Embedded System Project Proposal

Aircraft Combat (MINI Game)

Yun Miao (soc12ymi) & Tan Siyu (soc12sta)

2013/9/10

1 Introduction

The idea of this project is to design an Aircraft combat game. The player can control an aircraft by using keyboard. The aircraft is flying in a two dimension vertical scrolling space. There will be enemies coming from top side of the space. The player must avoid them or destroy them by using the auto firing cannon on the aircraft. These things will be displayed on a VGA monitor controlled by FPGA board. The design will consist of a general purpose CPU on the FPGA board, which is Microblaze, and a graphic accelerator. The CPU decides the content to be displayed while the graphic accelerator draws the content on the screen. A demonstration of our idea can be seen in Figure 1.

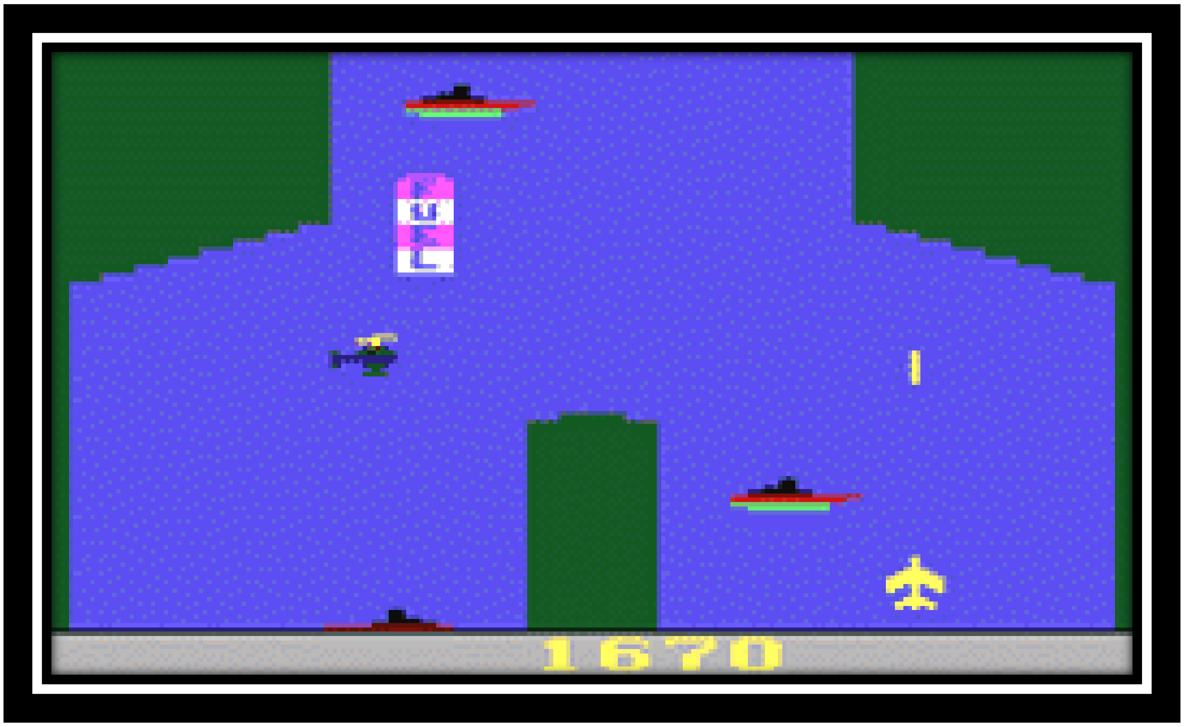

Figure 1 demonstration of Aircraft combat

2 Detailed description

A block diagram of the system is shown in Figure 2.

2.1 Graphics and VGA output

This game will have a resolution of 640*480 @60 Hz. For reducing the usage of memories, the whole frame is divided into a grid of 20*15 tiles making each tile 32*32 pixels in size. There will be different tiles for background and foreground. These tiles will be stored in the two BRAMs, which is memory block 2 & 3 in Figure 2. The screen scrolls 1 pixel per frame. So that a scroll value is introduced, and with screen position information, an address for BRAM storing bit map can be calculated to decide what to be drew on the screen.

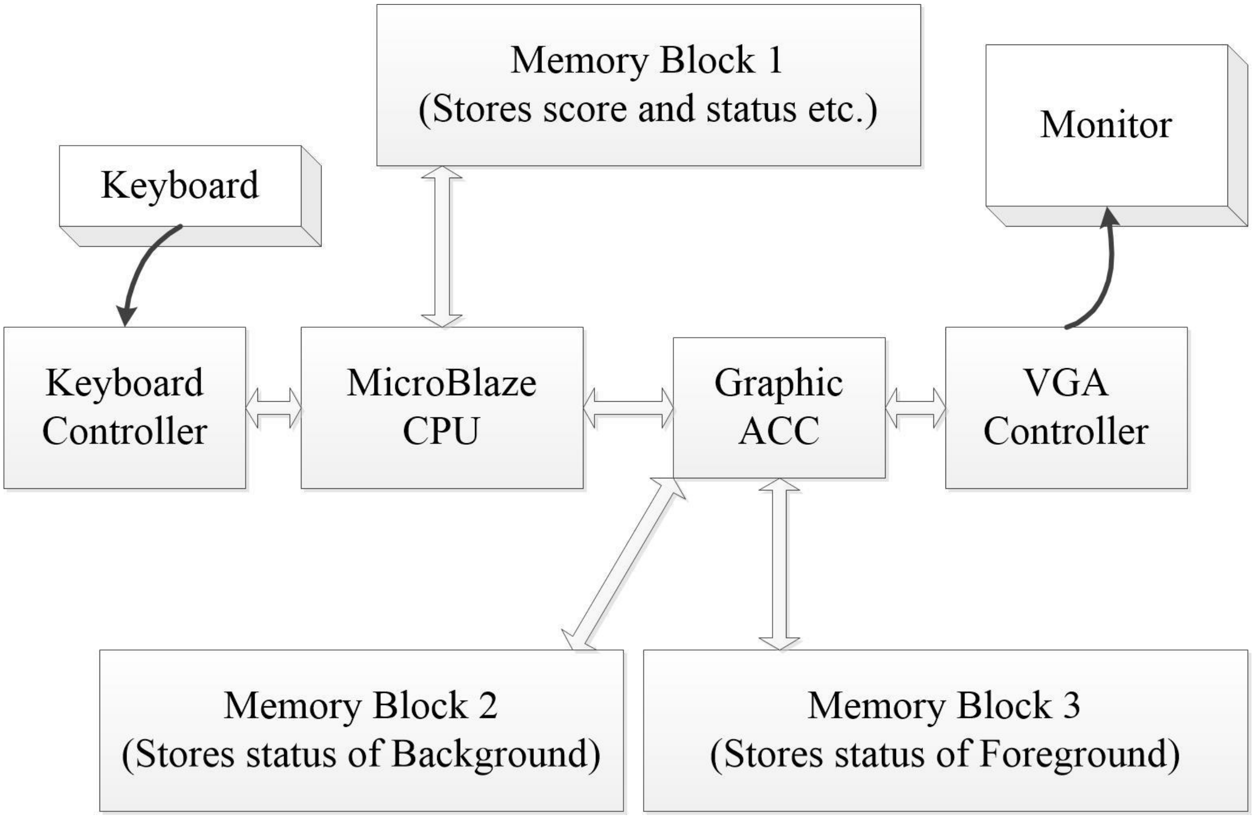

Figure 2 Block Diagram of the System

2.2 Input

The control from the player to the system is implemented by using USB keyboard, which is connected to the USB port on the FPGA board. The direction buttons will be used to control the movement of the aircraft. A register will be used for the keyboard controller to store the latest key state of the keyboard for one clock cycle. At every refresh of the screen, CPU receives an interrupt to read the register. The control signal generated by the keyboard is decoded by the keyboard controller with the help of VHDL. If there is extra time, other control methods can be implemented.

2.3 On chip communication

For communications between different hardware cores, PLB bus is used.

2.4 Memory

There will be three memory blocks in the system. There may be several BRAMs in one block for different storage.

- In memory block 1, stores the instructions for CPU and status of gameplay such as scores, lives.
- In memory block 2 which stores status of background, there may be one BRAM for storing the bitmap for tiles, one BRAM for tile map to store the arrangement of tiles, one BRAM for storing the next column of background to be shown on the screen.
- In memory block 3 which stores status of foreground, it's similar as in memory block 2: one BRAM for bitmap of object tiles to store different graphic elements, one BRAM for tile map, one BRAM for storing the next column of foreground to be shown on the screen.

2.5 Software

The CPU is interrupted at each refresh of screen. During the interrupt, several things are done:

- Read input signal from the register of the keyboard controller.
- Calculate position of objects in the game.
- Calculate and judge collisions of objects.
- Calculate background and foreground scroll value and position.
- Sending graphic information to graphic accelerator.
- Wait for next interrupt.

3 Time plan

Table 1 Time plan for the project

	Tan Siyu	Miao Yun
Week 1	Project planning	Project planning
Week 2	Architecture design	Architecture design
Week 3	VGA controller	Game logic design
Week 4	Graphic accelerator	Software design
Week 5	Other hardware	Software design & test
Week 6	Final implementations	Final implementations
Week 7	Report	Report