ECE 385

Spring 2020

Final Project

Final Project: Pac-Man

Aris Karnavas, Jeremiah Kwon

Section ABE / Tuesday 3-6PM

Jiaxuan Liu

Introduction and Plans

For the final project, a working version project of Pac-Man was the goal of the project. The working copy of Pac-Man would have movement according to user-inputed keystrokes, VGA display output, wall detection, removing pellets and detecting when all the pellets are gone, and simple AI to move the ghosts around with the sprites to make it look nice. The project would use hardware such as USB keyboard inputs, VGA display output values, and the on chip processor to move Pac-Man and to detect walls and pellets. The software component would be used to set up the USB keyboard and to determine where the ghosts would move to and other game states. Both the hardware and software components stated above should get the working version of the game. The final version of the final project was completed with the exception of working sprites.

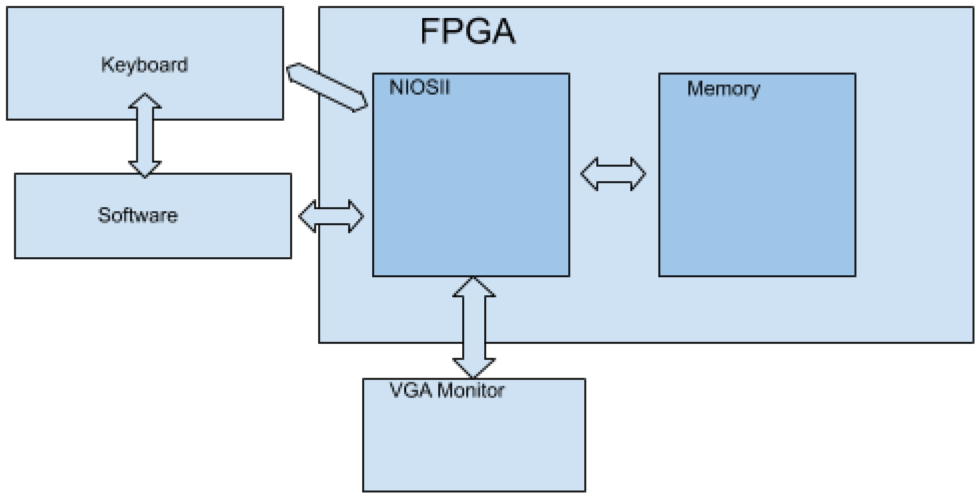
Written Description of the Hardware Components

The hardware components used modules to handle the base game processes and VGA display values and timing. The game processes that was handled by the hardware components included the wall and pellet values being stored in on-chip memory, the position of the player along with the new position, the basic outputs to tell the VGA display whether a pixel is a part of the player, wall, or pellet, and lastly the detection of walls and pellets. The hardware first starts out by initializing the module of on-chip memory that initializes the game board from a text file. This text file consists of array values that correspond to a specific tile, which is 11x11 pixels wide, and contains a two bit value. The first bit in the value corresponds to whether the tile should contain a wall or not; likewise, the second bit is used to hold whether the tile has a pellet or not. The processor then displays the values onto the VGA display with the help of a module that assigns simple colors onto the player, ghosts, walls, and pellets and the modules that control the VGA timing signals and the VGA clock. Wall detection works by reading the wall values at the index above, below, to the right, and to the left of the player. When the player inputs one of the direction keys, the processor checks the appropriate wall values to check whether there is a wall at the tile the player is trying to move to if there is then the player does not move to that tile. If the player can move to tile, then the pellet at the previous location is removed and the count of pellets left is reduced by one. The AI of the ghosts are controlled by the software code telling each ghost where to move. This involves sending the current position of the index the player is on and sending it the software portion of the project. When the software portion finishes computing the next location that the ghosts will move to, the processor will update the locations of the ghosts and check for collisions with the player. If there are collisions, then the player loses; elsewise, the game repeats the cycle of wall checking, movement, and AI movement.

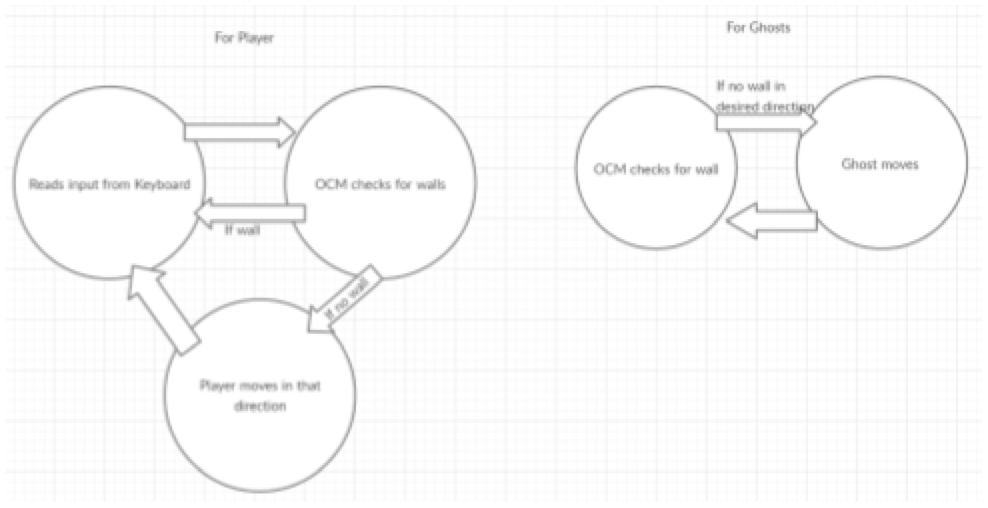
Written Description of the Software Components

The software components in the completed project of Pac-Man involves the code that allows the usb keyboard to properly read values from the specific registers that allows the FPGA to read keyboard inputs to send them to the processor, and the AI ghost collision. Keyboard input is done by having functions that reads and writes data to specific addresses like that from lab 8 with the exception that the player moves for every keystroke instead of changing directions on the keystroke. In addition to that, there is the code that computes the direction the ghost needs to move in order to get closer to the tile location the player is currently at. This information is then used with the grid the game is played on in order to figure out the next best moves of the ghosts and compares where the player is in relation to where the ghost currently is. This data would be sent back to the module that controls the basic game processes. From there the data would be used to update the motions of the ghost and to check if the new locations of the ghosts lines up with that of the player to see whether the player has lost or not.

Block Diagram



State Diagram of Pac-Man Movement Cycle



Module Descriptions

Module: final

Inputs: input CLOCK_50, input [3:0] KEY, input OTG_INT,

Outputs: output logic [6:0] HEX0, HEX1, output logic [7:0] VGA_R, VGA_G, VGA_B, output logic VGA_CLK, VGA_SYNC_N, VGA_BLANK_N, VGA_VS, VGA_HS, output logic [1:0] OTG_ADDR, output logic OTG_CS_N, OTG_RD_N, OTG_WR_N, OTG_RST_N, output logic [12:0] DRAM_ADDR, output logic [12:0] DRAM_ADDR, output logic [1:0] DRAM_BA, output logic [3:0] DRAM_DQM, output logic DRAM_RAS_N, DRAM_CAS_N, DRAM_CKE, DRAM_WE_N, DRAM_CS_N, DRAM_CLK, output logic wall

Inout: inout wire [15:0] OTG_DATA, inout wire [31:0] DRAM_DQ,

Description: The inputs and outputs are fed into their corresponding submodules.

Purpose: The top level module that connects all the modules together.

Module: hpi_to_intf

Inputs: input Clk, Reset, input [1:0] from_sw_address, input [15:0] from_sw_data_out,

Input from_sw_r, from_sw_w, from_sw_cs, from_sw_reset, // Active low

Outputs: output[15:0] from_sw_data_in, output[1:0] OTG_ADDR, output OTG_RD_N, OTG_WR_N, OTG_CS_N, OTG_RST_N

Inout: inout [15:0] OTG_DATA,

Description: This module connects each of its inputs and outputs to the final_soc module in order to act like the interface between the EZ_OTG chip and NIOS II.

Purpose: This module is the interface between the keyboard and the processor.

Module: vga_clk

Inputs: input inclk0;

Outputs: output c0

Description: The module, vga_clk, takes the input clk and slows it down to 25 MHz in order to be used by the output VGA_clk

Purpose: This module creates a 25 MHz clock that is used to run the VGA components.

Module: VGA_controller

Inputs: input Clk, Reset, input VGA_CLK,

Outputs: output logic VGA_HS, VGA_VS, output logic VGA_BLANK_N, VGA_SYNC_N, output logic [9:0] DrawX, DrawY

Description: This module takes all the inputs and outputs needed to run the VGA display and sets them up so they can be seen on a monitor.

Purpose: This module does all the required things to have each component appear on the monitor.

Module: color_mapper

Inputs: input is_ball, is_wall, is_pellet, is_ghost1, is_ghost2, is_ghost3, is_ghost4, input [9:0] DrawX, DrawY,

Outputs: output logic [7:0] VGA_R, VGA_G, VGA_B

Description: This module maps the colors onto certain objects by using a given RGB value.

Purpose: The color_mapper module allows the VGA components to assign color to the respective objects.

Module: ball

Inputs: input Clk, Reset, Frame_clk, vga_blank_n, input [9:0] DrawX, DrawY, input [7:0] keycode, ghost1, ghost2, ghost3, ghost4,

Outputs: output logic is_ball, is_wall, is_pellet, is_ghost1, is_ghost2, is_ghost3, is_ghost4, wall, output [10:0] ball_index

Description: This module feeds the inputs into the wall module and the outputs back into the final module as well as calculating player movement.

Purpose: The ball module is used as the top level module for all the game states and handles the movement of the player.

Module: wall

Inputs: input Clk, Reset, frame_clk, vga_blank_n, input [9:0] DrawX, DrawY,

input [9:0] Ball_X_Pos_curr, Ball_Y_Pos_curr, input [7:0] keycode, ghost1, ghost2, ghost3, ghost4,

Outputs: output [9:0] Ball_X_Motion, Ball_Y_Motion, output [10:0] ball_index,

output is_wall, is_pellet, is_ghost1, is_ghost2, is_ghost3, is_ghost4 , wall

Description: This module handles the main game logic by reading the keycodes along with the player's current position to see if the current move is valid. This module also checks the values of the map to see if there are pellets and removes when it should as well as checking if the player is touched by a ghost.

Purpose: This module handles all the wall and pellet detection as well as the locations of the ghosts.

Module: ghost

Inputs: input [7:0] ghost_direction, input Clk, Reset, frame_clk, vga_blank_n, start,

input [9:0] DrawX, DrawY, input [10:0] start_index,

Outputs: output [10:0] ghost_index, output is_ghost

Description: This module checks the direction that is inputted by the wall module and updates the current location of the ghost afterwards. The ghost module also outputs whether the current pixel from the VGA monitor has a ghost or not.

Purpose: The ghost module handles displaying the ghosts onto the screen as well as updating the location of the ghost.

Logic Unit Statistic Chart

LUT	4827
DSP	60 total
Memory(BRAM)	16,507 bits
Flip-Flop	2277
Frequency	113.2MHz
Static Power	105.25mW
Dynamic Power	0.94mW
Total Power	174.40mW

Conclusion

In conclusion, the goal we set out to reach for the final project was achieved with the exception of the sprites for the player, pellet, ghosts, and walls. Most of the things sought after were completed such as proper movement, VGA display values, wall detection, and pellet detection and removal, and AI ghost movement. One of the major bugs that took a long time to solve and fix was proper wall detection. The problem was that wall detection was occuring one cycle too late which allowed the player to move one tile into the wall before getting stopped by the wall. This bug was found after pellet detection and removal was completed since the project would replace the first wall with a tile with no pellet due to improper wall collision. In order to fix this bug the movement of the player needed to be delayed until all values of the walls were read, and after this change was implemented proper wall collisions occurred.