Samir Lavingia and Brennan Swanton
EE Final Lab Project Report
December 5th, 2013
EE201

Tetris:

Introduction:

For our final project we wanted to implement the famous game known as Tetris. The game interfaces with the VGA monitor and uses on board buttons and such. We build this entire thing from scratch except for the debouncer and VGA example that we build off of. Unfortunately we had some problems and were only able to finish some of the game pieces as we ran out of resources on the FPGA board.

Description:

In the game of Tetris the goal is to clear as many lines as possible. To do this the player must move and rotate random pieces that fall from the sky one at a time to create full horizontal lines on the board, clearing them. Each piece is made out of four Tetriminos. These pieces can be rotated and moved across the screen as needed. When a row is cleared the user gains one point. The blocks must stay within the allotted grid. The user can click and hold left and right to move multiple steps or click and hold down to teleport the piece to the bottom.

How to Play:

Connect the VGA to the board and make sure all the switches are set to zero. After this is loaded you toggle switch two up and down to start the game. Clicking the left button moves the block to the left. Clicking the right button moves the block to the right. Clicking up rotates the brick. Clicking and holding down will teleport the brick down. If you lose toggle the start switch up and down to restart the game. The score is recorded on the LED display

Description of Code:

The major modules of our circuit were VGA_demo.v and tetris.v. The way we worked out this project was to split it into two main bits (pun intended), one for VGA/display and one for the back end. The back end contained a huge array that represented the grid, where a one would represent filled and a zero would represent an empty space. There are five main states in this file: Initial, Generate Block, Move and Rotate,

Collision, and Lose. The initial state just is there for after the game loads and transitions to generate block when the game is started.

The generate block state would pull up a random number and generate a random block from the potential options, in our case a bar and a square. If the incoming block would be blocked by the current grid you would lose, just like in the real game. From here it moves to the rotate and move state.

The rotate and move state allows the player to adjust where the block will land. It is also here where the player is allowed to rotate the brick. After a certain loop condition the state would be changed to the collision state.

In the collision state the current brick is moved down. If this makes it hit another brick its location will be saved and a new brick would be generated. It is also in this state where it checks if a row has been filled. It checks the row and up to four rows to be deleted as this is the max case when a vertical bar is put into place. From here it can either go to generate piece or rotate. If it hits a place it needs to stop (another brick or the bottom) it will go to generate piece. It is does not hit anything it will go to rotate and the cycle will continue.

Conclusion/Future Work:

In the end we were not able to complete the project as we would have liked because we wanted to complete adding all the pieces in. Unfortunately we did not have the time nor the resources to accomplish this task as our FPGA ran out of resources. Throughout the past 4 days (and nights really) we have gained a lot of knowledge about Verilog and the FPGA board. For example all our labs up to now were small so we did not really have to worry about running out of time or issues with timing design/etc. Through this project, we also gained a lot of other skills, including time management, teamwork, and problem solving. We learned how to start with a large project and break it up into smaller tasks.

One of our main challenges was understanding how to use the VGA and being patient. By the end each compilation would take over 45 minutes so testing was very hard to do, which continued to become very frustrating. If we were given more time we would have implemented more of the blocks by basically rewriting all our code to allow us to use more blocks. Along with this more time would have allowed us to wait for the ridiculously long compilation times. We started optimizing everything and were able to reduce the space that it was using so we could have implemented all of the pieces but at this point it was too little

too late. Maybe if we had an additionality then we could implement everything have have it work. The problem was that we had over 100 comparators because we had an if statement for every bit so it made a new comparator for every bit. This was repeated many many times creating hundreds of wasted comparators as they could have been re used.

Overall, we learned a lot while working on this final projects. It was a lot of long hours (25 in total over 3 days), but we are happy with our result and hope you are too. The best part about this project was taking what we learned and making something cool and fun with it that we could see.

Additional Information:
Attached there is the tetris.v file that is what we used to generate the bit file. The file "TEST.v" is the latest file that we wanted to use. It is not stable but it has the majority of the code that we would have wanted to use if we had more time to fix it. The main differences are that this testing file has more of the code that we would have used to implement the additional shapes.