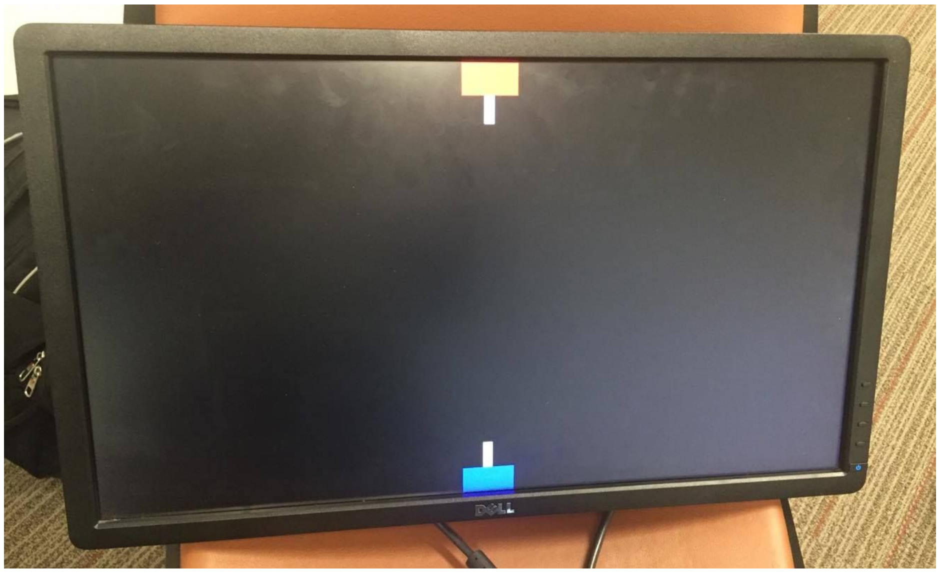
“Tank Game” Project Report

You Li, Shiyi Pang, Yang Liu, Zhonghan Li
November 30, 2017

In this lab we used VHDL to create a tank game on an FPGA board and visible through a monitor. The game is meant for two people and the controls are inputted through a PS2 keyboard.

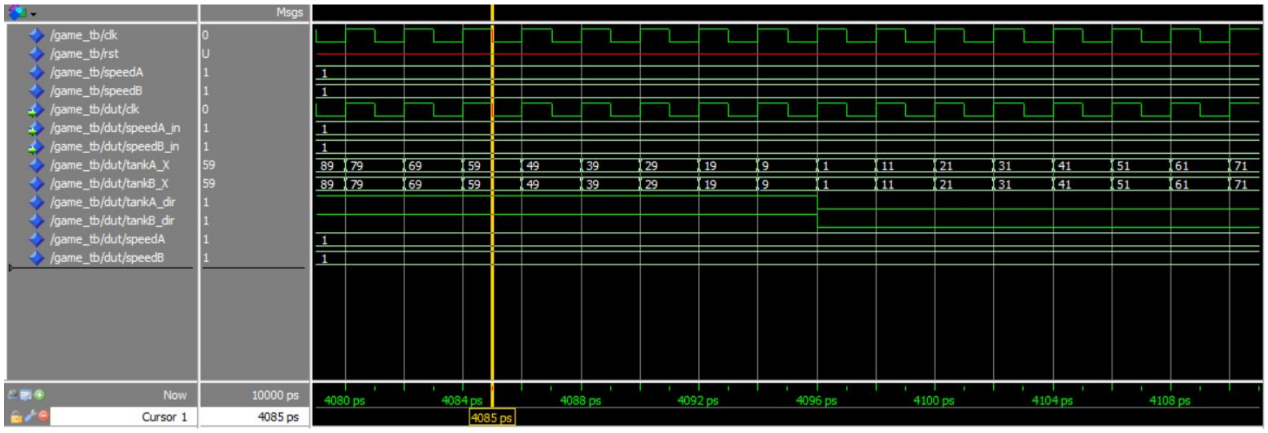

The Tank Game as it appears on the screen.

The gameplay revolves around destroying the other player’s tank by hitting it with bullets. First to land three bullets on the opposing tank wins. The tanks move back and forth on the top and bottom of the screen, and players can only control the speed of their tanks and when they fire. When a player wins, a message appears in the FPGA board’s LCD screen and the game stops.

To create this game, our design had to include entities and concepts such as tanks, bullets, constant package, VGA output, PS2 input, collision detection, score tracker, reset button and game over behavior. Detailed on these components are outlined below.

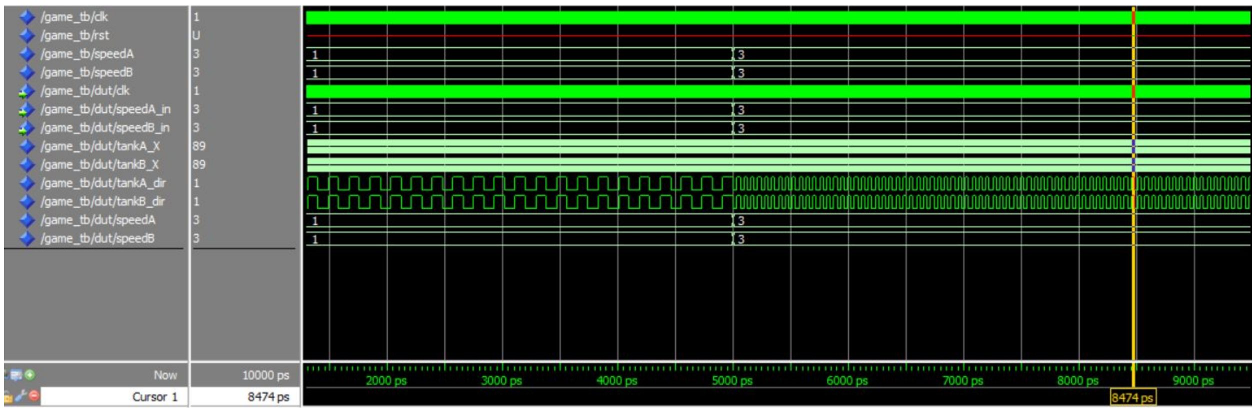
Tanks

Our tanks are drawn using the pixelGenerator component in the project files. This generates a VGA output of tanks taking a single location as an input. The actual behavior of the tank is based on this single location and drawn accordingly. In our top level file, we programmed the tanks to move back and forth horizontally, keeping their vertical positions constant. Upon reaching the edge of the screen, the tank's direction is reversed.



Below is a simulation of only the tank processes:

Simulation of the tank movement. Notice the direction is reversed as the tank's position reaches a boundary.



A zoom out of the same simulation. When the speed goes up, the directional changes occur more frequently, also indicating the tanks have sped up.

Bullets

The bullets are drawn in a similar fashion to the tanks in that pixelGenerator takes a single location set and draws the corresponding boundaries for the bullet. We implemented a simplistic collision detection algorithm that increments a score when a bullet overlaps with an opposing tank. The occurs when a bullet is simultaneously within the height and width range of a tank. If the bullet reaches the edge of a screen without hitting a tank, it is essentially nullified and only then a new bullet may be shot.

Constants

Our constant package(tank_const) includes important specifications about the game such as tank and bullet speeds, screen boundaries and object sizes. The difficulty and other specifications can be easily changed using this file.

VGA Output

The VGA behavior was the first mini-project that we examined. The project includes the vga_sync, pixelGenerator, VGA_top_level, and colorRom files. We first learned to draw squares on the screen by editing pixelGenerator, which were then extended to be our tanks. The VGA_top_level file is how our game interacts with the generator.

PS2

The PS2 mini-project taught us how to use keypresses as inputs. Relevant files include keyboard, leddcd, oneshot, ps2. The ps2 file is our main interface with our game, and it is in there that we bind keypresses to certain controls in the game. We used the leddcd to initially verify that our keypresses had the correct behavior and then used them to keep score.

Score Tracker and Reset

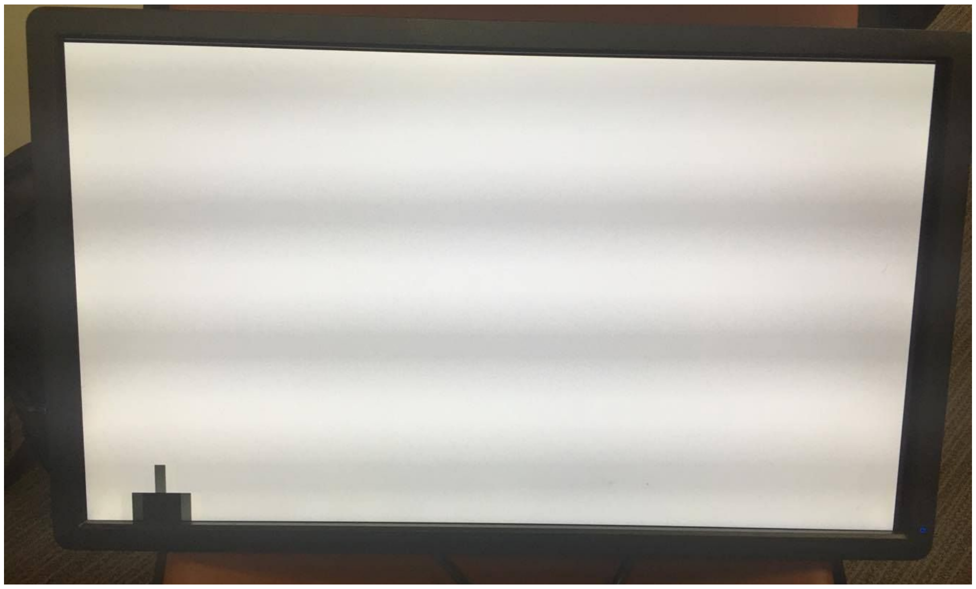
The score is kept with simple integer signals that are displayed using the leddcd component when the game is in progress. When a player is shot three times, the game goes into game over mode. We also implemented a reset button that resets the scores and positions of the tanks for each player and starts the game over.

The LED score tracker. Player 1 is currently in the lead with one point.

Game Over

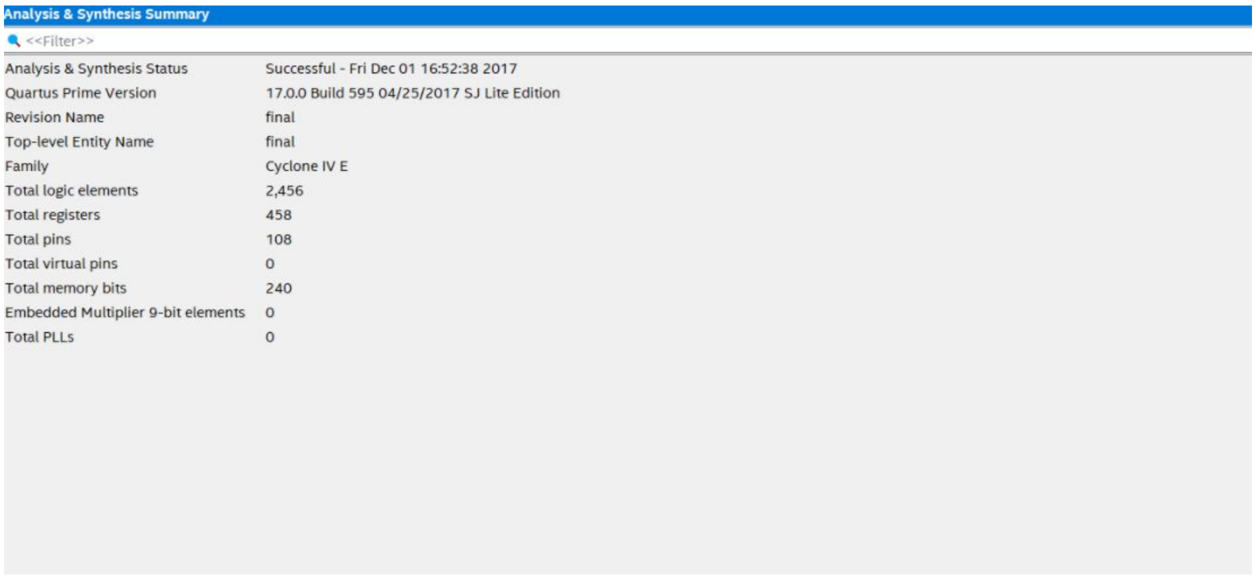
The game ends on three hits for either tank. When this happens, the game stops and the losing tank disappears from the screen. A congratulations message is then displayed on the FPGA board’s lcd screen. To restart, the reset button must be pressed.

LCD Screen upon victory.



Screen changes when one player wins.

Synthesis Results



Analysis & Synthesis Summary

Analysis & Synthesis Status	Successful - Fri Dec 01 16:52:38 2017
Quartus Prime Version	17.0.0 Build 595 04/25/2017 SJ Lite Edition
Revision Name	final
Top-level Entity Name	final
Family	Cyclone IV E
Total logic elements	2,456
Total registers	458
Total pins	108
Total virtual pins	0
Total memory bits	240
Embedded Multiplier 9-bit elements	0
Total PLLs	0

Final Thoughts

The tank game runs fairly smoothly and has all the required behaviors. Our design was based on the notion that we could keep track of all the behaviors more easily if they were programmed on the same top level, but as the project expanded, our main game file got more convoluted. As a result, we experienced decreasing efficiency and things started becoming harder to debug. For future implementations, a segmented, modular approach for the tanks and bullets may be a more feasible design.