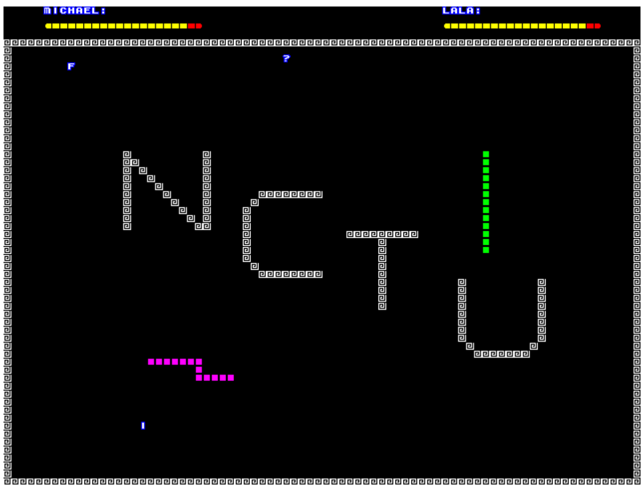
BATTLE SNAKE

Ming-Ju Wu (mjwu@csie.nctu.edu.tw)
Wei-Chen Sun (mow88088@yahoo.com.tw)

Proposal

Our project is a simple 2-player game. Each player controls a snake trying to choke the other to death. There will be many magical pills to make snakes becoming longer and faster. The game will have many different maps for the two players to battle in.



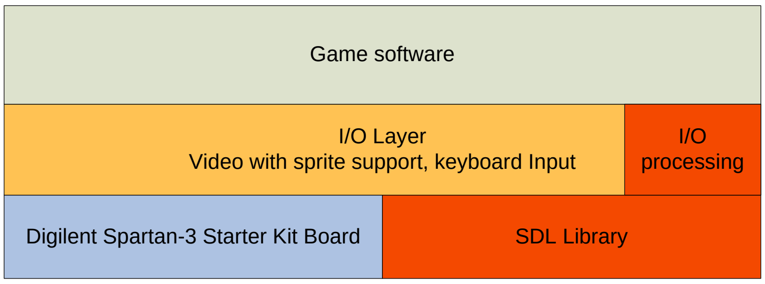
Project Structure

The project is composed of three primarily part, software, graphics hardware, and input peripheral. The software is written in C. Graphics hardware is a video controller capable of displaying color tiles. Input peripheral is a ps/2 keyboard.

Software

Except for smaller memory and slower processor speed (and less intelligent compiler), writing C program on the Xilinx MicroBlaze platform is not much different then on PC.

For our convenience of developing the software, we wrote a wrapper I/O layer which uses the SDL (Simple Direct Layer) Library to emulate the graphics controller and ps/2 keyboard input. With the help of C preprocessor and some #ifdef, #else, and #endif statements, we can maintain a single source code that runs both on the PC and the embedded hardware. The most important difference between SDL emulated hardware and real hardware is that real hardware runs concurrently to the software. When using real hardware, the software only needs to update values in share memory and then graphics controller will take care of the rest. While emulated hardware is actually software, the game software needs to call graphics update each time when something on the screen needs to be changed. In our experience, even a Pentium-4 2.8G PC runs slower then our embedded version of the game.


Game software
I/O Layer
Video with sprite support, keyboard Input
I/O processing
Digilent Spartan-3 Starter Kit Board
SDL Library

Our game state consists of two snake structures and some pill structures. The primary part of the snake structure is a queue that stores x-y information of each piece of the snake body. In each game loop, according to the speed that each snake has, the snake is either moved forward or stays until next game loop. If the snake head hits magical pills, attribute of the snake is modified accordingly. If the snake head hits the wall or the other snake, its health decreases. When one of the snake's healths becomes zero, the other player wins.

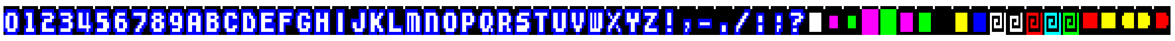
Key	Function
Up, down, left, right arrow	Player 1 direction
F5	Jump to previous level
F6	Jump to next level
Num Pad -	Player 1 length--
Num Pad +	Player 1 length++
Num Pad 9	Player 1 invincible
W, S, A, D	Player 2 direction
1	Player 2 invincible
2	Player 2 length--
3	Player 2 length++

Hardware/Graphics

Most of our graphic system was borrowed from the ps/2 keyboard project with minor changes. First, we reduced memory usage by using fewer BRAM blocks and did memory remapping. Second, we added color display ability by using three shift registers each being responsible for one color bit output. Finally, we had built a graphic conversion program so that we can design 8x8 tiles on a .bmp file then change the file into font RAM data structures.

The controller read each row of the 80x60 tile array then use counter to determine which row of the font should be read. The corresponding row will be read into shift registers then output to VGA connector.

The video controller is capable of storing 64 different 8x8 tiles. The tiles that we have in the controller are as follow. We've wrote a small utility that can encode BMP files into format that could be understand by VHDL and our software video controllers.



For example, the first tile is encoded into the following VHDL attribute:

```
attribute INIT_00 of RAMB16_S36_S36_1 : label is
 "00 00 00 7e            -- row 1
  00 3c 3c ff            -- row 2
  00 66 66 ff            -- row 3
  00 66 66 ff            -- row 4
  00 66 66 ff            -- row 5
  00 66 66 ff            -- row 6
  00 3c 3c ff            -- row 7
  00 00 00 7e";          -- row 8
```

A word of the font RAM consists of a null byte, red byte, green byte and blue byte.

Workload distribution

Software: Ming-Ju Wu

Game Design: Ming-Ju Wu, Wei-Cheng Sun

Hardware: Wei-Cheng Sun
Integration / Testing: Ming-Ju Wu, Wei-Cheng Sun
Report (Software design): Ming-Ju Wu
Report (Hardware/Graphics): Wei-Cheng Sun

Lessons learned

Wei-Chen Sun

Hardware design should be done as early as possible. Later design mostly depends on hardware. In the beginning of the class, Designers should discuss the proposal and software/hardware partition together. More importantly, software designer should specify what effect is needed and to let hardware designer do his job leisurely without worrying a changing specification or a changing interface method. Having hardware designed early, everything in the project will be fine.

Ming-Ju Wu

It is handy to have a software emulated I/O layer so that software development can progress parallel to hardware development. Because of the emulation I/O layer, our software design can start even before the hardware specification had been discussed. Hardware design also benefits from the experience gained wile writing software. We had a much more clear understanding of what the hardware should be capable of doing because we know what the software needs.

Fine-tuning the project take much time. The gap between a working game and a very good game is huge. On the software part, we spent more then half of the time improving game graphics and balancing game play, but still not quite satisfied with the result. After the preliminary edition of the game had been done, we have two choices. One is to add network support through RS-232 communication, or we can polish the game. We take the latter option, and we thinks it is the right choice. Rather then have a big project, have a nicer and smaller project.