Guitar Hero: Nursery Rhyme Edition

Project Proposal

Emily Hwang, Judy Ho

Introduction

This project will implement a version of the popular game Guitar Hero. The project will be composed of visual, audio, and interactive parts. The goal of the game is to be able to "play" the notes of a song chosen from a given list by pressing the appropriate buttons on the keyboard to match the ones on the screen.

On the PS2 keyboard, keys 1-8 will be used for the "notes" and Shift will substitute for strumming of the strings of the guitar. This allows the player to hold the keyboard like a guitar. Shift must be pressed for the note to be interpreted as "played."

Eight different tones will be matched with eight different 4 bit codes. A buzzer tone that indicates incorrect notes played by the user will also be a 4 bit code. Simple nursery rhyme songs will be created with these notes, such as Mary had a Little Lamb, Twinkle Twinkle Little Star, etc, and stored in a BRAM. The song will play as the user is playing the correct notes.

For the visual part, the appropriate notes for the song will be displayed, and scroll to the bottom of the screen as they should be played. The note should be played when the note reaches the designated area at the bottom of the screen. An additional feature will show that the correct note was played by slightly changing the color of the note in the designated area.

Additional features may include different levels of difficulty that would involve increasing the speed of the songs, and smaller detection window to hit the note accurately. A score module can also be included to keep track of the points depending on the player's accuracy. Another feature would include a rhythm for the song so that a note could sound for longer period of time and the visual aspect would indicate longer notes to the user.

Modules and Division of Labor

Keyboard

This module will be used in order to interface with the labkit. The shift key must be pressed in addition to the appropriate number key. The combination of buttons pressed generates a certain user code that will be mapped to the appropriate note code. These note codes will then be outputted to the Game Logic module to be compared to the note that should be played.

Memory: Judy Ho

- *BRAM*
  The BRAM will contain all the different song selections that we want to include for the user. It will output the encodings for the notes to the Game Logic module to be compared to user inputs as well as the video display, and it will take an address input from the Game Logic to select the correct notes within the memory each time the address increments.
- *Song Selection*
  This module allows the user to decide what song he would like to play. The switches will be used to indicate which song has been selected, and it will be referenced to an assigned song within the module that would indicate the appropriate start and end addresses for the

song. The switches will be the input, and the start and end addresses will be the outputs to the Game Logic.

Audio: Judy Ho

- *Audio*
  This module only has one input coming from the Game Logic. This input is the note encoding that is mapped to the appropriate frequency. This frequency is then outputted to the AC97 module.
- *AC97*
  This module transmits the data received from the Audio module in order to output it to a speaker that is plugged into the labkit.

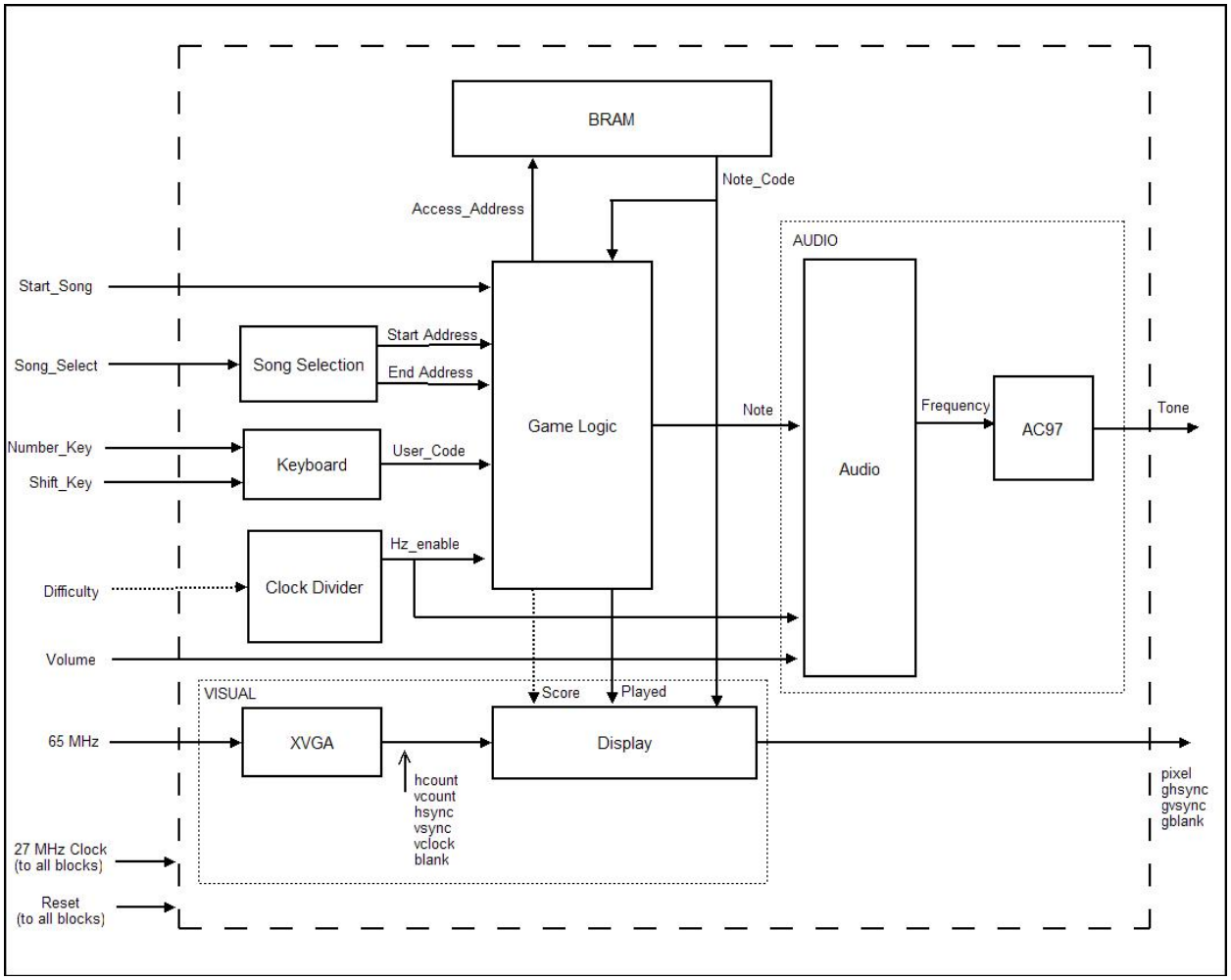
Game Logic: Emily Hwang

This module determines the note to be outputted to the audio module. The output note is based on the input of the user's note and the note that would be currently in the designated area of the display, which was previously received from the BRAM 8 seconds before since there will be 8 notes displayed. The Game Logic is given the inputs of the start address and end address from the Song Selection module. It will start incrementing the start address at every second until the end address when the start_song signal is asserted. It continues to increment when the one Hz enable is asserted. This address is used to extract data from the BRAM to receive the note coding at every second. If an additional feature of a score keeper is added, it will also output the current score.

Visual: Emily Hwang

- *XVGA*
  The XVGA from Lab5 will be used to generate the necessary signals for video display. These signals include hcount, vcount, hsync, vsync, vclock, and blank, which are outputted to the Display module.
- *Display*
  The Display module displays the scrolling notes to be played. It will at most display 8 notes. These notes will be received from the BRAM, where the access addresses are determined in the Game Logic module. The current note received will be newly displayed at the top of the screen while the previous notes will continuously scroll downwards until it reaches the bottom of the screen. Additional features will provide the slight change of the color of the correct note, if played. The outputs will be ghsync, gvsync, gblank, and pixel.

Block Diagram



External Components

A PS2 keyboard that would simulate the guitar controller would interface with the labkit. The address of the necessary key strokes would be identified as the corresponding four bit key that distinguishes each note. Since the act of strumming on a guitar is essential, it is necessary to know the addresses of the combined key and strum signals. Therefore, we would use the shift key as the strumming signal. Pressing the number keys 1, 2, 3, 4, 5, 6, 7, 8 would not be interpreted as a note played, but additionally pressing the shift key would create !,@#,$,%,^,&,* addresses that would be used to create the four bit numbers for the game logic.

Testing/Debugging

Keyboard

To test that the implementation of the keyboard works appropriately, we would display the four bit number created from the pressing of the PS2 keyboard keys. The display would only appear when the keyboard keys are pressed. Therefore, we would test each input: !@#$%^&* and verify that the inputs create output keys 1, 2, 3, 4, 5, 6, 7, 8. No other inputs of the keyboard should create an output key, not even the input of 1,2,3,4,5,6,7,8 because the strum button (shift) is not included so the input should not be processed.

Audio

The audio module would be tested by using the switches on the labkit as the encodings for the notes that should be outputted. The tone corresponding to the 4 bit encoding described by the switches will be audible through the speakers.

Visual

The visual module would be tested by inputting the four bit key through the switches of the labkit. At every second, by a Hz enable signal, the module will sample the key indicated by the switches. At every second, the visual component should process the new key and display this at the top of the screen. The previous notes should also be continuously scrolling down and the note should not change. Each note will be tested to verify the corresponding placement of eight different vertical positions on the screen.