Bradlee Harrison, Adam Tran
ECE 4273 - Digital Design Lab

Digital Dance Lab (DDL)

Final Project Design Report

Purpose

The goal of our project was to emulate an arcade-style music rhythm game based on the music video game series *Dance Dance Revolution*. An example of this arcade game can be seen in Figure 1. The goal of the game is to tap arrows on the pad with the correct timing using your feet when prompted by arrows displayed on a screen.

Figure 1: Dance Dance Revolution Machine

We accomplished a basic demonstration of this using our LPC1769 microcontroller by interfacing with a serial PlayStation 2 Dance Pad controller, a speaker output, and a VGA monitor. The display is output with a DE-15 VGA Connector connected to the FPGA with the transmitter connection from the microcontroller to the GPIO[1].

Directions

- Turn on the microcontroller and compile the main code.
- Press the SELECT button to select the song from two different tracks.
- Press the START button to start or restart the song selected.
- Press the TRIANGLE button to change from display mode to music mode. In display mode, the display will show the arrows, and in music mode, the display will freeze and the music will play.
- Press the CIRCLE button to increase the music volume.
- Press the X button to decrease the music volume.
- Press the LEFT arrow to decrease the music tempo.
- Press the RIGHT arrow to increase the music tempo.

Software Description

The software implements the primary function of playing music using the Digital-to-Analog Converter present on the LPC1769. When an input switch is triggered to begin the audio playback, we start a timer which will control all of the audio timings for the audio output and begins the audio playback. Using other inputs, the volume can also be controlled using the X and Circle pads of the controller mat as well as the Left and Right pads for the tempo of the music.

A variety of different musical notes are created by pulsing square waves at varying frequencies. To set the list of notes that will be output when the music is played, we created two arrays of notes and defined each note with a set amount of time to wait in between pulses. Multiple channels are able to play at the same time, generating harmonies and chords. A percussion sound is generated using a linear feedback shift register. Each channel of audio has varying levels of decay, which gradually pulls the amplitude of the square wave lower.

The software implements the function of displaying a graphical video output to a display using a data connection to a basic VGA controller implemented on a DE10-LITE FPGA board. The board controls the timing and horizontal and vertical sync required for a 640 x 480 resolution VGA interface. To reduce the memory usage, we save the pixel values in an array of 320 x 240 memory, with two bits of color information per pixel. This results in only 57.6 kB of memory required to store all pixel values. The results of the VGA controller's horizontal sync, vertical sync, and color data output can be seen in the following Figures 2, 3, and 4.

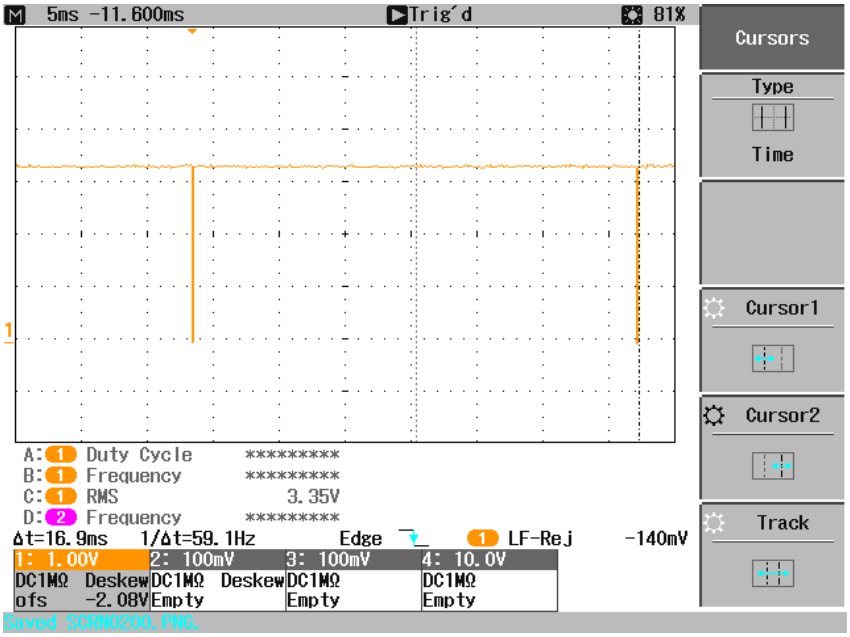

Figure 2: Vertical Synchronization for VGA

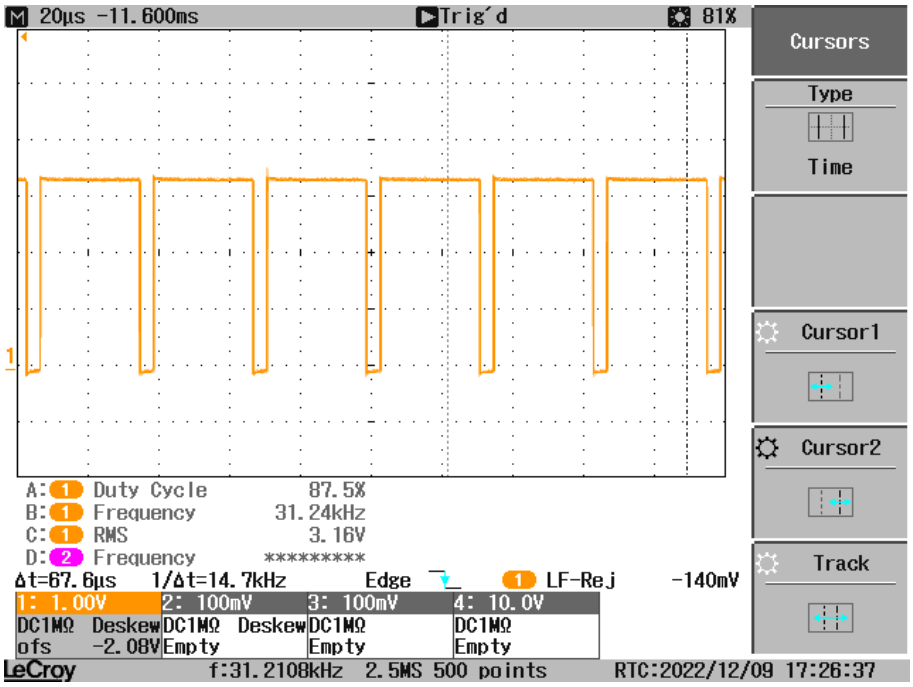

Figure 3: Horizontal Synchronization for VGA

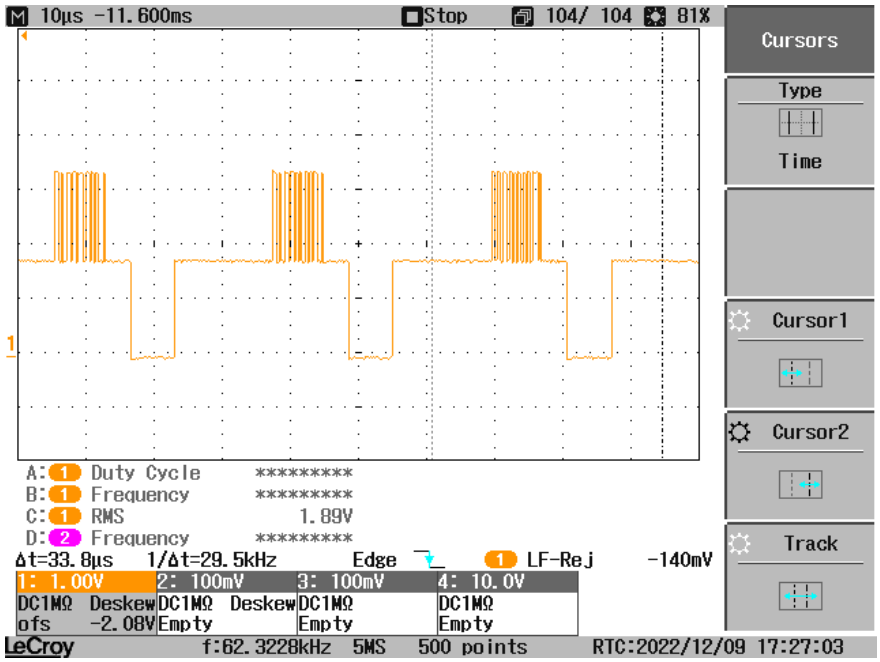

Figure 4: Color Data Line for VGA

To communicate with the DE10-LITE FPGA board, we implemented a UART interface running at 900,000 baud rate. The UART transmission can be seen in the oscilloscope screen capture in Figure 5. The board receives 8-bit samples from the LPC1769 to indicate the position of a rectangle to draw in the memory. The bytes transmitted are in series. The first byte sent is an indicator of 0xFF to start the transaction. Following, the initial x position, final x position, initial y

position, final y position, and the 6 bit color information are sent to the DE10-lite board through a GPIO input pin, which then has a UART receiver and decoder to write the received values as blocks into the board's memory.

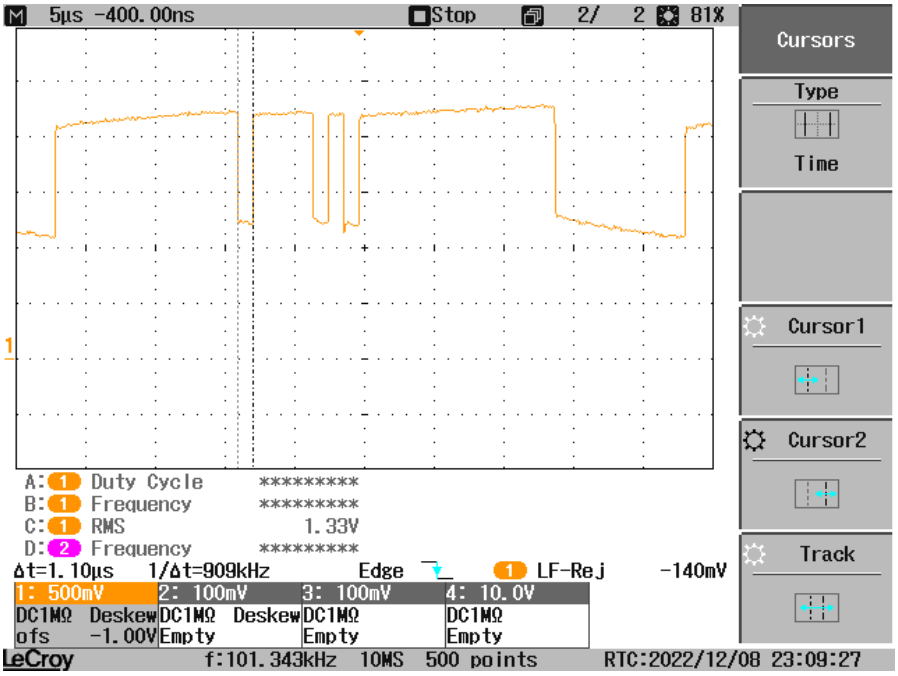

Figure 5: UART Transmitter LPC1769 to GPIO[1] of FPGA

The software implements the function of receiving digital input from a PlayStation 2 Dance Dance Revolution Regular Dance Pad. This dance pad uses a serial interface to send input signals. Using an SPI connection, the LPC1769 functions as the controller, and the dance pad functions as a target. The LPC1769 polls the dance pad using a series of command words and receives 5 bytes of data from the dance pad. The 4th and 5th bytes of data contain the information for the buttons pressed on the dance pad, including the left, up, right, down arrows, start, select and the cross, circle, square, and triangle buttons. Based on the inputs from the dance pad, we save the state and detect the edge when the switch is pressed by checking when the signal changes from low to high.

Hardware Description

As seen in Figure 6 and Figure 8, our audio hardware uses a similar schematic from our assignment #5 with the audio amplifier connected with our speaker. We wired our PlayStation2 Dance Pad with an SPI connection to the LPC1769 with SPI IN Command into pin 11, SPI OUT Data into pin 12, Clock into pin 13, and Attention into pin 15. The Acknowledge connection as well as the vibration motor connection were not needed. Our connection wires of the PlayStation2 Dance Pad were not color coded properly as it was not the standardized dance pad. The only difference of our connections compared to the standardized dance pad connections was that the ground and the vibration motors were switched. A pull-up resistor

value of 1k ohms for the Data connection of the SPI was needed due to an open collector output of the lines, which cannot drive voltages.

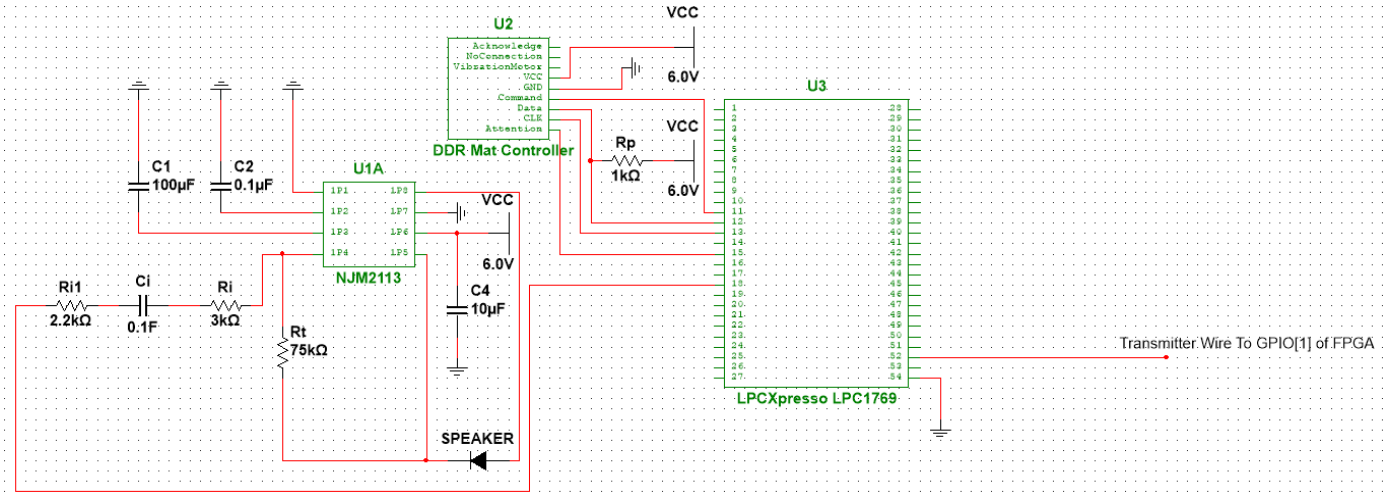

Figure 6: Complete LPC1769 with NJM2113 Audio Amplifier Schematic, Playstation2 Dance Pad, and Transmitter Wire

For our VGA display output, a jumper wire was used to connect to the DE10-Lite GPIO[1] pin and another jumper wire was used to connect the grounds between the DE10-Lite and the LPC1769. The VGA connector on the DE10-Lite provides an interface for analog VGA output. We wired our display to the connector on the end of this board. The pins necessary for basic VGA output that was controlled by our DE10-Lite VGA controller can be seen in Figure 7.

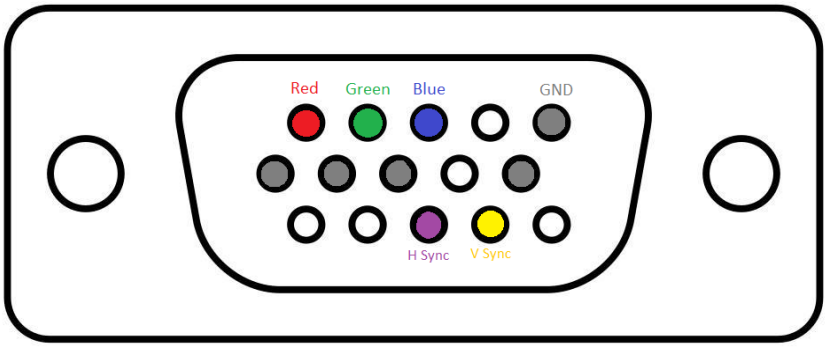

Figure 7: DE-15 VGA Connector Pin Descriptions

For the audio amplifier circuit, we followed the application circuit from the NJM2113 data sheet shown on Figure 4, but we replaced the capacitor C1 with a 100uF capacitor instead of a 1.0uF capacitor in order to improve power supply rejection ratio and used a C2 of 0.1uF instead of 5uF.

■ APPLICATION CIRCUIT

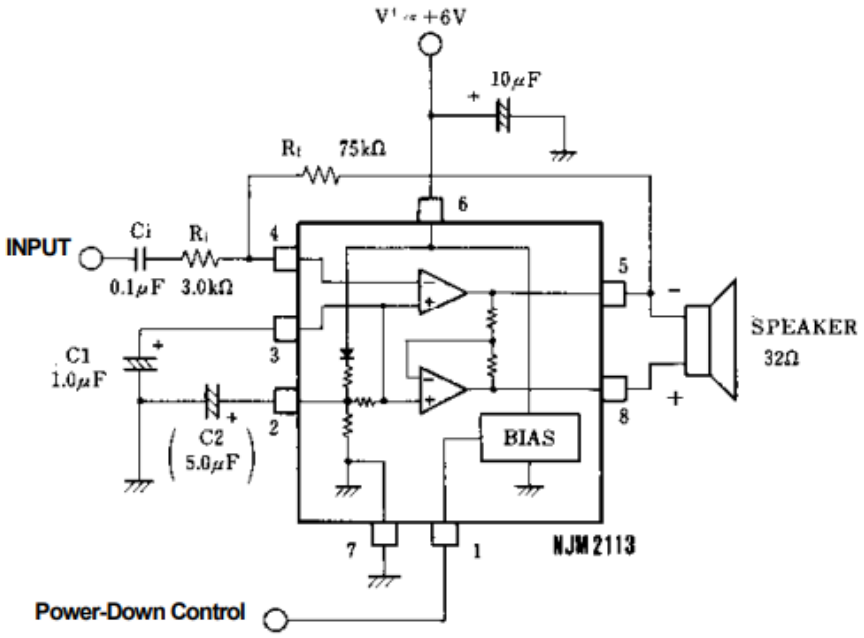

Figure 8: Application Circuit from NJM2113 Data Sheet

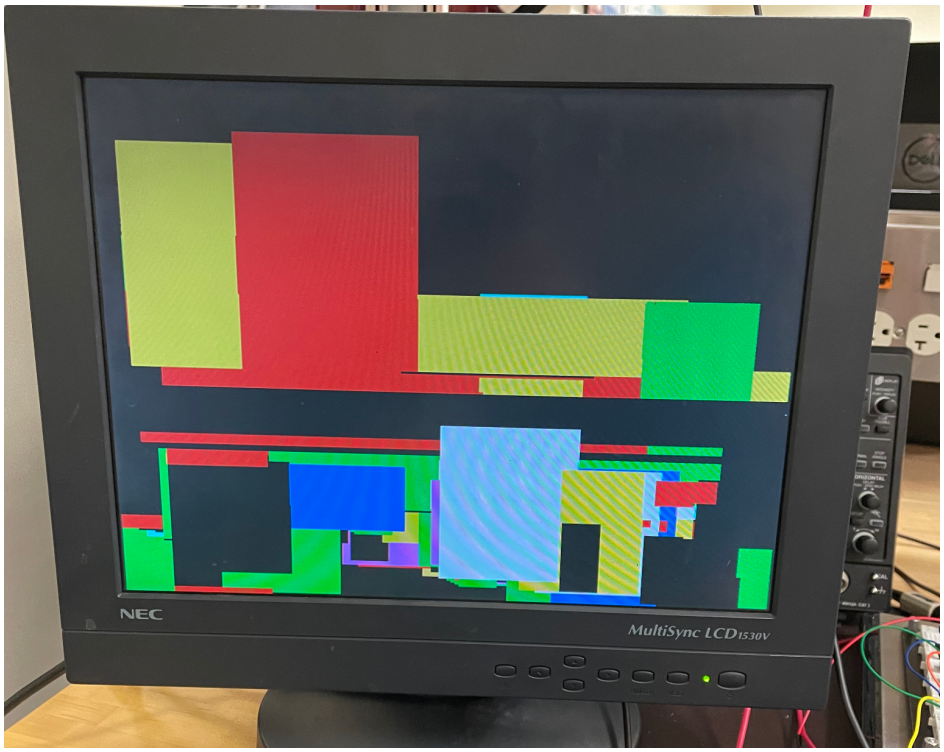

Figure 9: Screen Showing Test Mode

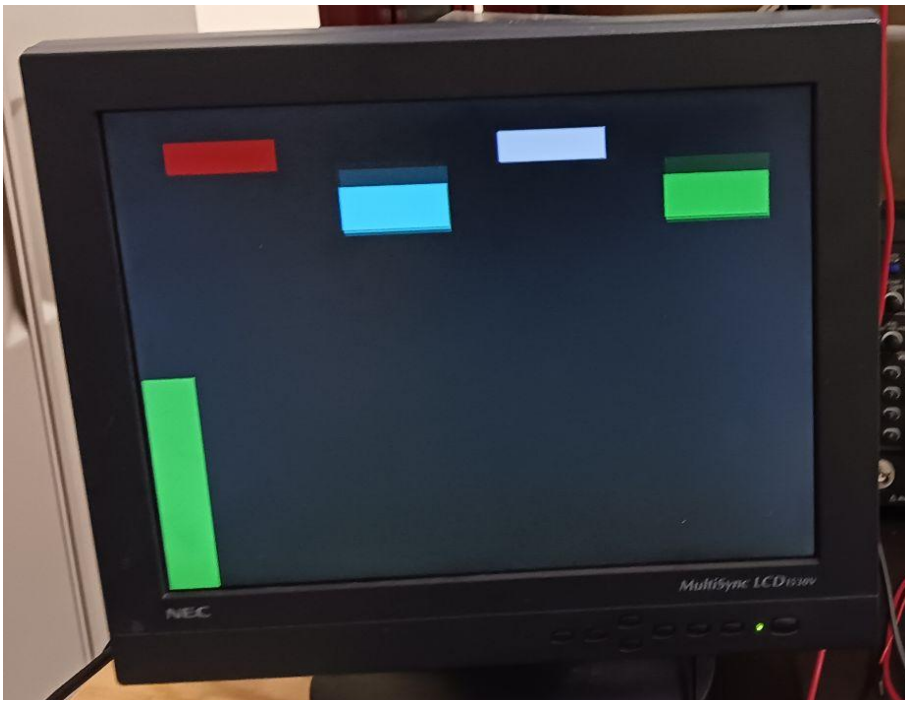

Figure 10: Screen Showing Columns of Beat Indicators Scrolling up

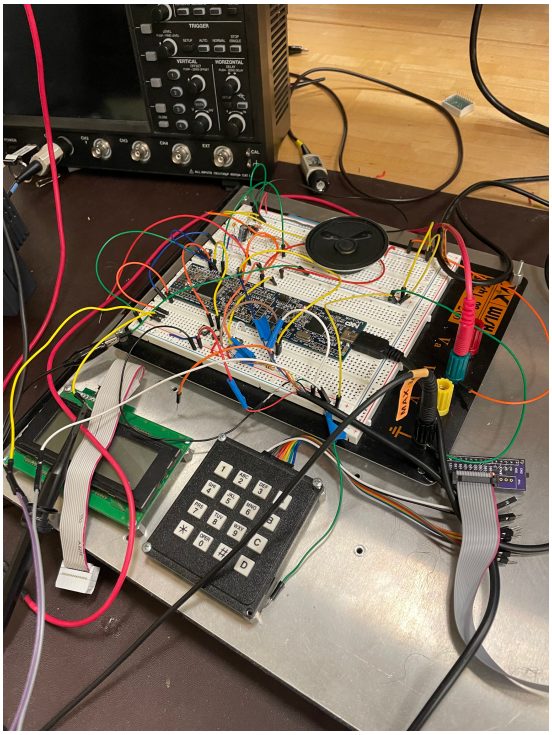

Figure 11: Breadboard Setup

Bradlee Harrison, Adam Tran
ECE 4273 - Digital Design Lab

Sources Used

https://blog.thomaspoulet.fr/bit-banged-vga/ - We consulted this source often when attempting to implement VGA using the LPC1769.
https://hackaday.io/project/170365-blueretro/log/186471-playstation-playstation-2-spi-interface - We followed this source's instructions when implementing our playstation 2 dance pad controller
https://ktln2.org/2018/01/23/implementing-vga-in-verilog/ - We used this instruction guide and the VGA examples to generate our VGA output interface using the DE10 Lite board.