FPGA Hero! (Guitar hero)

Rock of Gates Edition

Martin Smoger, Jonathan Williams, Devin Poirier, Fatemah M A Y H Almarhoun
Electrical and Computer Engineering Department
School of Engineering and Computer Science
Oakland University, Rochester, MI
e-mails: mvsmoger@oakland.edu, jtwilliams@oakland.edu, devinpoirier@oakland.edu, Falmarhoun@oakland.edu

***Abstract*—The purpose of this project is to create a functional clone of the Guitar Hero video game utilizing several pieces of hardware. We send a sequence of signals to power LEDs at which they are lit up in a certain order as to follow a beat or a song. The overall challenge of this project was getting all hardware to communicate flawlessly, requiring different communication protocols to function successfully. The end result is a fully functional game that can be modified and adapted to play other beats or songs as well.**

I. INTRODUCTION

Our goal with this project is to create a game based on the Guitar Hero game series. Using several pieces of hardware, we are able to implement and execute a video game on our FPGA board. The main motivation for this project was to take a more complex video game and design and work with it in a VHDL environment. We wanted to challenge ourselves to come up with something unique to any other project. With this idea, this really pulls together a lot of different topics that we covered in class. From I2C communication, finite state machines, a seven-segment display serializer, UART and more. We even had to use external resources to figure out how to properly connect to the user's controller (the guitar) as well as reading the actual data from it itself.

II. METHODOLOGY

A. Broad Idea

The basic idea behind this project is that the Guitar Hero controller will send and receive several signals that will communicate with the FPGA. The guitar sends out a confirmation signal (which we denoted as green) that begins the game, and then there is also a strum signal, that designates the note being played. The other signals it sends out are simply the signals used for the colors, telling the FPGA what the user is currently holding and once strummed, comparing it to the current note that the user must match. If the user input matches to the corresponding lights, the user score will increment by one.

B. Design Ideas

We went through several design ideas that could possibly work for the project. For the hardware component, we wanted to be able to communicate freely with the Guitar Hero controller. After conducting research, we concluded that the guitar will function with the I2C communication protocol [1]. We needed to figure out a way to get the FPGA to read these I2C signals, as there isn't a simple pinout on the FPGA for I2C. As such, we resorted to using an Arduino Uno R3 due to the fact the complication of adding a I2C protocol module in VHDL. To connect the Uno to the FPGA, we simply used the UART communication protocol that is available with the FPGA.

For the lights, we wanted to use a light matrix that we had available. It is an array of 7 rows of 5 lights. The original plan was to use a series of relays and transistors to boost and protect the signal required for the light matrix (which was pulling almost 350W!) but unfortunately it broke so we shifted over to bread boards. We used a series of colored LEDs (Green, Red Yellow, Blue, Clear) and resistors and wired them on a breadboard to create the sequencing that we needed.

To display the score and the countdown from ready state, we choose to use the serializer. We only needed to utilize two seven segment displays for which one on the left displays score and one on the right to show the countdown before the game starts.

It should be stated that both the UART_RX and the serializer modules used in our system were already developed available code. The serializer came from a Workshop examples website provided from our Professor [3] and the UART_RX module came from an online source in which we deemed would fit in to our project [2]. The rest of the code was developed by ourselves.

C. Design Flow

See page 4 and page 5 for a full block diagram and FSM design. We also had attached the test bench for the final Top file simulation. We needed to convert our timers to nanoseconds in order to have a fast and efficient wave pattern. It would take several hours for the simulation to complete.

III. Experimental Setup

A. Hardware

This project is heavily focused on external hardware which requires accurate and precise integration, as each piece of hardware must communicate flawlessly to function properly. For hardware, we used the following:

1. Nexys-A7 FPGA
2. Arduino Uno
3. 2 Breadboards
4. Wires/LEDs/Resistors
5. Nintendo Wii Guitar Hero Controller

Using the Arduino Uno, we were able to communicate via the I2C communication protocol @ 400kHz with the Guitar Hero controller [1]. This required some additional research as to how the Guitar Hero controller actually functions with the Nintendo Wii. The guitar has two pins, SDA and SCL, required for the communication protocol [1]. To actually read the data being sent and received from the guitar, we communicated via UART between the Arduino Pro Mini and the Nexys-A7. To provide power to everything, we will use the standard UART pin on the Nexys-A7 while connected to the computer. As both the Guitar Hero controller and the FPGA runs off of 3v3, we needed to utilize TTL bidirectional converters between the controller and Uno and FPGA to Uno.

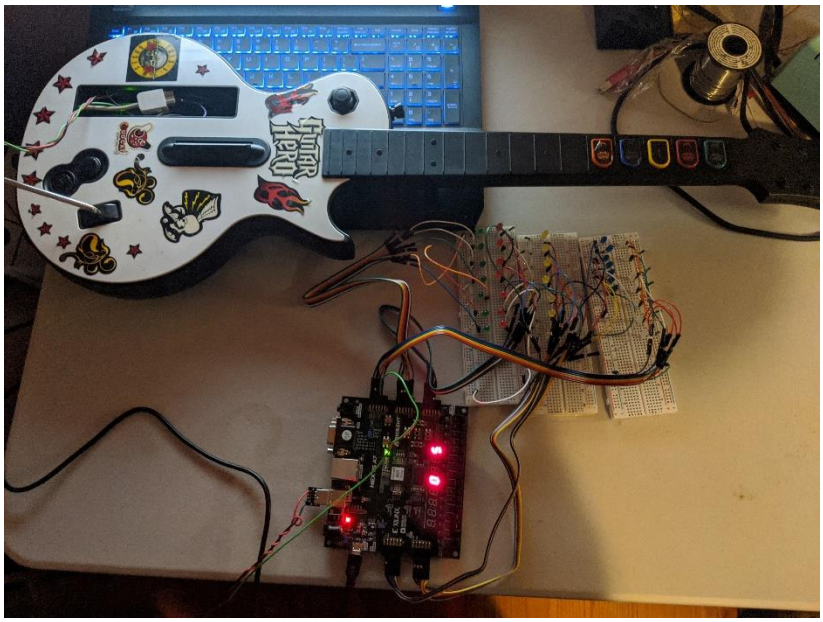


Figure 1: Final Experimental Setup

Using three breadboards and several resistors, we created an array of colored LEDs to simulate the song and the user input. We used the Pmod ports on the FPGA to hook the wires from the breadboards to the FPGA.

B. Software

For the software side of this project, we mainly used Vivado, but we also used the Arduino software with Wire.h library to design and execute the code on the Arduino itself. The code on the Arduino is simply reading the I2C outputs on the guitar itself, which would be the colors the user is holding as well as that strum signal.

The FPGA houses the main code, which includes three finite state machines, counters, gen pulse, a hex-to-seven-segment decoder, a serializer, a UART Rx, and a top file. One was for the actual state of the game, as to whether or not it is running, and the other two were for controlling state of the song the song. We needed to tell the FPGA to send out a specific array of signals to the breadboard to properly light the LEDs in sequence. For testing purposes, we hard coded a simple staircase pattern of colors, only lighting one color up at a time. The code can be expanded to use two or more colors. To create the sequencing of the lights, we wanted to hold a color for about one sec of LEDs. We cannot achond and send it to the next rowieve this using the internal clock of 100MHz, as it would simply be too fast for the human eye to even track it, let alone get any points. To achieve this, we used a Gen pulse that pulses every second to trigger the song FSM to go to the next state to display the next row of lights. This also gives the user en fingers to the correct notes on the guitar and strum oncough time to coordinate hise.

To display the score, we had to try a couple different methods. The simplest way we decided to do it was to create a serializer to use four seven-segment displays on the FPGA. Inside of the FSM for the song, we implemented a check to see if the user was correct. Upon strumming, it compares the note that the user is holding to the second to last row of LEDS (where the last row of LEDs is showing the user input) and if it is correct, send it to the top file. When the user was wrong, we did not change the score in any way. In the actual Guitar Hero game, the user will never lose points but if they continuously get notes correct, they get a score multiplier. We did not implement this in our project.

IV. Results

This project really pulled a lot of different topics together from the class. We expanded on I2C communication as well as the UART connection on

our Nexys-A7 boards. Our code itself used several pieces from topics we learned in class as well, such as arrays, FSMs, serializers, counters, and Gen Pulses. A video of our system working can be found in the folder attached with our VHDL code.

Conclusions

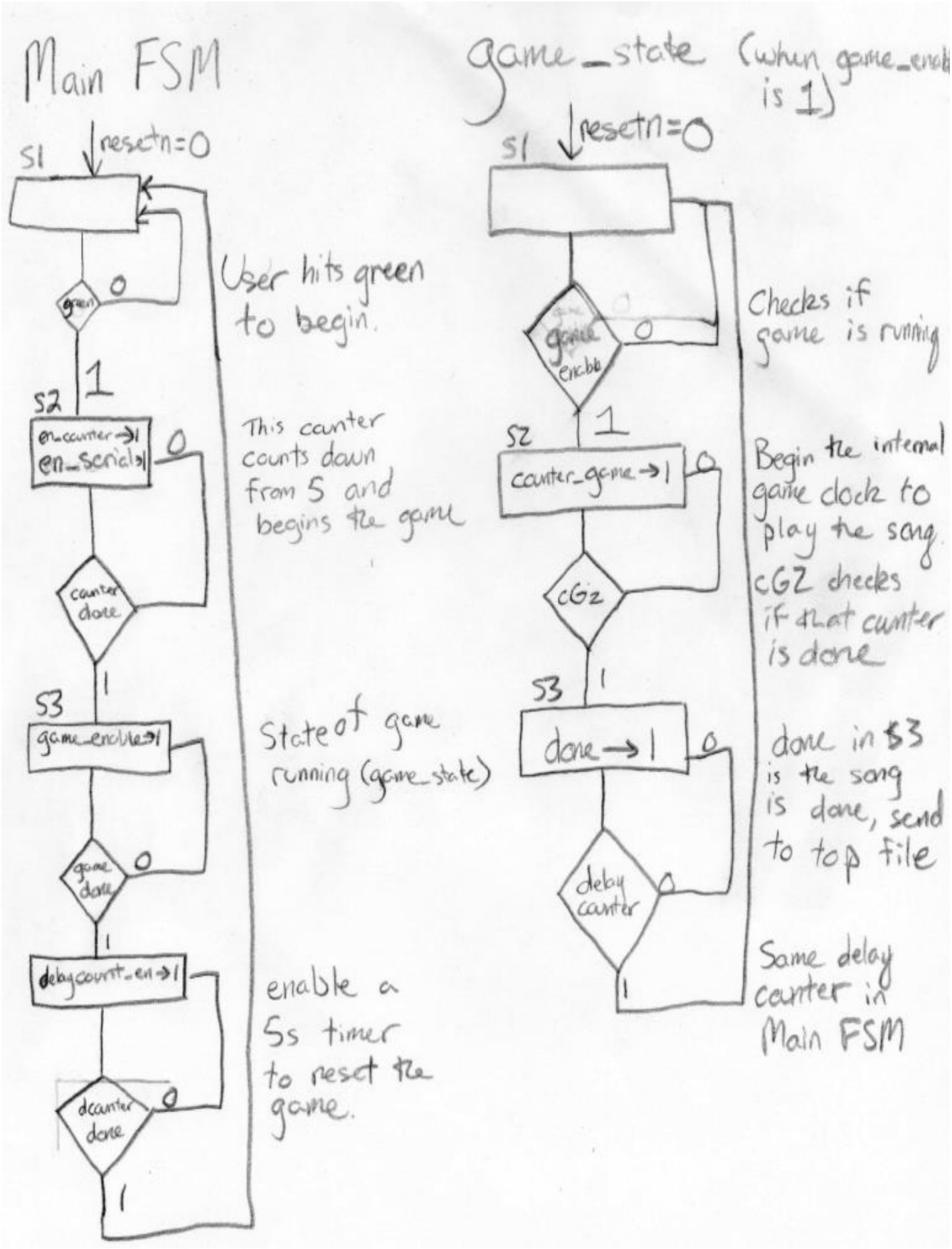
Deciding to choose a project like Guitar Hero has really reinforced our knowledge of the topics that we covered in class. It also helped us show how simple it is to really integrate different pieces of hardware. As long as you have access to the actual pinouts of the hardware (in our case, the guitar hero controller) it is super easy to. This was a great learning experience for all of us. One challenge we had with this project was managing two state machines at once. Once we figured out how to properly component them (which it isn't really different at all) the program worked. Another difficulty we ran into was debugging. When we are pulling from all of these different hardware components, it's difficult to try to debug and figure out where the problem is.

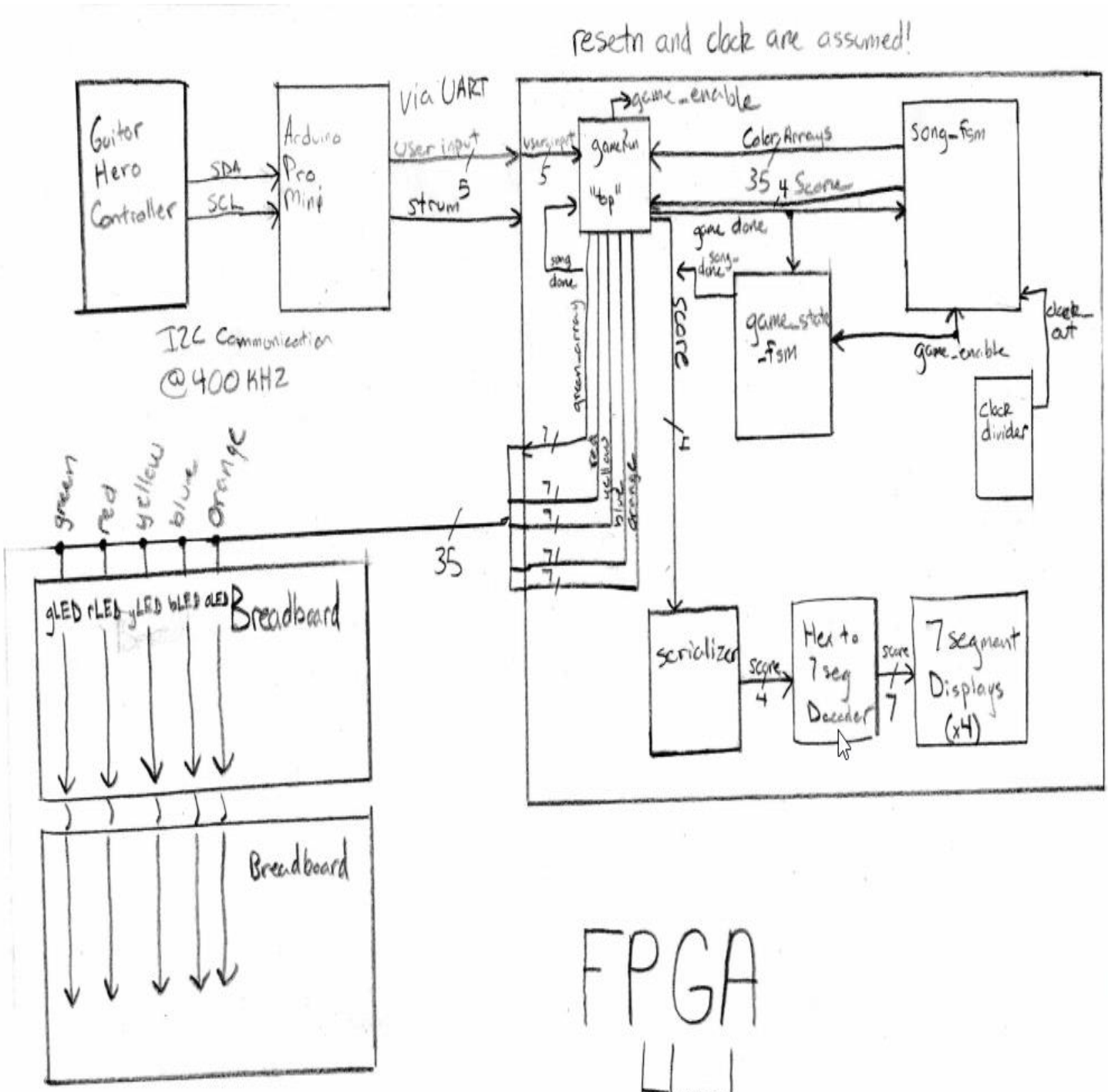
References

[1] WiiBrew. (2009, November 15). W*iimote/Extension Controllers* [Wiki page info]. Retreived from https://wiibrew.org/wiki/Wiimote/Extension_Controllers

[2] Russell. NAND LAND *UART, Serial Port, RS-232 Interface - Code in both VHDL and Verilog for FPGA Implementation* [Website blog]. Retreived from https://www.nandland.com/vhdl/modules/module-uart-serial-port-rs232.html

[3] Llamocca, D. (2013) RECRLab *VHDL Coding for FPGAs* [Workshop examples]. Retrieved from http://www.secs.oakland.edu/~llamocca/VHDLforFPGAs.html


Main FSM
S1
resetn=0
green
0
1
S2
en_counter→1
en_scrick→1
0
counter done
1
S3
game_enable→1
game done
0
1
debaycount_en→1
dcounter done
0
1
User hits green to begin.
This counter counts down from 5 and begins the game
State of game running (game_state)
enable a 5s timer to reset the game.
game_state (when game_enable is 1)
S1
resetn=0
game enable
0
1
S2
counter_game→1
0
cGz
1
S3
done→1
0
delay counter
1
Checks if game is running
Begin the internal game clock to play the song
cGZ checks if that counter is done
done in S3 is the song is done, send to top file
Same delay counter in Main FSM


resetn and clock are assumed!
Guitar Hero Controller
SDA
SCL
Arduino Pro Mini
I2C Communication @400 KHz
via UART
user input
strum
5
game_enable
gameFsm "top"
Color Arrays
35
4 Score
Song_fsm
game done
song done
song done
game_state_fsm
game_enable
clock out
clock divider
score
green_array
red
yellow
blue
orange
7
4
35
green
red
yellow
blue
orange
gLED rLED yLED bLED oLED
Breadboard
Breadboard
serializer
score
4
Hex to 7seg Decoder
score
7
7 segment Displays (x4)
FPGA Hero!
Marty Snoyer, Jonathan Williams, Devin Pariser, Fatemah Almarhoun