CSEE 4840 – Embedded Systems

Final Project

Rally-X Design

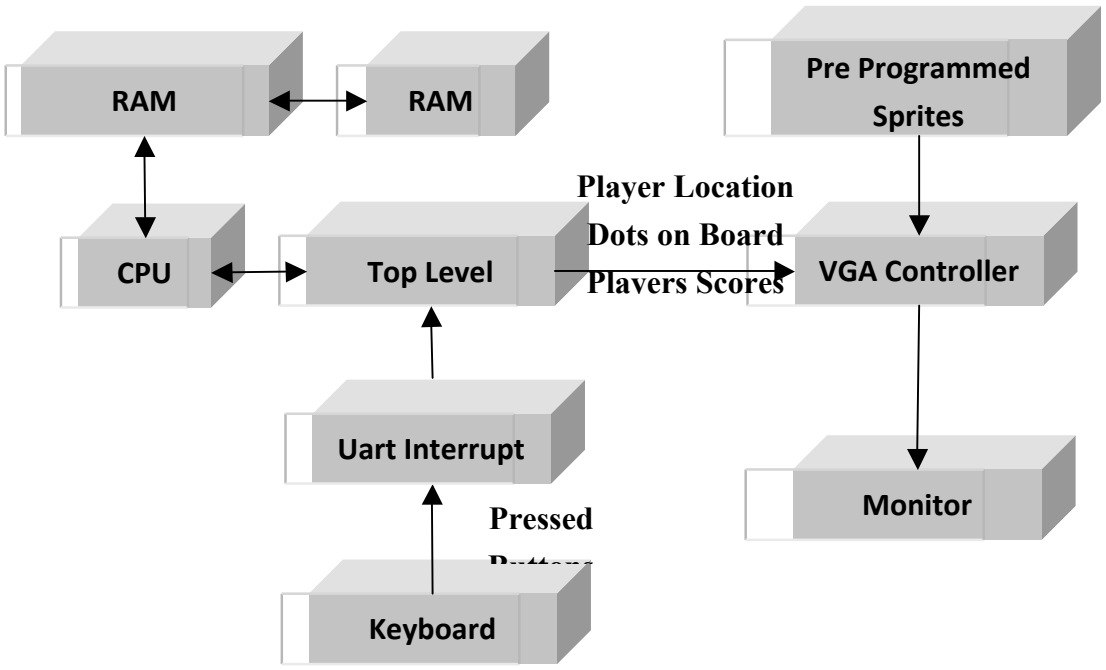
Group Members:
David Jew (drj2109)
Shangche Peng (sp2922)
Edward Lai (el2564)
Yun Seong Nam (yn2213)

-Introduction
-Block diagram
-Game logic
-Game Graphic
-Software.
-Memory Mapped I/O Layout

Introduction:

Our goal is to design a Rally-X style arcade game using a combination of hardware (VHDL) and software (C). In VHDL, we will design a VGA controller, RAM and ram controller, PS2 Keyboard controller, and CPU. We will display the game using the VGA output on the Altera board. The sprite graphics will be initialized in hardware. In C, we will write software that controls the game state/logic (player, enemy cars, flags, fuel (timer), score, map positions etc) depending on inputs from a PS/2 keyboard.

Block Diagram:



Game Logic:

Number of Players: 1

Controls: 4 arrow keys (for steering), 1 button for "Smoke Screen", 1 Start Button

There are 10 flags placed through out the game map/maze. The goal of the game is to collect all 10 flags by navigating your car throughout the maze. You must do this while avoid enemy

cars and obstacles (rocks/blocks) and also collect the flags.

Each flag you collect is worth a certain amount of points. Flag values increase each time you collect a new flag. There also special flags that act as multipliers that will double (or multiply by whatever factor we set) your score for the rest of the round.

Enemy cars will patrol the maze for the player's car. These cars will spawn either at the bottom or top of the screen. Enemy cars can be stunned for a small amount of time if they run into "smoke screen". Smoke screen also causes an enemy car to chase the player using a different route.

The fuel gage acts as a timer. When fuel runs out, your car will move very slowly. Also, you will not be able use "smoke screen" anymore. Fuel decreases with time (due to driving) and when you use "smoke screen". The player can use "smoke screen" when there is enough fuel.

Upon death, the player will lose a life. Any score multipliers will be reset and the base flag value will be reset to the lowest value.

Because the game map is much larger than the resolution of the VGA display, we will need to implement vertical and horizontal scrolling. The player's car will be centered on the screen.

Graphics

There will be VHDL code that defines and draws the game map. The VHDL code will also draw cars(player and enemy cars will be different colors), flags, and "smoke screen", fuel gauge, score, and lives. We will store all of the graphic sprite/ghosts in the RAM.

The graphics will be made using 2 dimensional arrays. The values in each spot will tell if the pixel is on/off and the color of the pixel if it is on.

Software will tell the VGA controller what part of the map to display. Software will also tell the VGA controller where to display cars, flags, and smoke screen. Software also controls the fuel gauge, score, and lives.

Software

The software will be calculating:
-Player (car) position in the map. (Player will always be in the center of the map)→ Show which part of the map to be display.

-Player (car) state (direction, alive/death, etc)

-Enemy car (computer) positions and states (direction, alive/death, etc)

-How the enemy car trying to catch the player, give out where the enemy car heading to for the next position.

-Keep the count of the score/lives for the player

-Count how much flag left in the current stage

-Send all the display info to the hardware (how much fuel, score, lives, flags to the VGA)

Memory Mapping

Memory between software and the graphics component is mapped as follows:

Address | 32-bit words

1 Player's y-position (16), Player's x-position (16)

2 Player status (32)

3 enemy1's y-position (16), enemy1's x-position (16)

4 enemy2's y-position (16), enemy2's x-position (16)

5 enemy3's y-position (16), enemy3's x-position (16)

6 player's score (32)

7 player's fuel(32)

8 player's lives (32)

9 flag1 data (32)

……

18 flag10 data (32)