Team Metal Squad Final Report

Project: Street Fighter 1.5

Team Members: Tarana Chowdhury, Igor de Paula, Dung Pham, Alexis Weaver

Project Description

Our project was to create a stripped-down, two-player version of the iconic Street Fighter II arcade game on the Nexys 3 Spartan 6 FPGA board. We have successfully developed a graphically complex arcade game that includes the characters Ryu and Ken. Each character has an idle state, jump, crouch, walk, kick, punch, hit, special, victory, and knockout state. The game has illustrations for the Street Fighter Logo, a KO logo, and Health Bars. Each character is initialized with full health and every attack move (punch and kick[1]) decrements the health bar, with the special attack decrementing three times as many points as an ordinary attack. Ordinarily, a jumping character cannot be attacked, but when Ken uses his special move, Fiery Shoryuken, he can attack a jumping player. Additionally, instead of a kick move, Ken has an additional punch move. This was implemented in order to bring a variety of actions for each character. We have used the JAMMA control interface, of which we use the joystick inputs and three push buttons - two for each character's attack moves, and one to start the game.

Project Modules

processor.v

This main module acts as a container for all submodules (*vga_display.v* and *cellular_ram_controller.v*). The inputs include all the JAMMA signals, which are processed with a standard debouncer, the VGA inputs and outputs, and the cellular DRAM control signals.

cellular_ram_controller.v

This module allows use of the on board DRAM. It connects with the DRAM using the clock and a list of pre-specified signals (see attached CellarRAM Manual, Table 2, Page 8). We have implemented this controller to use asynchronous mode to request data using the address provided by the vga_display.v module. This module outputs all control signals to the DRAM.

vga_display.v

This module contains all logic for the game. It takes as inputs all the debounced signals from the JAMMA interface, the VGA interface signals, and the pixel value (pixel_val) provided by the DRAM controller output using the address requested by this module (ram_addr). It has registers for storing the vertical and horizontal location of each character, the health, the current move and state of that move, and the state of the game (start, pause, knockout).

The game starts in the pause state. In this state, the Street Fighter logo appears in the middle of the screen with both characters frozen at each side. Upon pressing the start push button the characters will unfreeze and enter their idle states. To reset the game, the start button can be pressed returning all register values to their initial conditions. The game enters the knock out state when either character's health bar reaches 0. The flashing KO symbol appears in the middle

of the screen, with the appropriate character in the knockout state and victory state. In order to restart the game, the start button must be pressed.

Character moves:

All character moves are detected using an always block with a sensitivity list including all inputs from the JAMMA interface. The current move register is updated according to the input. After the character cycles through the corresponding frames for each move, it returns to its idle state, except for the victory and knockout moves which continuously loop until the start button is pressed. In the hit response state, the logic checks the current move and both the vertical and horizontal location of the other character in order to detect the collision.

Character states:

Using an 8Hz clock in an always block we cycle through all the frames for each character move by updating the corresponding move state register. For the jumping and walking moves the vertical and horizontal location registers are also updated respectively. Note: A faster game can be achieved by simply using a faster clock; it does not have to be particularly 8Hz.

Character display region and pixel address calculation:

The pixel address location of each character is calculated depending on the current move and the state of that move. The frames are defined using the corresponding sprite sheets that have been loaded onto the DRAM, and vary depending on the sprite sheets used. The pixel address is calculated depending on the horizontal and vertical coordinates of the VGA. In order to transpose the second character (Ryu, in our case) so the characters are facing each other, the width of the character is calculated and then subtracted from the original pixel location. The display region is also calculated using the current location, state, and move of the character.

DRAM address muxing:

Depending on the region of display and of each character, the corresponding pixel address is selected from player one and player two. Note that since the DRAM has a 16-bit return value while we only have 8 bits for each pixel, two pixels are requested at every clock cycle by dividing the pixel address by two. Each pixel that is extracted is displayed sequentially.

Health Bars:

These registers start at 310 pixels horizontally and decrement at every character hit by 10 pixels (unless special move, which decrements by 30 pixels). At reset the health bars return to 310.

RGB assignment:

The RGB assignment depends on three sources. The first is between the player 1 and player 2 regions. If the VGA is within that region it outputs the corresponding values from DRAM (note that if the background color is detected then it is set to black). If the VGA is in the health bar region, the health bars are drawn. Lastly, if the VGA is in the KO or Street Fighter logo region then the proper logo is drawn depending on the state of the game (these two illustrations come from the block ram).

Loading the DRAM and Block RAM

The reduced sprite sheets for each character are loaded onto the DRAM using the Digilent Adept tool. The 16-bit .png files are converted to 8-bit binary files using a MATLAB program. Each sprite sheet must be loaded in the exact location specified by the vga_display.v module. The same MATLAB program is used to create .coe files from the .png files of the KO and Street Fighter logo images to make a block RAM module using the IP Core generator in Xilinx.

Instructions

- Compile the processor.v module to generate a bit file.
- Go to the "Memory" Tab of the Digilient Adept tool
- Select RAM on the right side
- Under "Write File to Memory"
- Write *ken_reduced_odd.bin* to address 0 (decimal) in the DRAM
- Write *ryu_reduced_odd.bin* to address 950000 (decimal) in DRAM.
- Go to the "Config" Tab
- Load *processor.bit* file to program the FPGA
- Click "Program"

Contribution Breakdown

Tarana: character graphics, moves, testing

Dung: character graphics, moves, testing

Igor: game design and logic

Alexis: game design and logic