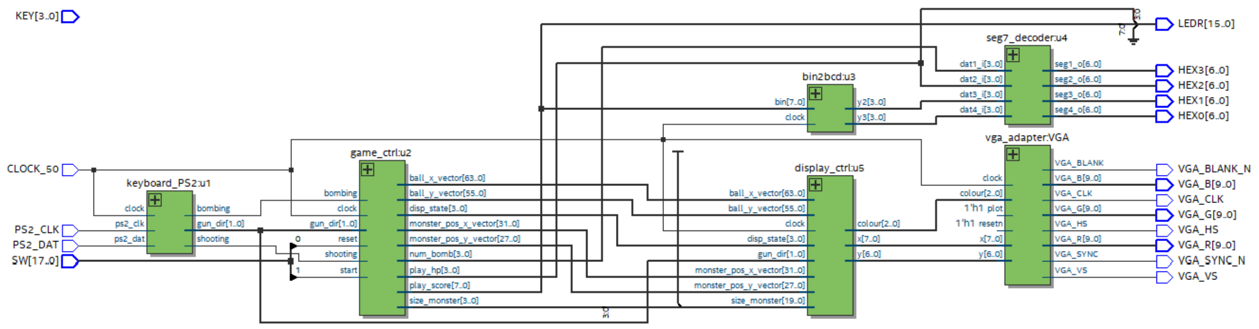
CSC 258 Project Final Report

Project name: Shooter
My name: Xinyi Zhang
Partner's name: Linjun Ye

Description:

It's a simplified shooting game with a shooter and monsters. Each user is equipped by a gun positioned at centre of the screen. The gun can rotate itself to four directions which are north, south, east and west controlling by keyboard W, S, A, D. Monsters come out from the four directions with different speed and time intervals. The goal is to protect us from attacking by monsters. Each turn contains three lives. Whenever monster touches your gun, you lose one live. The bullet will come out from the gun whenever O is pressed. Each time you shoot a monster, you get two scores for the shooting. There will be a winning screen if you achieve 99 scores without losing 3 lives. Have fun playing the game!



High-Level Schematic:

Project Success:

1. We managed to pull off using keyboard as game controls which takes a lot of time to figure out but finally pays off and works like a charm.
2. After a long time of recompiling and testing, we are able to find the best settings of the game like the size of the objects, monster spawn interval, movement speed to keep the game balanced and challenging at the same time.
3. We are able to add an airstrike button that clears the whole screen to help the player survives, which makes the game a whole lot fun and tactical.
4. Apply what we learnt in labs, like seven segment decoder, FSM design, data path of VGA controller and rate divider.
5. Apply hierarchy structure properly and build a systematic code construction for game control so that it is readable for everyone and easy to debug.

Unexpected event:

1. The original plan for the gun is that it cannot only rotate itself, but also move horizontally and vertically by keyboard. However, this needs much more efforts since for each clock cycle, we need to calculate the position of the gun and draw it again. Moreover, we need to view bullets as obstacles since we can't draw on the bullet. So, this will add a lot of if statements to calculate if they are in the same position.
2. It supposed to have different types of monsters which have distinct lives. However, since the way we deal with collisions is to calculate distance of positions of monsters and bullets and we need to trace the position of each monsters and bullets, the condition for the collision will depends on the size of the monsters. The block for collision will be much more redundant if we check the size each time there is a collision. Moreover, there will be two random generators to control monsters. One is for colour and one is for direction.
3. It is hard to trace all monsters and bullets if there is no limit of quantities. Since we need to trace the path of each monster and bullet on the screen in order to calculate the collision position, there is no way to generate unlimited quantity of monsters and bullets. Therefore, we constrain the quantity of monsters which is 4 and bullets which is 8.
4. The same as the second problem, if we calculate the score according to the type of the monster, the code will be unimaginable redundant since there will be many if statements to distinguish the cases.
5. Since we calculate the collision by distance between two items, there is no way to deal with several different collisions happens at the same time within the collision region.
6. The edge detector of PS2 keyboard is more complicated than we think. We didn't notice that the keyboard signal is triggered by negative edge of PS2_clk at first.
7. The background music is so complicated that we give up at the end.

Lessons learned:

1. Keyboard setup needs to consider edge detector and jitter filter so that the board can receive accurate scanning code.
2. VGA is not that complicated to setup. We only need to provide x, y position and colour for each pixel so that it can be drawn correctly. Also, we need to refresh the screen pixel by pixel at each clock cycle.
3. VGA is coded pixel by pixel, there will be a hard time if we expect fancy effect on it such as explosion and collision.
4. For identical items on the screen, we need to decide the maximum number shown on the screen at the same time. Because we need to trace their positions and directions so that we can manipulate them after initialization. Otherwise, there will be huge amount of if blocks and the code will be hard to compile.
5. Random generator is not a real sense of randomness. It's a number generator with a systematic cycle and we pick one part of the sequence to fulfill our randomization.
6. Do not design too many functionalities and fancy effect at the beginning. Otherwise, even the base case will be much more complicated than you think.
7. The high-level schematic will be helpful before the design so that we will have a good understanding and sense to construct the whole structure.

8. The data path for each block should be as simple as possible at first so that it can be implemented for sure. Extra functionalities can be developed later.
9. Deciding the priority for signals is always important. We don’t want to trace the signal which relates to the transition of states among hundreds of codes.