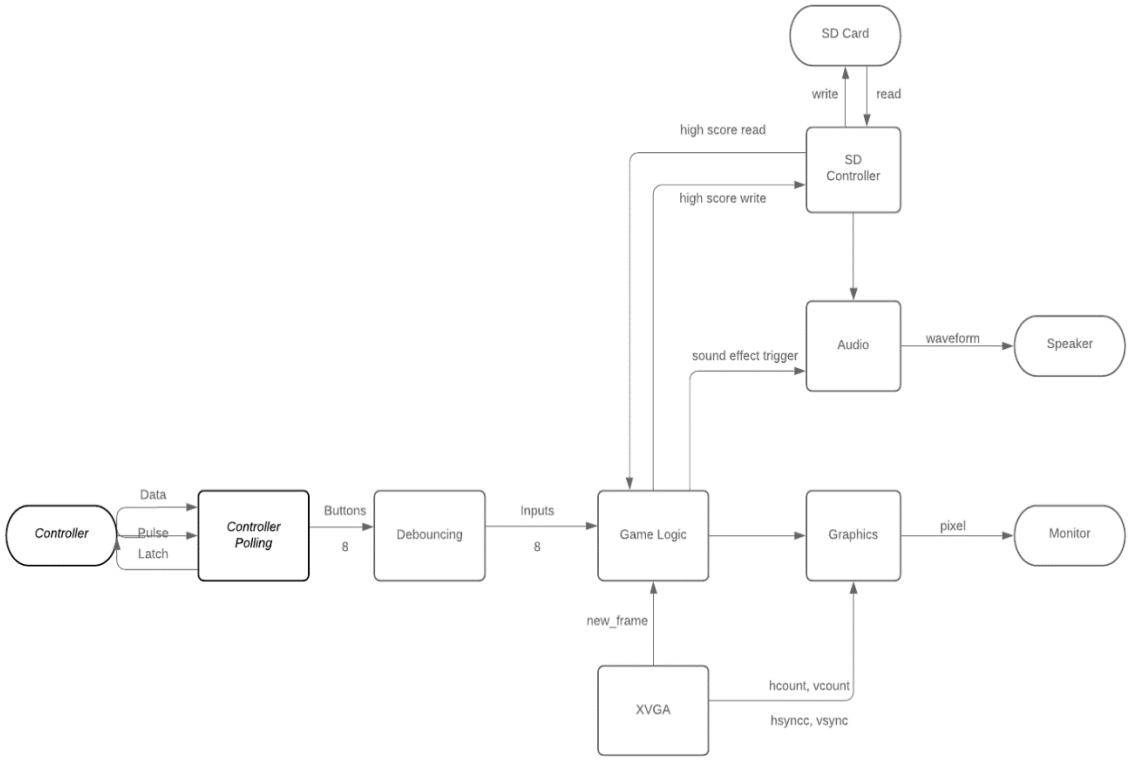
6.111 Project Proposal: FPGA Tetris

Overview

The goal of this project will be to implement the classic puzzle game Tetris. Tetris is a game where the player must neatly stack a sequence of falling shapes onto a 10x20 playing grid. There are 7 different shapes, or Tetrominoes, each of which is comprised of 4 squares. The Tetrominoes can be shifted left or right, and be rotated clockwise or counterclockwise as they're falling. Every filled row of squares grants points and clears itself from the screen. As the game progresses, the Tetrominoes' falling speed increases, increasing the difficulty. The game ends when the uncleared squares reach the top of the playing grid.

Block Diagram



Game Logic

Playing Grid

The playing grid information will be stored as a matrix of 200 FSMs, each of which encodes information about the current state of a single square - whether or not it is filled in, as well as its color (each Tetromino type will have a unique color). The current bottommost square in every column will also be flagged. A line clear will occur when every square in a row is filled, shifting all rows above it down appropriately. Since a falling Tetromino cannot clear lines until it is landed and locked in, it shall be stored within a separate FSM. Therefore, the playing grid state will only update every time a Tetromino lands.

Falling Tetromino

The falling Tetromino will be stored within an FSM which will encode information about its center coordinate, shape, and rotational orientation. Every position refresh, the state will update based on the player's inputs.

Landing Detection

The Tetrominoes cease falling when any of its squares reach any previously placed Tetromino, or the floor of the playing grid. At this point, the playing grid will update to reflect the terminal position of the falling Tetromino. Any filled rows will be cleared, and a new Tetromino will spawn in.

Tetromino Generator

The next Tetromino will be determined via a pseudo random number generator, which can be implemented with a linear-feedback shift register. A maximal length LFSR 31 bits wide will have a cycle length of approximately 69,000,000. Every 4 bits generated will correspond to an encoding for one of the seven shapes, or will be discarded if it is the eighth. In lieu of generating a random seed on power up, it will be sufficient to continuously run the shift register in the background. This way, the first sequence of Tetrominoes is not determined by the static initial seed, but by how long has passed since power up. At any given time, three Tetrominoes will be generated - the one currently on the playing grid, the next Tetromino, which will be displayed on the screen as information available to the player, and a hidden third Tetromino.

Scoring

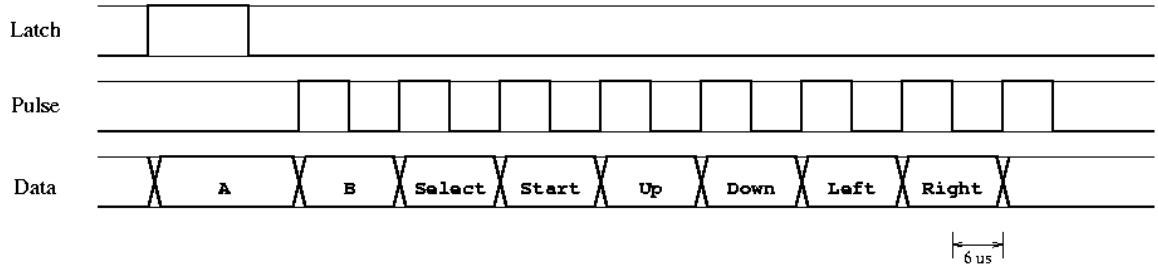
The scoring guidelines will be copied from the NES version of Tetris, and are partially determined by difficulty level. The difficulty level will determine the Tetromino fall speed in units/frame, and will increment every 10 lines cleared. High scores will be stored in an SD card to retain information across play sessions. Upon power on, the high scores will be loaded into SRAM and displayed on the main menu.

Graphics

There will be two possible display states: the menu, and the game screen. The menu will allow difficulty selection, and will display high scores. The game screen will display the current playing grid, the next Tetromino available, the current score, and the current difficulty level. Only two type of images will have to be generated: individual squares, and text. The individual squares can be generated by mapping a color state (Tetromino type) to an appropriate RGB value. Blocks of text will be saved as individual image ROMs. Scores will be parsed together from ROMs of specific digits.

Controller Polling

To input commands, the player will use an original NES controller that has been interfaced with the FPGA. The controller will run at 3.3V provided by the FPGA. In order to receive data from the controller, the FPGA will send a 12us latch signal to the controller. This is followed by 8 6us pulses on a pulse line, which correspond to a different button's data availability on the data line. This will be repeated every 120us to update the input signal as often as possible.



Audio

The primary audio output will be the Tetris theme, stored as COE data on an SD card. This theme will repeat upon reaching the ending address in the SD card's memory. In addition, some simple sound effects will be present. Two different single tone chirps will be used: one for a Tetromino landing, and another for a lone being cleared. Since they are so short, the background music can be stopped whenever either tone is triggered.

External Memory

The FPGA will interface with this card using the SPI SD controller provided under 6.111 Tools. Dealing with managing an SD filesystem can be avoided by writing the preloaded music COE data directly to the SD card using the HxD software. Some additional reads and writes will be performed in a separate memory location, for high score storage.