Snake+

CSEE 4840 Project Design Document

March 26, 2013

Nina Berg, Joseph Corbisiero, Ilan Elkoby, Molly Karcher, Brian Wagner
Department of Computer Engineering, Columbia University

ABSTRACT

This document contains the design specifications for an implementation of a video game called Snake+. The game is a modern adaptation of the Snake cell phone game. In order to implement it, an Altera FPGA board will be used, as well as a PS/2 keyboard and a VGA monitor. VHDL will be used to program the hardware, and C will be used to write the game logic. The design process is broken down into three milestones, and the expected progress for each is listed at the end of the document.

INTRODUCTION

We aim to recreate the classic cell phone game Snake, but with a few twists to bring it into the modern age. The goal of Snake is to eat as many food blocks as possible to grow your snake (and your score) while carefully navigating the field to avoid running into yourself. Aside from updates to the graphics, audio effects, and general aesthetics, we will also introduce power-ups, obstacles, and a two-player mode. This will be accomplished using the Altera FPGA, a PS/2 keyboard, a VGA monitor, and VHDL and C programming.

GAMEPLAY

The game is played with two players. One uses the A, S, D, and W keys to navigate; the other uses the arrow keys. Snakes automatically move forward, and the keyboard is used to change the direction of the snake. As play continues, the speed at which both snakes move forward increases up to a critical rate. Player 1 controls a green snake, and Player 2 controls a red snake. Players can grow their snakes by eating food and power-ups. These power-ups include slow-down, score-boost, shrink, wall-building, and other similar advantages. The first player to collide with his or her own snake, the other player's snake, or one of the obstacles or walls loses the game.

DESIGN ARCHITECTURE

The following figure displays the basic block diagram of our system. The main components are listed here, and they are each described in detail below.

- NIOS II CPU
- SRAM Controller
- SRAM
- PS/2 Keyboard Controller

- PS/2 Keyboard
- VGA Controller
- VGA Display
- Audio Controller
- Audio Driver

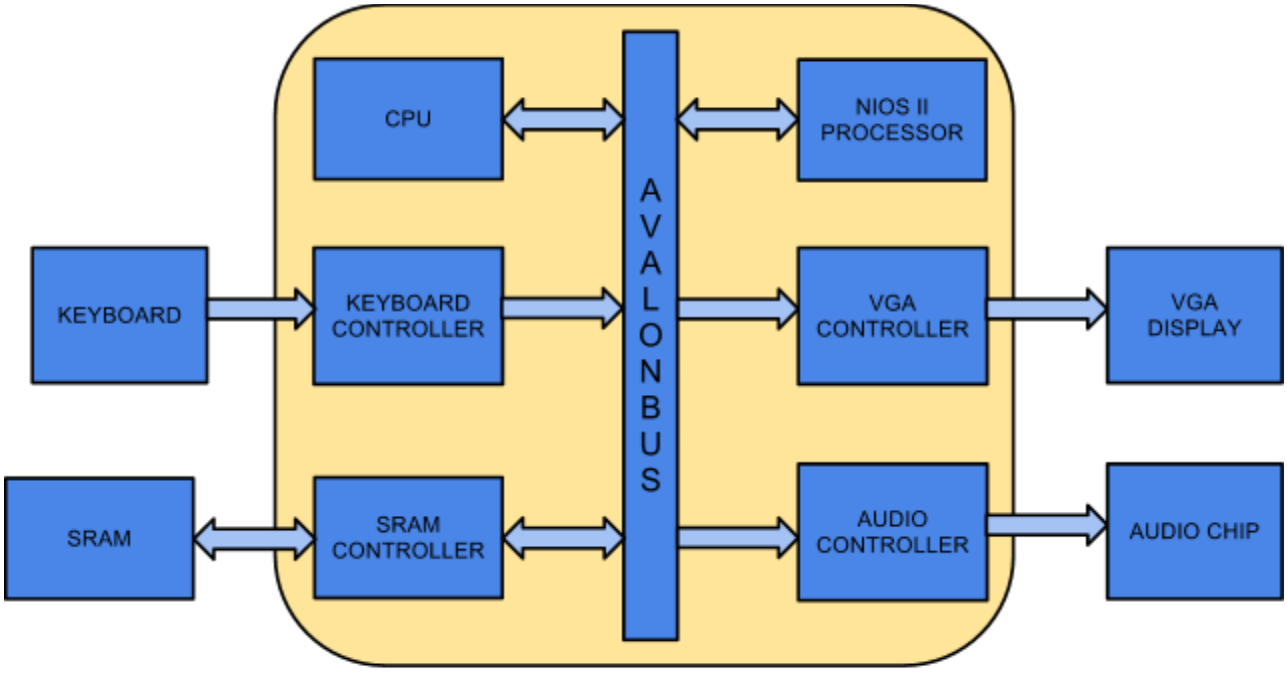


Figure 1: Block Diagram

HARDWARE DESIGN

The hardware components of the design include the Altera board, a PS/2 keyboard to receive user input, and a VGA monitor to display the game screen. The Altera board will provide the SRAM that will store sprites and audio. Below are detailed specifications for each hardware component.

PS/2 Keyboard

The keyboard use in our case is fairly simplistic, as only ten keys will be used in total. Each player has four keys to determine the movement of his or her snake, the enter key will be used along with the arrow keys to navigate the main menu, and the space bar will be used to pause and unpause the game.

During gameplay, the software will check the keyboard for input each time the hardware is preparing to re-draw the snakes. Directional input in either the direction the snake is already

traveling or the opposite direction will be ignored, but input to the left or right (oriented from the direction of the snake's travel) will cause the hardware to re-draw the snake in a turning position in the corresponding direction. If at any point the software detects a press of the space bar, the game will be paused until the space bar is paused again.

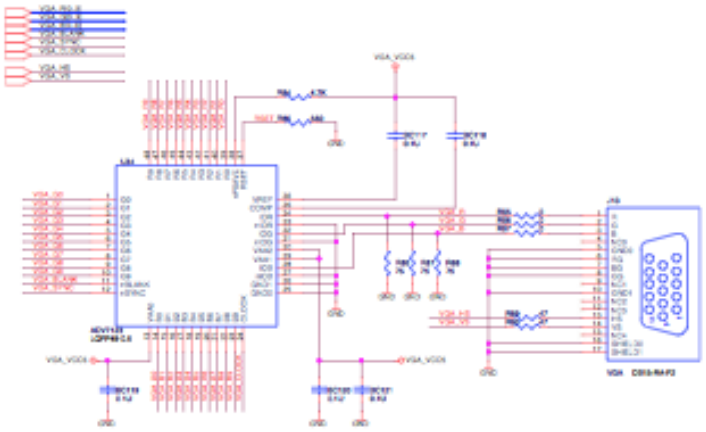
Audio

Sound effects will play a large role in Snake+. When an event is triggered that requires a sound effect, the software will instruct the hardware to fetch and play the corresponding audio file. We have six audio effects, so we will use a 6-bit controller. The effects and controller values are listed below.

Audio Effect	Controller Value
Eat food	100000
Growth power-up	010000
Freeze power-up	001000
Speed power-up	000100
Hit other snake	000010
Main menu	000001

VGA

The VGA controller is an Avalon peripheral that displays a frame buffer located on SRAM to the VGA display screen. The VGA controller does most of the work in accessing the SRAM every time it needs to draw a new pixel in order to get the desired color of the pixel at the specified location.



GRAPHICS

Snake+ is a video game, so graphics will play a crucial role. The basic sprites we will use, along with designs for some simple stages, are detailed below. The colors displayed on the VGA will be encoded as 3-bit vectors, so we have the ability to store up to 8 different colors. Color encodings will be as follows in our design.

- 000 - Black
- 001 - White
- 010 - Red
- 011 - Green
- 100 - Blue
- 101 - Yellow
- 110 - Brown
- 111 - Pink

Segments of our snake, as well as power-ups, food and all other visuals that will appear on the screen will be created within a standard 8x8 pixel size block. This means that we require an 8x8 size matrix in order to store each standard block.

Sprites

The sprites that will be used in Snake+ fall into two categories: snake components and edibles. They are listed below:

Snake Components	Edibles
Head	Food
Body	Growth power-up

Tail	Freeze power-up
	Speed power-up

Designs
The maps in Snake+ occupy a 640x480 frame buffer. There are several map variants: plain (easy), walled (medium), and obstacle (hard) courses. We will write software to draw the maps once the hardware support for the VGA is integrated.

SOFTWARE DESIGN

Game Logic
After constructing the hardware controllers and drivers, we need to create a method of managing the game's content. All of the programming will be done in C and compiled and debugged with the NIOS II IDE.

Two snakes will fight be pitted against each other in a fight for survival. These snakes can grab power-ups that will affect game-play. For example, snake movement may be increased or decreased, allowing one a player a speed advantage over the other. As the game continues, the snakes will grow and the amount of free room on the screen will run out. As this happens, we need to be able to track a collision. A collision may happen between two snakes; a snake may collide with itself, or it may collide with the boundaries of the board. If this event occurs, the game automatically ends and the winner is declared.

Some of things that may appear tricky to implement are playing the correct sound for a specific event, processing multiple key presses (two players on the same keyboard), and checking for a collision between the snakes on any of its segments.

Keyboard Module: Depending on the input from the keyboard, the snake can go up, down, left or right.

Power-Up Effect Module: At any point, a snake may slither over a power-up. This effect can change the length of snake, the speed of its movement, or can cause some other visual effect to take place. This module will track which snake slithered over what power-up using flags to indicate which power-up should be applied and to which snake. The location and time when a power-up appears can be either random or pre-determined, but the power-up should only be displayed in a region of the screen not currently occupied by the snake or another power-up.

Audio Module: Whenever a power-up is acquired, the games starts/end, and the game is being played, background music will be played to keep you entertained for hours. This module should be able to communicate with the power-up module, display module, and win/lose to know when to play a sound.

Display Module: Based upon which direction the snake is moving, the screen will display the correct image for the head based on that direction (North, South, East, West). Also, as the snake continuously eats food, it will begin to grow in size and therefore, more “links” for the snake’s tails should be displayed. This module will most likely be used to display our start screen as well.

Win/Lose Module: Once a snake crashes into another play or wall, a “game over” flag will indicate to declare the winner and end the game.

MEMORY REQUIREMENTS

VGA Frame Buffer

The frame buffer we’ll be using is a 640 x 480-pixel VGA display. This means that we’ll be using a total of 307200 pixels. We encode colors using a 3-bit scheme, so each pixel requires 3 bits. Thus, the number of bits is 307200 * 3 = 921600 and the number of bytes is 115200, or 116kB. This will be stored in SRAM.

Sprites

Each of our sprite objects is 8 x 8 pixels, so there will be 64 pixels in each sprite image. Each pixel will be 8 bytes in size, so each sprite will take up 512 bytes of RAM. We will also store this in SRAM.

Audio

As shown above, Snake+ will use six sound effects. The five in-game effects will be approximately .5 seconds in duration. The main menu sound effect will be approximately 2 seconds. This makes a total of 4.5 seconds of audio. The audio will be relatively low-quality, so we plan on sampling at a rate of around 4000 samples/second. We’ll use a bit depth of 8, which makes for a memory requirement of 4.5 * 4000 * 8 = 144000 bytes, or 144kB. The audio will also be stored in SRAM.

MILESTONES

Milestone 1 - April 2

- Get the PS/2 controller to communicate with the board properly and transmit the user inputs correctly
- Create a connection between the software and the hardware such that the hardware can update and store all information necessary to display the game screen.
- Assure that all working parts of hardware are connected properly to one another and

communicate accurately with a very basic software component.

Milestone 2 - April 16

- Finalize a high-level algorithm for all game logic to be later implemented in software.
- Display all game screens and sprites in hardware.
- Have a snake be able to move around the screen controlled by at least one user via a keyboard input.

Milestone 3 - April 30

- Two snakes controlled by two users moving around the screen
- Detect and respond to when a snake eats, gets a power-up, or hits an obstacle
- Enable audio

REFERENCES

http://en.wikipedia.org/wiki/Audio_bit_depth