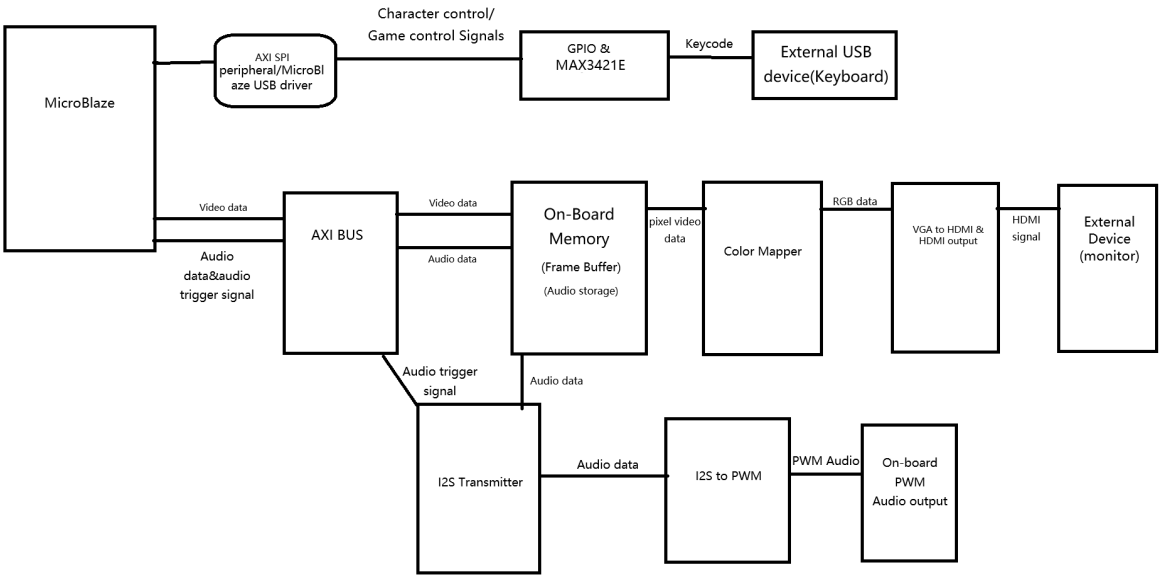


Figure 1 Example of gameplay

Our design contains critical components developed through both hardware and software approaches. The main hardware components include a MicroBlaze microprocessor, onboard memory, an audio output component with a PWM signal, HDMI IP as output to monitor, and several in-game objects that can be tracked. Those components will be realized through SystemVerilog using the FPGA's capability to handle real-time data processing and peripheral management. Some of the related resources will be shown in later sections as references. The core logic for character controlling will be programmed in C(using MAX3421E chip and SPI protocol to receive keyboard signal via USB port). The graphic output may be controlled by a frame buffer, which receives data via the AXI bus and sends it to the onboard memory. Then, the onboard memory will be used to produce video output via a color mapper.

We will start with lab 6.2 to realize our game. Our final demonstration will showcase the "I wanna" game played on a standard monitor, with inputs from a USB-connected keyboard, highlighting the successful integration of software and hardware components to recreate this classic gaming experience on modern FPGA technology.

2. Block Diagram



3. List of Features

Base features:

- Video Output via FPGA
  Implements the game's graphical user interface on a standard monitor. This involves using SystemVerilog to handle graphics processing and sending the output to a display through VGA or HDMI.

- Control Input Interfacing
  Manages the input from a USB-connected keyboard to control game actions. This includes reading key presses and translating them into game movements and decisions.

- High Score Tracking
  Implementation of a system to keep track of high scores through on-chip memory

Advanced difficulty tasks:

- Audio Output
  Generates audio feedback for game events (like jumps, collisions, and bullet sounds) using FPGA. The sound effects are expected to be synchronized with game actions to enhance user engagement.
- Multiple Episodes of the game:
  Two episodes in total. The normal level will serve as the standard gameplay environment where players get accustomed to the game mechanics. This level will feature typical obstacles, enemies, and puzzles that players must navigate through.

The other episode is boss-level. The boss level is a special episode that presents a significant challenge compared to the normal level. it typically culminates in a confrontation with a major enemy, which requires some strategies to defeat.

4. Expected Difficulty

For our final project, the expected difficulty is about 7 out of 10 (7/10), and the baseline difficulty would be 5 out of 10.

This project covers the essential aspects of FPGA, and it requires a robust understanding of both hardware and software principles and good integration skills. Since a rhythm game will receive difficulty points of 7, we consider audio output to be a medium-high hardness task. However, because audio files are huge compared to the on-chip memory we have, and we may not plan to use the external SDRAM(which may be needed for rhythm games), we would expect to add some more to achieve 7 difficulties. To maintain a certain level of difficulty, we decided to add high score tracking(5 difficulty points task) to our game. Altogether, we expect a difficulty score of 7.

5. Proposed Timeline

Timeline	Things to Do	Goal
Project Week 1 (4/8)	(4/8) Submit Final Project proposal.	Finalize project proposal. Clarify game mechanics and design. Begin hardware interfacing for control inputs.
	Do some research and make a plan for the game.	
	Get feedback from the TA. Modify the block diagram and plan.	
	Working on the Keyboard and interface.	
Project Week 2 (4/15)	Research video buffers, audio, and SDRAM(Optional).	Establish a basic video output to monitor. Define architecture for video and graphics handling.
	Start groundwork for the baseline of the game.	
	Start implementing basic video output through VGA.	
	Outline SystemVerilog modules for video processing.	
	(4/15) Lab report seven due.	
	(4/19) Mid-point Check with TA. Show tangible progress.	
Project Week 3 (4/22)	Continue development on video output modules.	Have working video output with preliminary game graphics. Audio
	Begin integrating audio output capabilities.	

	Test video and audio synchronizations.	synchronizes with game events.
	Implement basic game mechanics (movement and bullet animation).	
Project Week 4 (4/29)	Finalize all hardware components (audio, video, control input).	Complete integration of all components. Ensure stable operation and begin performance optimizations.
	Integrate a high score tracking system.	
	Begin comprehensive testing of the game (debugging and refinement).	
	Performance optimizations.	
Final Week (~ 5/8)	Demo the project and finish the final report.	Ensure smooth gameplay and functionality for the demo. Complete and submit the final report.
	(5/3) Demo date	

6. Some related resources
Audio:
https://www.hackster.io/Kampino/playing-audio-with-an-fpga-d2bc85#:~:text=The%20interface%20uses%20the%20following%20signals%20for%20data,the%20length%20of%20the%20transmitted%20data%20word.%20
https://github.com/rubinsteina13/SV_I2S_RX_CORE/tree/master
https://www.realdigital.org/doc/496fed57c6b275735fe24c85de5718c2

Frame buffer:
https://github.com/opengateware-modules/video-framebuffer