THE AWESOME GUITAR GAME

Design Document

Embedded System Design
CSEE 4840

Spring 2012 Semester

Academic supervisor: Professor Stephen Edwards

Laurent Charignon (lc2817)
Imré Frotier de la Messelière (imf2108)
Avijit Singh(asw2156)

Abstract

This document describes our preliminary project design implementation based on our research so far. In this project, we implement an interactive game where the user can play a dummy guitar to match up the notes as displayed on a screen. The expected music will be played softly in the background and will grow louder whenever the user presses the correct key. For this project, we will use an Altera DE2 board, a NIOS II processor, a dummy guitar, a VGA display and a pair of speakers.

1 Introduction

This game is inspired by the game Guitar Hero video game series [1] We use Guitar Hero controller for PlayStation 2 [4] to serve as the "dummy guitar", it has 5 colored buttons as well as a button to simulate the action on the guitar string. When the user starts the game he/she will hear the music. The screen would display a stream of notes. If the user presses the correct button at the correct instant, the particular note is added to the music and is played loudly. To make the user more involved, the display will reflect that the note has been correctly played.

The implementation of this project will be done on an Altera DE2 board. We will use a NIOS II soft processor as the central element of our design, and will write programs in C for this processor. The processor will interact with memory mapped peripheral modules implemented in VHDL.

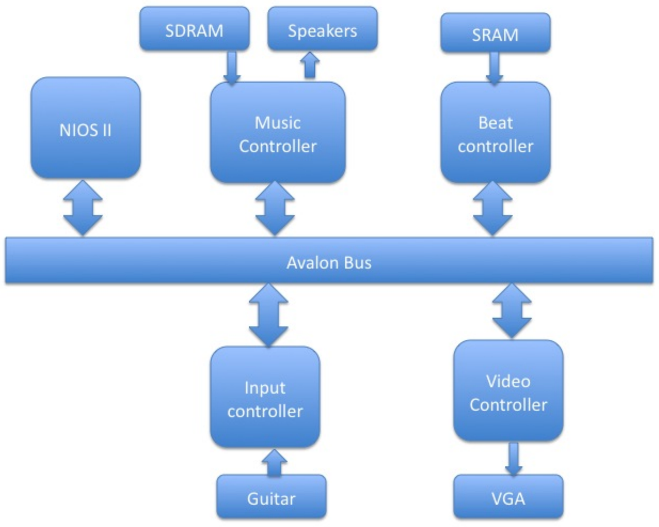
2 Hardware Implementation

2.1 Altera DE2 FPGA

The Altera DE2 FPGA will be used to include a NIOS II processor, that will interact with several controllers. It is the central part of our design and is reponsable for the following:

- get the user input through the dummy guitar,
- draw on the VGA display,
- control the speakers,
- contain all the information needed for the game (songs, beats and sprites) in its onboard SDRAM.

The NIOS interacts through an Avalon Bus with the controllers. This leads to the following hardware architecture:



2.2 Dummy Guitar and input controller

As said before, we use a Guitar Hero controller for PlayStation 2 as our dummy guitar [2]. The dummy guitar is used to get the user input in an enjoyable way for the player.

The controller will take the input from the the 5 buttons and when the string button is pressed it updates a memory location (assigned to it) with the color button pressed and sends an interruption to the processor. The interruption is caught and the click is processed. The processor then clears the memory location and the interruption. We will filter for the button bouncing effect at the controller level, not at software level. The connection from the guitar to the board will be one of following solutions: Using a custom cable that maps every button on a single wire using GPIOs or reusing whatever we could find in the guitar: USB, RS 232 or whatever we will find inside the guitar.

2.3 Audio format and memory dimensioning

The song is stored in the SDRAM in Binary format. It is sampled down from a sampling rate of 44.1Khz to 8 Khz in order to reduce the memory requirements for each song. We tested that the song still sounds good at a sampling rate of 8Khz. Every sample is represented on a 8 bits scale. The total memory requirements of a 3 minutes song are 1.44 MB(3*60*1*8000 bytes). This order of magnitude suggests that we use the SDRAM (we need more than the SRAM and less than the SDCARD). We will therefore be able to store a few songs.

We will use the SRAM to store the sprites and the beats. We will have at maximum 5 sprites of 32x32x8 bit (monochromatic sprites): 5kbytes. The size of the beats binary stream is not fixed as of now. More details are given at the end of this document in the section representation of the notes.

2.4 Video controller

The video controller is in charge of displaying the sprites on the screen. It is controlled by the processor with the following commands:

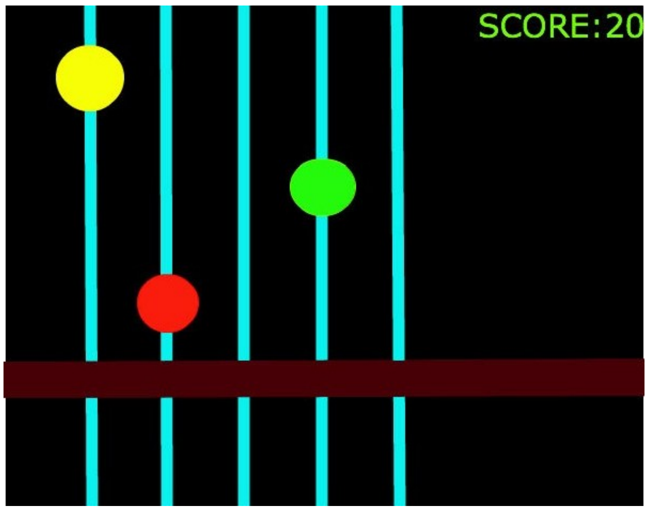
- Add a sprite with as parameter the color, the id
- Clear the sprite list

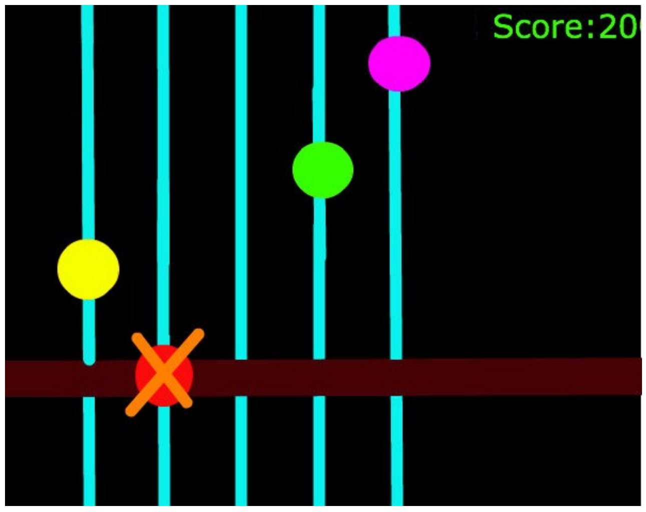
The video controller has a dedicated SRAM for the sprite. We will take as a model the video controller in the nintendo nes. We aim a sprite stack of size 4 (4 sprites in the same line maximum). We will have 4 custom sprites for the project: for the note, the note pressed, the note rightly pressed and the background. We will have sprites for the text and numbers in the same fashion than LAB2.

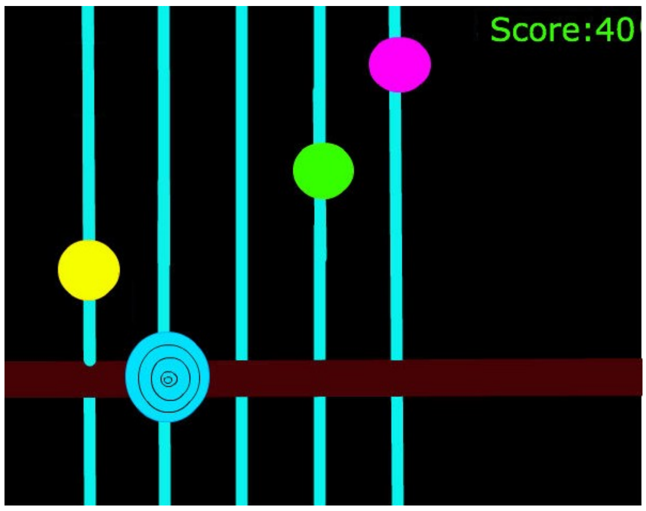
3 Software Implementation

3.1 The NIOS program: game loop and event handling

The program that runs on the NIOS processor will be in charge of the Main Game loop (generally used in video games). It will rely on a timer (either as an Avalon Peripheral) or a component of the NIOS processor if available to ensure a constant frame rate. Every iteration of the main loop, we will update the display by computing the new positions of the falling notes. Meanwhile, the program could be interupted to deal with key presses. Every time a key is pressed, a visual feedback is given for the few next frames and the new score is immediatly computed. The graphical interface will look like the following:


SCORE:20


Score:20


Score:40

3.2 Beat tracking

The instants at which the notes appear on the screen are not random events.They are synchronized with the beats in the song so as to give a more natural effect.The beats of the song are extracted by a Beat Tracking Algorithm we implemented in Matlab. [3] These beat instances are saved as a .txt file for further usage. They are extracted once and for all so that we do not need to use Matlab when running our game.

3.3 Representation of the notes

After having run the beat tracking algorithm on Matlab we obtain a text file that contains n notes n_i, with i in $[0; n-1]$. Each note is characterized by a color and a timestamp when it is played. To transfer the beats in the FPGA, we will convert this text file into a binary format.
We built it inspired by domain specific binary format such as ILDA for laser shows [5].

The binary "stream" associated with a song is made of a header (metadata) and the notes. The header contains:

- the id of the song on 1 byte
- the number of notes in the song on 2 bytes. 2 bytes seem reasonable based on the number of beats we detect per song in matlab (roughly 400 so 256 = 1 byte is two low).

Following the header we have the notes. Each note is characterized by:

- its "color" on 1 byte. We take more than 3 bits (which is the minimal bound to represent the 5 colors of our guitar's button) since we might add new information later. We also go for it to remain consistent with the rest: every field is made of one or more bytes
- its timestamp on m bytes

We haven't yet determined m but to do it we will do some experiments. We will start by finding u integer such as $t/2^u < t_{min} < t/2^u - 1$ with t_{min} the minimum space in seconds between two following beats in the note list and t a constant which represents the maximum length of a song we permit in seconds. We will then chose m such as the multiple of 8 (a byte) : 8.m comes after u. If it happens not to be precise enough in practice, we will increase the value of m.

To compute the binary stream from the output of the Matlab algorithm:

- We set the song id and write it in one byte
- We discretise all the timestamps to the closer multiple of $t/2^3m$ and store these values.
- One it is done we know the number of notes, we write it in the binary stream in two bytes
- We write the notes one by one in the binary stream following the right format

4 Milestones

Milestone 1 (March 27th)

- We will buy the dummy guitar.
- We shall detect key inputs with the dummy guitar.
- We will play a song (raw sound format) from a SD card in the FPGA.
- We will develop a program to build a script of a given song. This means, to produce a file that contains the notes and their corresponding positions for this particular song.
- We shall finally make a prototype of the base game engine in Java.

Milestone 2 (April 10th)

- We will work on the sprites and study how to do graphics and how to encode the sound efficiently (how many bits, how many information we can store...).
- The Java game prototype will integrate the work on scripts from the first milestone.
- We shall have designed the game internal functioning to ensure a constant frame rate (on paper).
- We will have started implementing the game.

Milestone 3 (April 24th)

- We shall finalize the game.
- We will develop an algorithm to compute the score.
- We shall improve the performance of the game and work on the graphics.

Bibliography

[1] http://en.wikipedia.org/wiki/Guitar_Hero.

[2] http://i14.ebayimg.com/01/i/001/22/73/0b2412.JPG.

[3] http://labrosa.ee.columbia.edu/projects/coversongs/.

[4] http://www.ebay.com:80/itm/ws/eBayISAPI.dll?
ViewItem&item=220902361725&ssPageName=ADME:L:OU:US:1123.

[5] http://www.laserist.org/.