Project Design: Tetris

Prof. Stephen Edwards

Spring 2020

Arsalaan Ansari (aaa2325)

Kevin Rayfeng Li (krl2134)

Sooyeon Jo (sj2801)

Josh Learn (jrl2196)

Introduction

The purpose of this project is to build a Tetris video game system using System Verilog and C language on a FPGA board. Our Tetris game will be a single player game where the computer randomly generates tetromino blocks (in the shapes of O, J, L, Z, S, I) that the user can rotate using their keyboard. Tetrominoes can be stacked to create lines that will be cleared by the computer and be counted as points that will be tracked. Once a tetromino passes the boundary of the screen the user will lose.

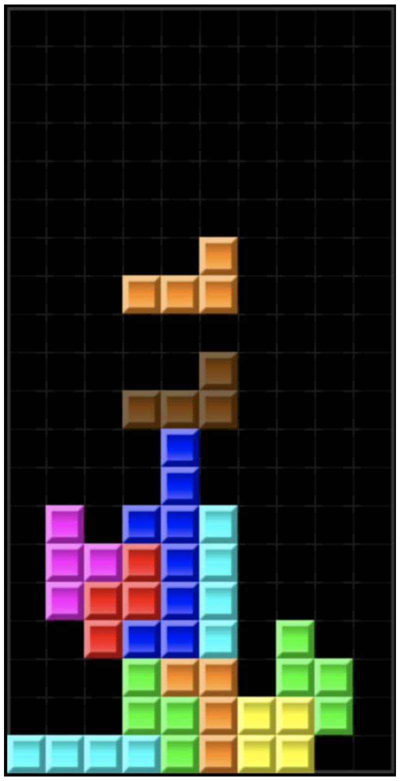

Fig 1: Screenshot from an online implementation of Tetris

User input will come through key inputs from a keyboard, and the Tetris sprite based output will be displayed using a VGA display. The System Verilog code will create the sprite based imagery for the VGA display and will communicate with the C language game logic to change what is displayed. Additionally, the System Verilog code will generate accompanying audio that will supplement the game in the form of sound effects. The C game logic will generate random tetromino blocks to drop, translate key inputs to rotation of blocks, detect and clear lines, determine what sound effects to be played, keep track of the score, and determine when the game has ended.

Architecture

The figure below is the architecture for our project

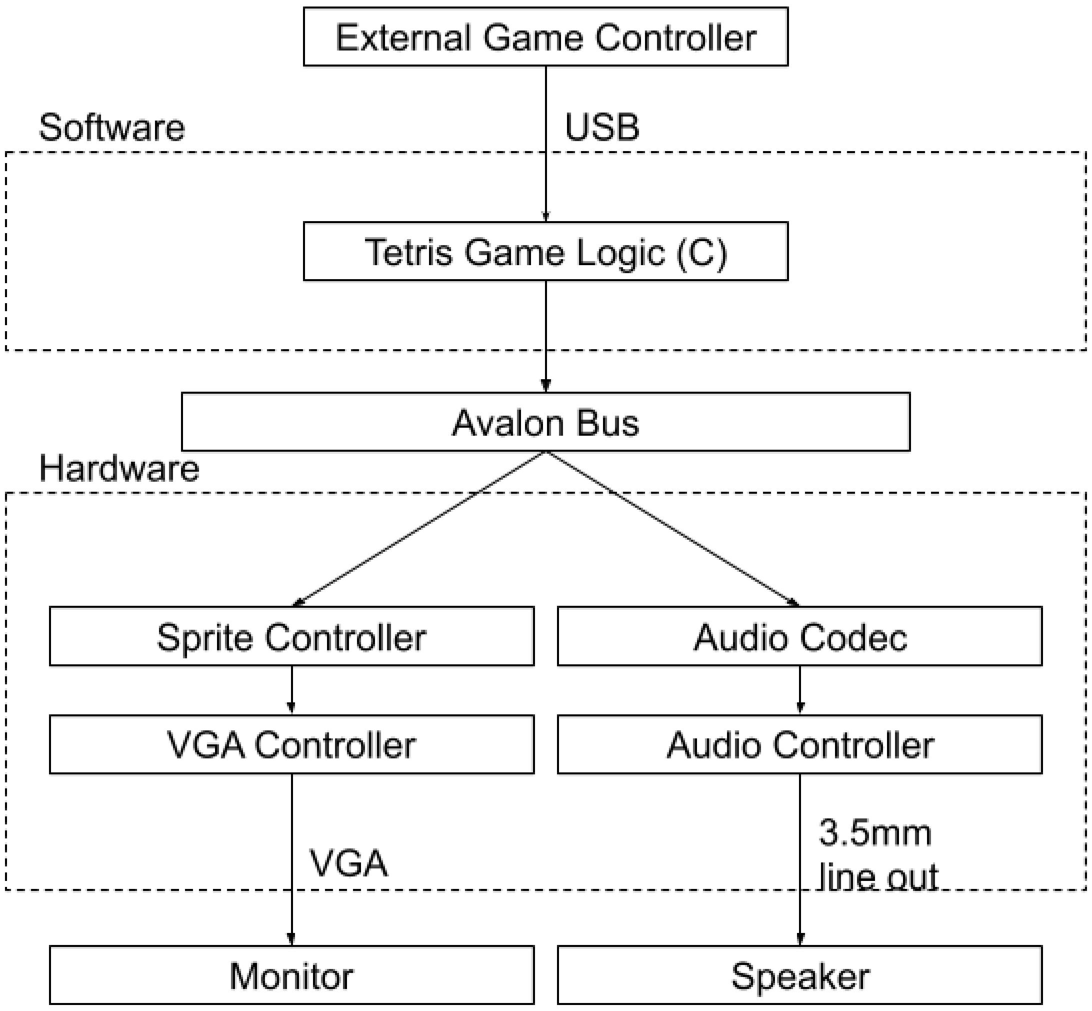

Fig 2: Proposed architecture

Hardware Implementation

VGA Block

The Tetris game will have 3 layers of graphics. The bottom layer will contain the background of the game. The middle level will contain the falling and resting tetrominos. The surface layer will show the counter that keeps track of the score and a “Game Over” overlay.

Each of the tetrominos will correspond to four identical sprites, which are the component blocks that make it up. We will have signals coming from software that we map to each new tetromino block, as well as the rotation of the block. Our hardware side simply receives the indices of blocks it has to render on each tick of the game. This means that enough clock cycles must pass between each in-game tick for the hardware to receive the data (1 or 0) for all of the possible block locations.

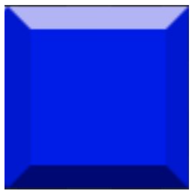

Fig 3: A proposed sprite for one tetromino block. This would obviously need to be lower resolution.

Wolfson WM8731

For the audio functionality, we will be using the Wolfson WM8731 CODEC on the DE1-SoC board, which offers 24-bit audio

DE1-SoC I^{2}C Multiplexer

We will be using the I^{2}C multiplexer to determine if the input to the I^{2}C bus comes from the HPS or the FPGA part. The FPGA part will handle the I^{2}C configuration.

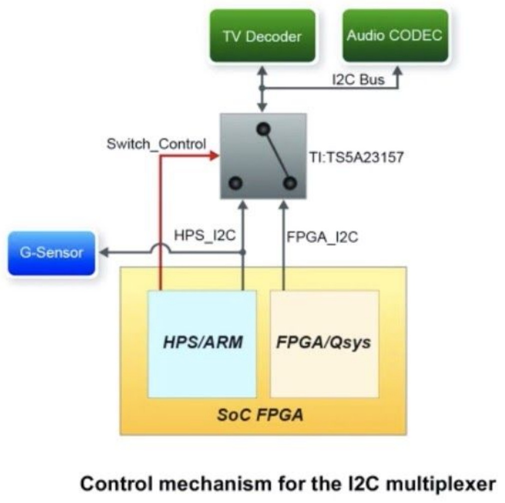

Fig. 4: I^2C Multiplexer

Software Implementation

Game Logic

All game logic is controlled through C code that communicates with the System Verilog program.

Tetromino control: The movement and rotation of the Tetrominos will be controlled by a keyboard. Tetrominoes can be moved left and right using inputs from the left and right directional keys, and rotation of the tetromino is done with the space key.

Tetromino generation: Tetromino generation is handled through a random number generator algorithm that can choose between the 6 shapes. Additionally another random number generator will determine the initial rotation of the tetromino. Finally, a third random number generator will determine what X coordinate the piece will fall from (keeping in mind that the piece generated from before fits within the bound of the screen at that coordinate).

Line Clearing: The software will keep track of every X coordinate pixel at each Y coordinate row. When all pixels in a row are detected to be filled the line will clear, and the software will tell the hardware to shift the display accordingly.

Game Over: The software will detect if the most recent tetromino placed has any y coordinates greater than the bounds of the screen. If so the game will end and a Game Over overlay should appear.

Audio

The game would have sounds in the following times:

1. When the game starts
2. While the game is playing
3. When the player clears the row
4. When the player finishes the game