6.111 Final Project – Abstract
Jonathan Burnham, Christopher Hoffman, and Kevin Miu
October 29th, 2006

Introduction

This projects goal is to create a full motion dance machine. We hope that the main functionality of the dance machine will include the recording and processing of user motion and the comparison of the motions to predetermined dance moves.

The system will have four main components: video, audio, system control, and user interface. Video of a user will be displayed on a computer screen. The video components will detect colored swatches that the user wears and from that data, determine the location of various body parts. An algorithm will be developed to determine if the user is moving, and the relative location of the body parts over time. From this information, we can process the video to determine the user's motion and beat. The user position will be indicated on the screen with a stick figure, which will give the user general feedback about his or her location.

The audio component starts with building an understanding of the encoding of audio files and how to store them in memory. Obviously, due to memory constraints, any desirable length of music will require significant downsampling and a low bitrate. Because of the downsampling, the music will have to be upsampled on the front end of the audio player. Additionally, we will need to derive the beats from the audio signals. To do so, we will examine the relative power, or magnitude, of the song relative to a larger-time-scale local power. When the instantaneous power exceeds the local power, a beat can be detected. This processing will be coded. Eventually, the beats determined within the song will be coupled with the frequency of the dance moves required.

The system control component compares the beat from the user and the beat from the music. It first displays the beat determined from the music onto the screen so that the user can attempt to mimic this beat. The system will display the user's beat as he or she attempts to match the music beat, and compute the difference between the two. The system will also tell the user to do simple dance moves, such as moving the hand to certain parts of the screen. The system will compute an overall score depending on how well the user mimics the beat of the song and follows dance move directions.

Finally, a user interface component will allow the user to control which song is played and the type of song, display the user and song beats, and draw a stick figure representation of the user on the screen. The overall system will allow the user to experience a full motion dance experience.

Nominally, Jonathan will focus on the video component, Kevin on the audio component, and Chris on the system control and interface components. In actuality, we all want to get a feel for the entire project, so each member will work on modules within all of the components.

Video Component

The video component is responsible for taking images of the user and processing them to interpret the beat at which the user is moving, locate the position of certain key body parts, and draw a stick figure of the user onto the screen.

Image Capture and Conversion

The first part of the video component is getting the data from the color video camera issued from lab. The data from the camera is received over the composite video input on the FPGA. This data is in the YCrCb format, but in order to easily process the data, we need to convert this into RGB data to be able to pick out certain key colors that the user is wearing. We have already configured a module to convert this YCrCb data into 18 bit RGB data for processing. We have also set up a module to store and read the signal through ZBT in order to get a valid video signal that can be sent to the ADV7125 video digital-to-audio converter. The video signal is then displayed on the screen for viewing by the user.

Swatch Detection

In order to locate certain key body parts (hands, elbows, shoulders, etc.), certain brightly colored swatches will be worn by the user. After converting the video signal into RGB, the video component will process the frame and look for chunks of the image of a specific color. These chunks are the swatches worn by the user. The center of mass of each specific color will be computed. This should give us the screen x-y coordinate of the swatch, which can be used by the later modules.

Motion Calculation

To interpret the beat of the user, the video component will use the x-y positions of the swatches to calculate a velocity. This velocity can be computed by keeping track of the amount of change of the x-y position over a number of frames. The lower order bits of the position will not be used in the motion calculation as a means of reducing the effect of noise in the calculation. Sudden changes in the sign of the velocity can be interpreted as a beat. This beat will then be compared to the audio beat to determine how closely the user is matching the beat.

Drawing the Stick Figure

In order for the user to see how the system is interpreting its position, a stick figure of the user will be drawn on the screen. This will be performed by drawing lines between the certain x-y positions computed during swatch detection that correspond to the body parts found. This stick figure will be drawn on the screen, either by itself or overlaid with the original image.

Quadrant Detection

Finally, the video component will also detect if certain body parts are within certain "quadrants" of the screen. The user can be told to put certain body parts in certain quadrants via the user interface, and the system can use this information to define a virtual quadrant system and then determine the location of the various body parts within

the system. The quadrants will be drawn on the screen along with the stick figure, so the user is able to see how they are complying with the directions.

Audio Component
The audio component is responsible for storing, playing, and processing a given song.

WAV Processing
The first song to be used was "Yeah!" by Usher, and featuring Lil John. Audio conversion software was used to translate the MP3 to a 8-bit, 16kHz .wav file. The .wav file was opened in MATLAB, and an interesting part of the song was selected to be used. Because the song was stereo, the two lines of the song were averaged. We chose to minimize the use of memory and sacrifice the stereo sound. From there, the .wav values, which ranged from -1 to 1 were scaled to 0 to 255. From there, the numbers were converted to an 8 bit two's complement. The values of the song that were greater than or equal to 128 had 128 subtracted from them, as they would be the positive values in a two's complement. The values smaller than 128 were subtracted from 255 and then 1 was added, to make them the equivalent of a negative value in two's complement. These two's complement values were printed in decimal base and comma separated.

ROM
A ROM large enough to hold the song was created. The ROM used a radix of 10 since the .wav conversion was done in decimal. Then all the values of the .wav conversion were copied into the ROM. A program will be created to store the data in the ROM into the Flash memory integrated into the labkit. This should only have to be once, since the data stored in the Flash is stored until it's erased. Once the data is stored, the actual project file can be written onto the FPGA and the song data stored in the Flash can be used.

Upsampling/Interpolation
Because the AC97 plays at a 48kHz sample rate and the song was originally 16kHz, the song is upsampled by a factor of three. Interpolation will also be done as part of the upsampling in order to improve the quality of the song.

Audio Playback
The audio playback will function completely out of the AC97, as it did during Lab 4. The data will be fed to the AC97 and played. Volume and start/stop capabilities will be added.

Beat Detection
The beat of the song also must be calculated for the dancing function. The beat is calculated by looking at the instantaneous energy of a location in the song relative to the local energy around that point. If the local energy exceeds the instantaneous energy, then a beat is detected. The intervals at which these beats occur will be recorded and stored in a RAM. This processing will be done before the user dances to the song in order to accurately calculate the beat. Initial tests were run with MATLAB to ensure that the beat processing was working as expected. The algorithm will later be implemented on the FPGA.

System Control Component
The system control component is responsible for comparing the beats from user and from the audio and calculating a score and controlling song playback.

Beat Comparison
The system will take the beat outputs from the motion detection module of the video component and the audio processing module and compare them. The closer the user's beat is to the beat detected by the audio beat detection, the higher score they get. Additionally, the location of the body part is also factored into the score. This can be tested by just having the beat inputs as a predetermined sequence and have the user beat input from a switch. The score can be analyzed to see if it is as expected.

Song Playback
The system will control the audio component's playback mechanism and song selection. It should keep track of the starting and ending memory locations of each song and the types of songs. This data will be used by the user interface and the audio component.

User Interface Component
The user interface component is responsible for creating a medium for which the user can input and interpret the output of the dance machine.

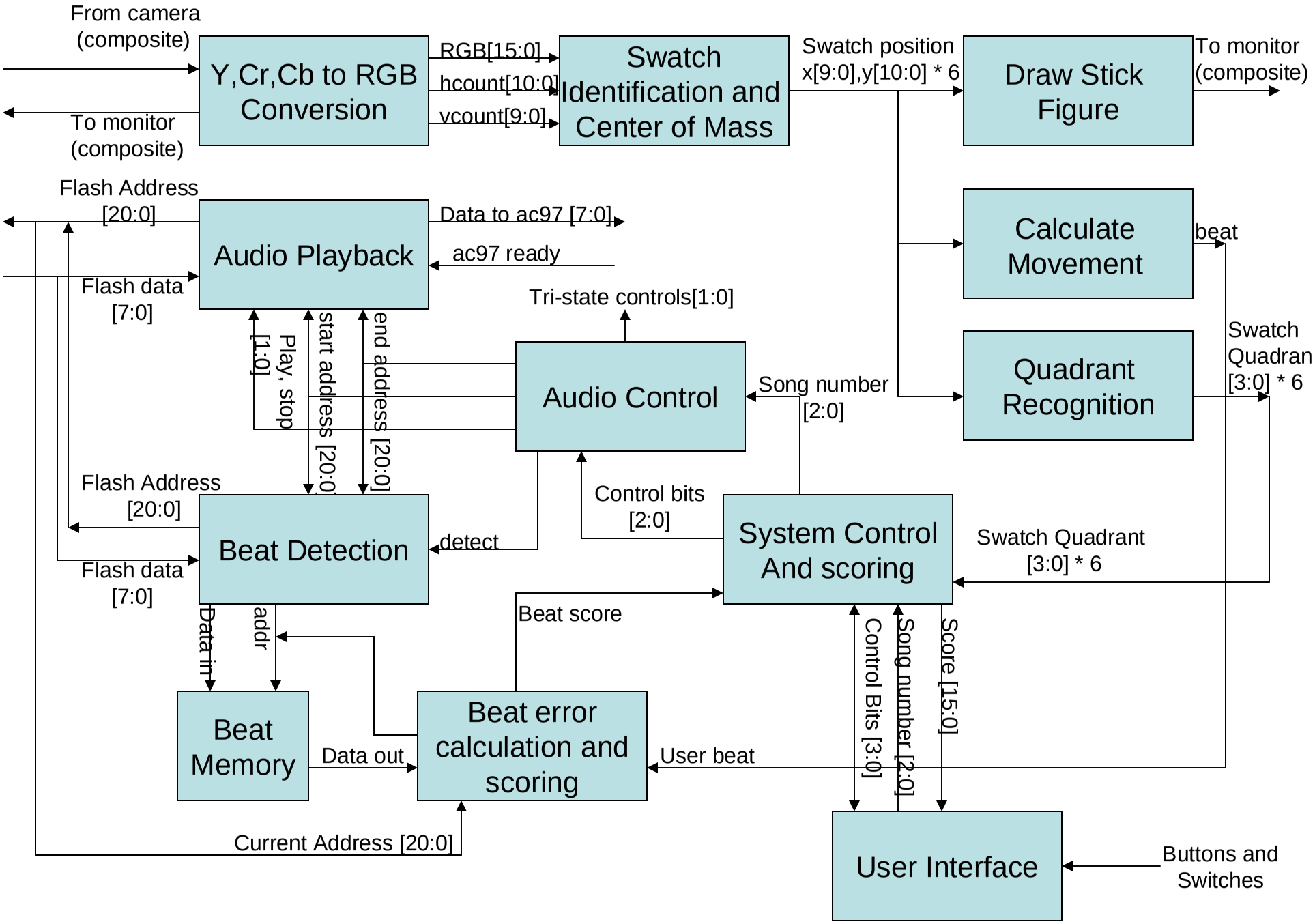
Song Selection and Volume
The user will be able to perform song selection and modify the volume of the system via the buttons on the labkit. The user will also be able to reset the game via a button press. The buttons will be debounced.

Command Determination
When a given song is stored in the system, the song should have a choreography original to it. We plan to do this either by manually choreographing all the songs to be used, or to choreograph a single song with particular moves based on the song type, beat, and note, and then to apply that algorithm to other songs. The required location of various parts of the body will be determined and this information will both be displayed and used in scoring.

Command Display
The location of the commands will be displayed on the screen relative to stationary parts of a cartoon mockup of a human body. For example, the chest area and trunk may appear as a stationary object, which the shoulders and the hips move to relative locations described by a grid on the screen.

Beat Display
A visual cue will be given to help the user adjust to the beat of the song. This may be done by switching the display of two differently sized images stored in the ROM. Later beats in the song may also have to be shown if the song beat changes, in order to give advance notice to the user that the dance frequency will change.

Score Display
The user should be given visual feedback on the dancing accuracy. This can be done at the most basic level by storing images of various numbers in the ROM and then displaying those numbers in a sensible way to represent the score.


From camera (composite)
Y,Cr,Cb to RGB Conversion
To monitor (composite)
RGB[15:0]
hcount[10:0]
vcount[9:0]
Swatch Identification and Center of Mass
Swatch position x[9:0],y[10:0] * 6
Draw Stick Figure
To monitor (composite)
Flash Address [20:0]
Audio Playback
Data to ac97 [7:0]
ac97 ready
Flash data [7:0]
Calculate Movement
beat
Tri-state controls[1:0]
Play, stop [1:0]
start address [20:0]
end address [20:0]
Audio Control
Song number [2:0]
Quadrant Recognition
Swatch Quadran [3:0] * 6
Flash Address [20:0]
Beat Detection
detect
Control bits [2:0]
System Control And scoring
Swatch Quadrant [3:0] * 6
Flash data [7:0]
Data in
addr
Beat score
Control Bits [3:0]
Song number [2:0]
Score [15:0]
Beat Memory
Data out
Beat error calculation and scoring
User beat
Current Address [20:0]
User Interface
Buttons and Switches