6.111 Final Project Proposal

Yi Wang and Stephen Pueblo

November 3, 2006

Overview

For our project we plan to design the classic arcade game Donkey Kong. Our project will generally follow the game's original schema. In the game, Mario tries to climb up to the top of the platforms to save the Princess, while Donkey Kong tries to throw barrels to stop him. If a barrel hits Mario, Mario loses a life and starts from his initial position at the bottom platform. Once Mario loses all three of his lives, the game ends and the player loses. If Mario reaches the top of all the platforms and saves the princess, the player wins and the game ends. The game we plan to implement, however, has a twist to it. Instead of controlling Mario with a joystick and a button, we plan to create a system where Mario will be controlled by the player's motions.

Description

The inputs to our video game system are the reset button and the data from the motion sensor. The data from the motion sensor will be created by a gyroscope and/or an accelerometer. The output from our overall system will be the video signal to the LCD display. The system will contain the following basic modules: the XVGA, the Controller Interface, and the Game Logic modules. Figure 1 below shows the Basic Block Diagram of our system.

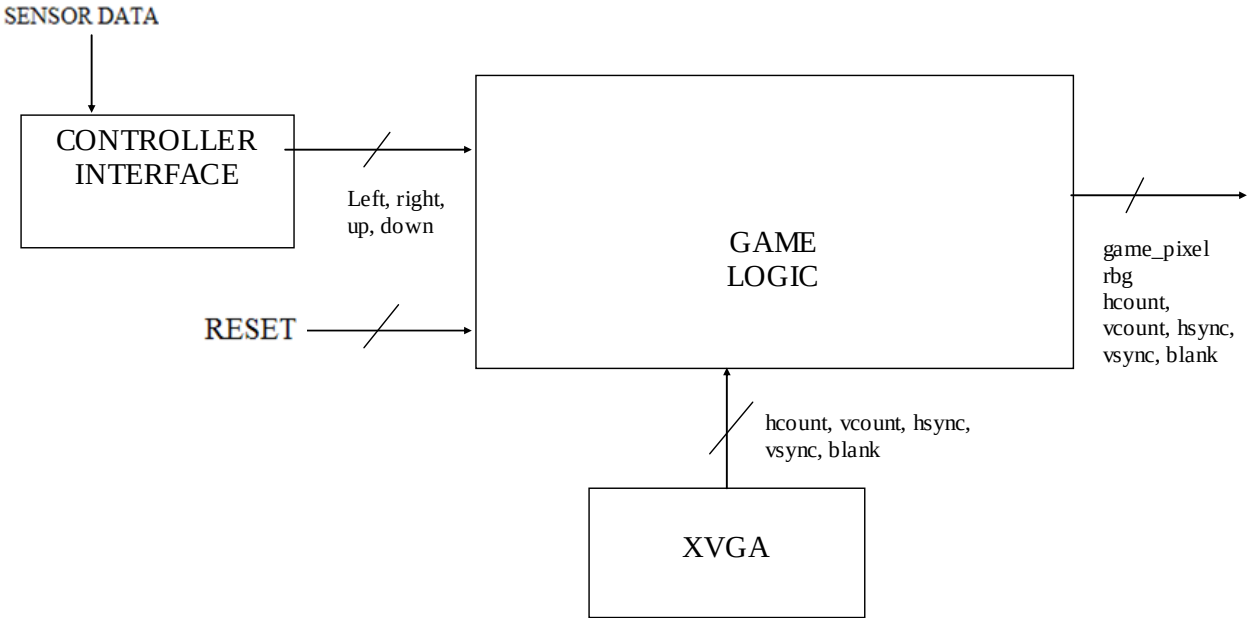

Figure 1: Basic Block Diagram

The XVGA module is the same module that was used in Lab 5 in the implementation of the Pong Game. Since the screen size of the monitor is 1024x768, the clock is 65 MHz. The XVGA module will take the 65 MHz clock and create the signals *hcount, vcount, hsync, vsync,* and *blank.*

The Controller Interface Module will take in the data from the sensor and encode it into the signals *left, right, up,* and *down,* which will be used for the Game Logic module. The analog signal from the sensor will need to be converted into a digital signal. There also needs to be a creation of a protocol between the Analog to Digital Converter (ADC) and the Sensor. Once the ADC samples and converts the signal from the sensor, the ADC will output that data to the Synchronizer/Data Converter Module.

The Synchronizer/Data Converter module will synchronize the data to the system clock. It will also convert the signals from the Analog to Digital Converter to signals the Game Logic Module can understand, which are the left, right, up, and down signals. Figure 2 details the schematics of the Controller Interface Module.

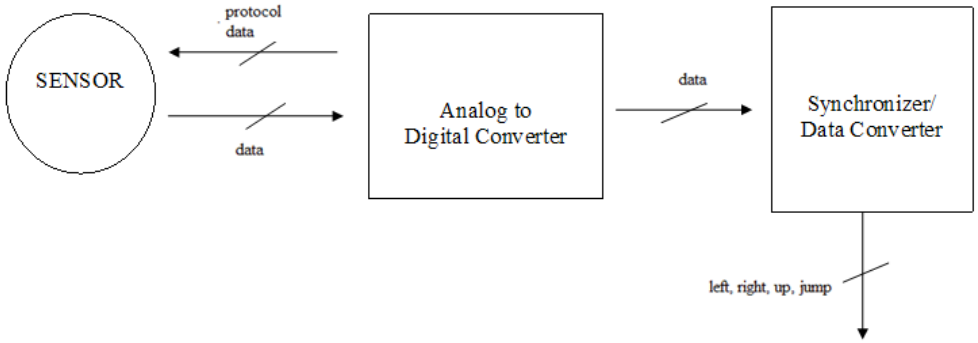

Figure 2: Controller Interface Block Diagram

Game Logic Module

The game logic is the core of the system. It consists of the following modules: Game Finite State Machine, Donkey Kong Logic/Sprite, Collision Detector, Barrel Logic/Sprite, Mario Logic/Sprite, Game Map, Score Counter, Life Counter, and Video Combinational Logic.

The game FSM determines which state the system is currently in. It takes as input *dk_xy, barrel_xy, left, right, up, down,* and *collision,* as well as the global *reset* signal. These signals respectively specify the location of Donkey Kong; the location of the barrels; the direction the user wants Mario to go; a collision signal that contains information about whether there has been a collision between a barrel and Mario; and a reset signal that resets the FSM back to its original state. It outputs *dk_action* to Donkey Kong Logic; *incr_score* to Score Counter; *incr_life* to Life Counter; *b_down, b_left, b_right* to Barrel Logic; and *m_up, m_down, m_left, and m_right* to Mario Logic. These five sets of signals tell Donkey Kong, the Barrel, and Mario how to move depending on the current state of the system, and they tell the Score and Life Counters to increment/decrement the score/life when necessary.

Donkey Kong, Barrel, and Mario logic all take inputs from the FSM, which directs their respective movements. As output, they tell the FSM (or the Collision Detector) their current position. They also output their pixels to the Video Combinational Logic.

The Game Map contains the information of all the stationary objects on screen. The information includes the map Mario and the barrels will traverse, as well as the stationary Princess who is near the top of the screen.

The Video Combinational Logic takes the pixel outputs from the FSM, Game Map, Donkey Kong, Barrel and Mario logic, and outputs to the video display *pixel, rbg, hcount, vcount, hsync, vsync and blank*. Figure 3 below details the block diagram for the Game Logic.

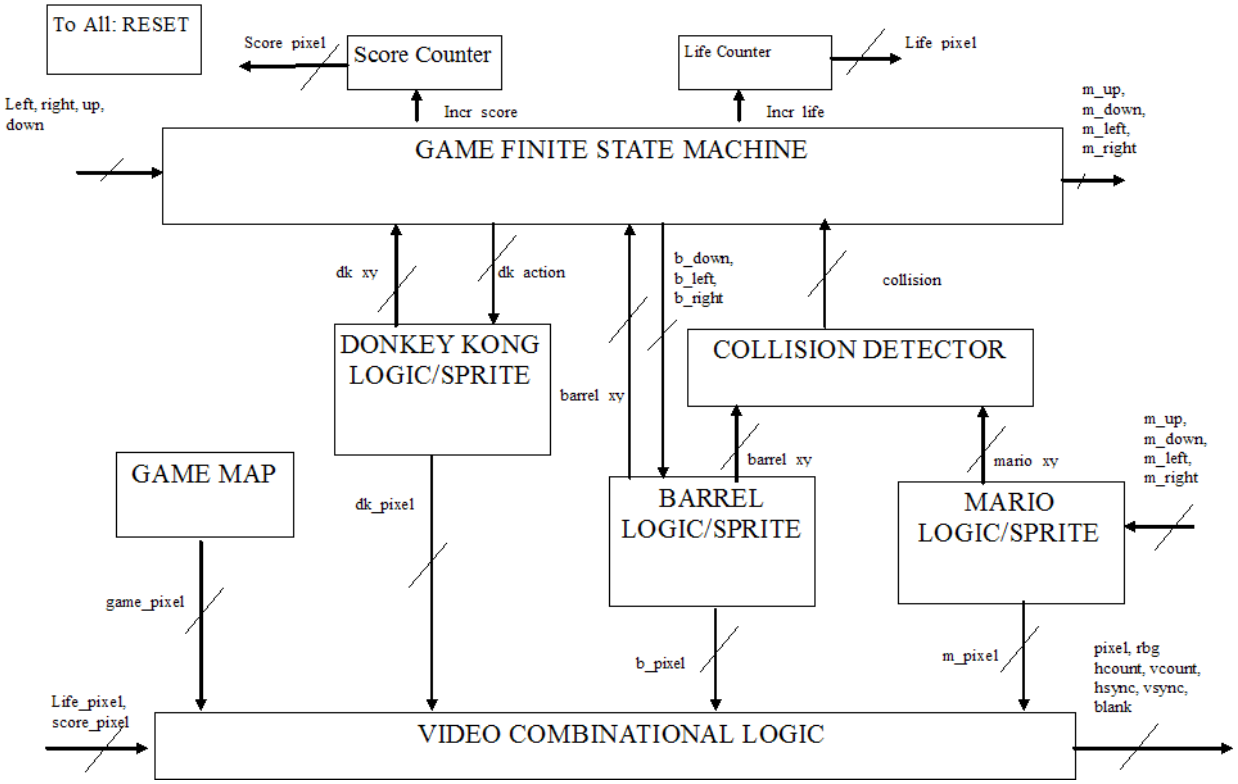

Figure 3: Game Logic Block Diagram

Testing

Our first step is to thoroughly plan the design. Then, we will code each module individually and separately. We plan to first develop the video component of the project. The visual aid will help us to more easily debug our game module. After completing each module, we will test and debug that module to make sure that it works. The process of creating a module and then testing and debugging it will be applied to all of the modules. Once the individual testing and debugging of all the modules is done, we will start to connect the modules together one at a time. We will test

and debug a group of modules together to make sure that they interact with each other as planned. Group testing will be done with all of the modules. Once group testing has been done in a satisfactory manner, every module that makes up the Donkey Kong game system will be wired together, tested and debugged. When the testing and debugging of the entire Donkey Kong game system is done, we will (hopefully) have a working and complete Donkey Kong game.

Division of Labor (*tentative*)

If necessary, both partners may be contributing to all parts of the system. Yi will focus on the Controller Interface, the Video Combinational Logic, and the Score and Life Counters. Stephen will focus on the Game Logic for Mario, the Barrels, and Donkey Kong; the Collision Detector; and FSM. We will both work on the Game Map. Please note that this is a tentative division, and it is subject to change.