DIGITAL TUNER: PROJECT PROPOSAL

Roshni Cooper, Linda Fang

DESCRIPTION

This project will implement a digital tuner. It has three major functions – record, playback and play a tone. The functions will be selected via the buttons on the FPGA. Record will take a user microphone input and store it in memory. Playback will playback the last recorded tone and display its position on a music staff, its frequency and its note number (the numeric value corresponding to the key of the tone on a piano). Play tone will take one of three user inputs (button/switches, mouse, keyboard). It will play the corresponding tone and display its position on a music staff, its frequency and its note number.

PROJECT INPUT/OUTPUTS

Input: microphone, switches[6:0], buttons[2:0], mouse, keyboard
Output: speakers, video

microphone	Take an audio input, which must be a single tone (little noise should be OK).
switches[6:0]	Take a binary input that corresponds to the note number.
button[0] (record)	When pressed, record. (overrides all other inputs)
button[1] (playback)	When pressed, play back last recorded tone. (overrides all but record)
button[2] (playtone)	When pressed, play the tone specified by the switches. (overrides all but the button[1:0])
mouse	The user can click on the music staff (on the screen), and it will play the tone selected.
keyboard	A numeric input will play the tone specified (corresponding to the note number). A letter input (A-G) will play the corresponding note on middle C and the octave above.
speakers	Outputs sound when requested.
Video	When initialized, will display a treble and a bass staff. Whenever a tone is recorded, played back, or requested, it will display the corresponding note on the staff, the note number, and the frequency.

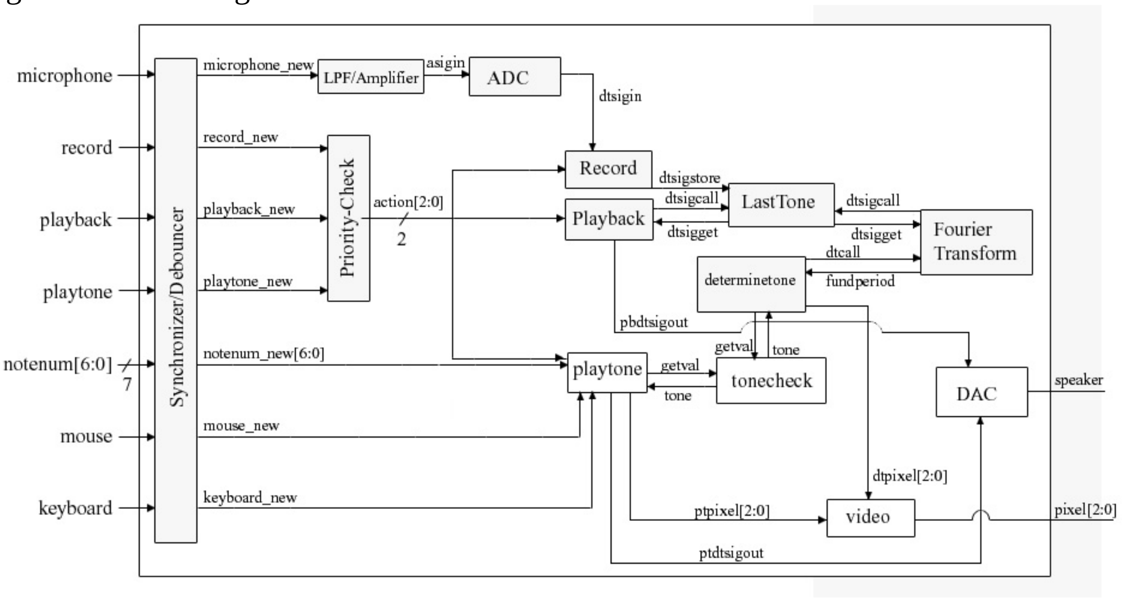
TESTING AND DEBUGGING

Each separate module will first be tested virtually using the testbenches in Verilog. Once each module is functioning correctly individually, the entire project can be tested using testbenches.

In order to ensure that the actual system is working, we will test that each component, and finally the entire system, is working by inputting pure tones (signals) generated by Matlab into the microphone and checking that the video outputs the correct frequency and note. We can also measure the frequency of the tone inputted using the logic analyzer, and checking to make sure that our subsystems are providing the correct frequency and tone.

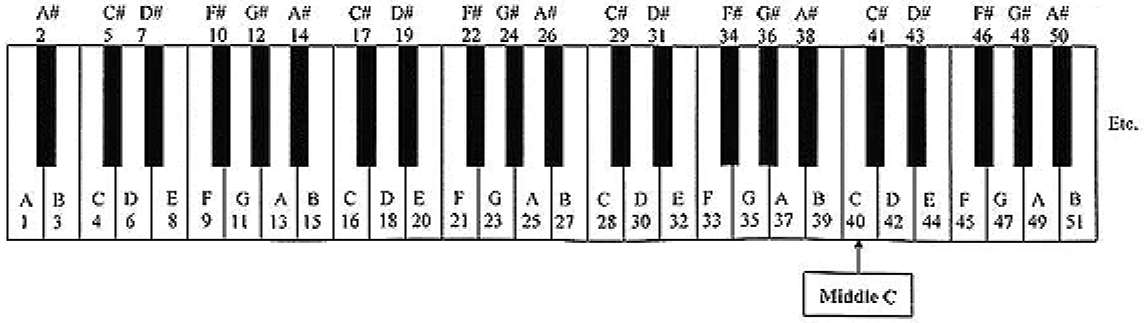
FIGURES

Figure 1: Block Diagram



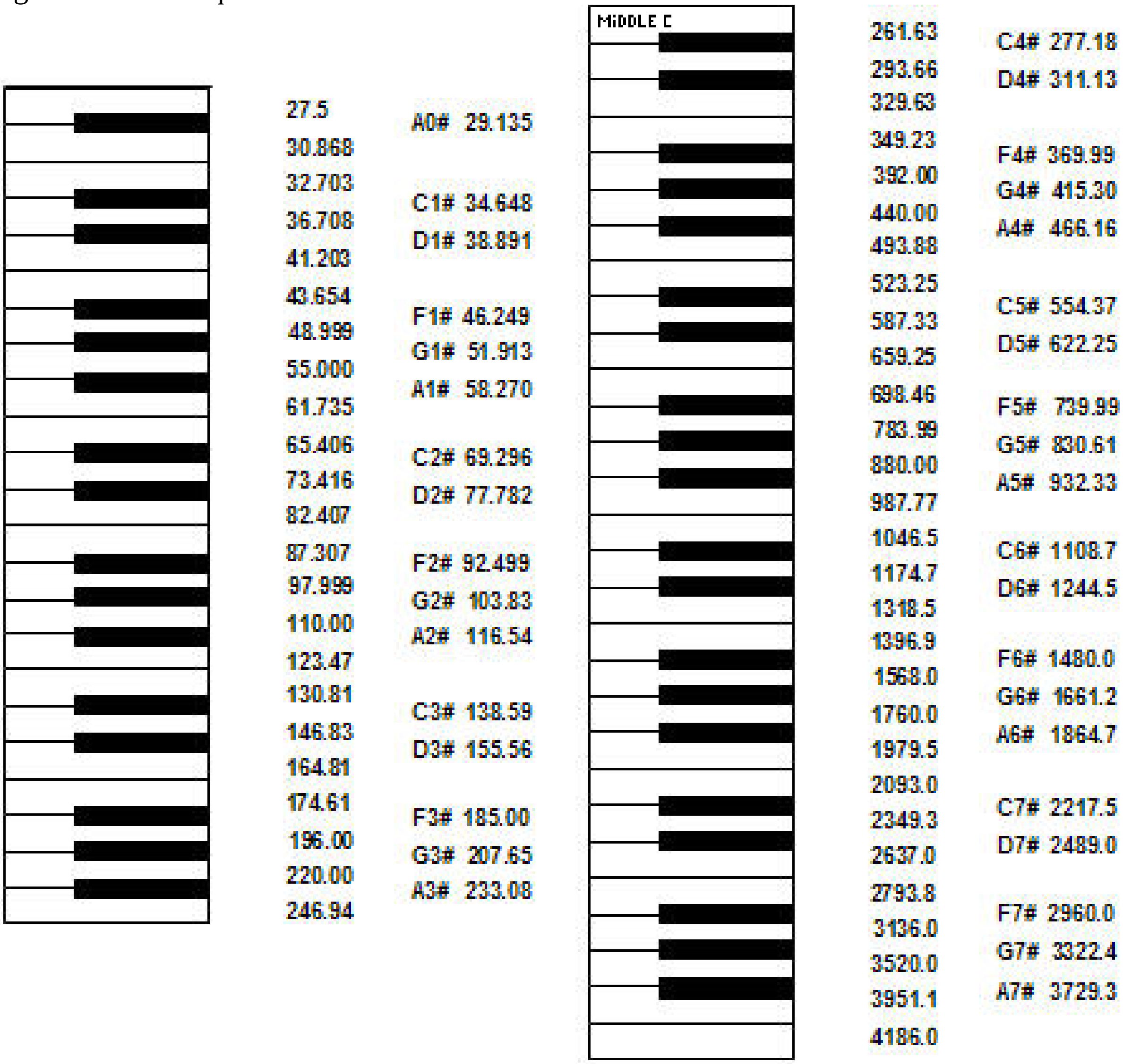
This is the block diagram that we will be implementing.

Figure 2: Keyboard Numbers



A typical piano has 88 keys (black and white), numbered lowest to highest frequency from 1 to 88. These numbers (note numbers) will be used to reference each tone.

Figure 3: Tone frequencies



The keys on a piano range from 27.5Hz to 4.186KHz. An increase by one octave corresponds to doubling the fundamental frequency.

MODULE DESCRIPTIONS

Subsystem 1: Start

Synchronizer/Debouncer
This module will synchronize all the inputs at the same clock edge.

Priority Check (PC)
This module will take the record, playback and/or playtone inputs and call for the appropriate subsystem to start. If record is called, neither playback nor playtone will run,

even if they are called on the same clock edge. If playback is called, playtone will not run.

Subsystem 2: Sound

LPF/Amplifier
This module will take in the input from the microphone and put it through a low-pass filter to get rid of extra noise and an amplifier to expand the amplitude of the signal. This will make it easier to work with and take accurate samples from.

AD Converter (ADC)
This module will take as input the output from the LPF/Amplifier module, and will convert the analog signal into a digital one.

DA Converter (DAC)
This module will take a discrete audio signal and convert it to an analog signal that can be played by the speakers.

Subsystem 3: Memory

Last Tone (LT)
This module stores the recorded tone in a RAM.

Tone Check (TC)
This module will have all 88 possible tones pre-stored in a ROM. Their addresses correspond to their note numbers.

Subsystem 4: Record

Fourier Transform (FT)
This module will call the RAM, and takes its output data as inputs. It will calculate the Fourier transform of the signal and find its fundamental period.

Determine Tone (DT)
This module will call the FT module to find the fundamental period of the last recorded tone. It will find the frequency of the tone, which it will send to the ROM and get back the corresponding piano number, which it will then output to video.

Record
This module takes the ADC output as input and stores it into a RAM (it will effectively clear the RAM, then store the new data in).

Subsystem 5: Playback

Playback

This module will call for the DT module to start. It will also access the RAM and output its data to the speakers.

Subsystem 6: Playtone

Playtone

This module has three possible user inputs. The primary input is the binary input from the switches and button[2]. If button[2] is not pressed, then the second priority input is the mouse, and the final priority input is the keyboard. The module will access the ROM with the address value specified by the input, and output the data of the selected address to the speakers.

Subsystem 7: Video

Video

This module will take inputs from the DT module and from the Playtone module and display the correct note on a staff on the screen as well as its corresponding note number and frequency.

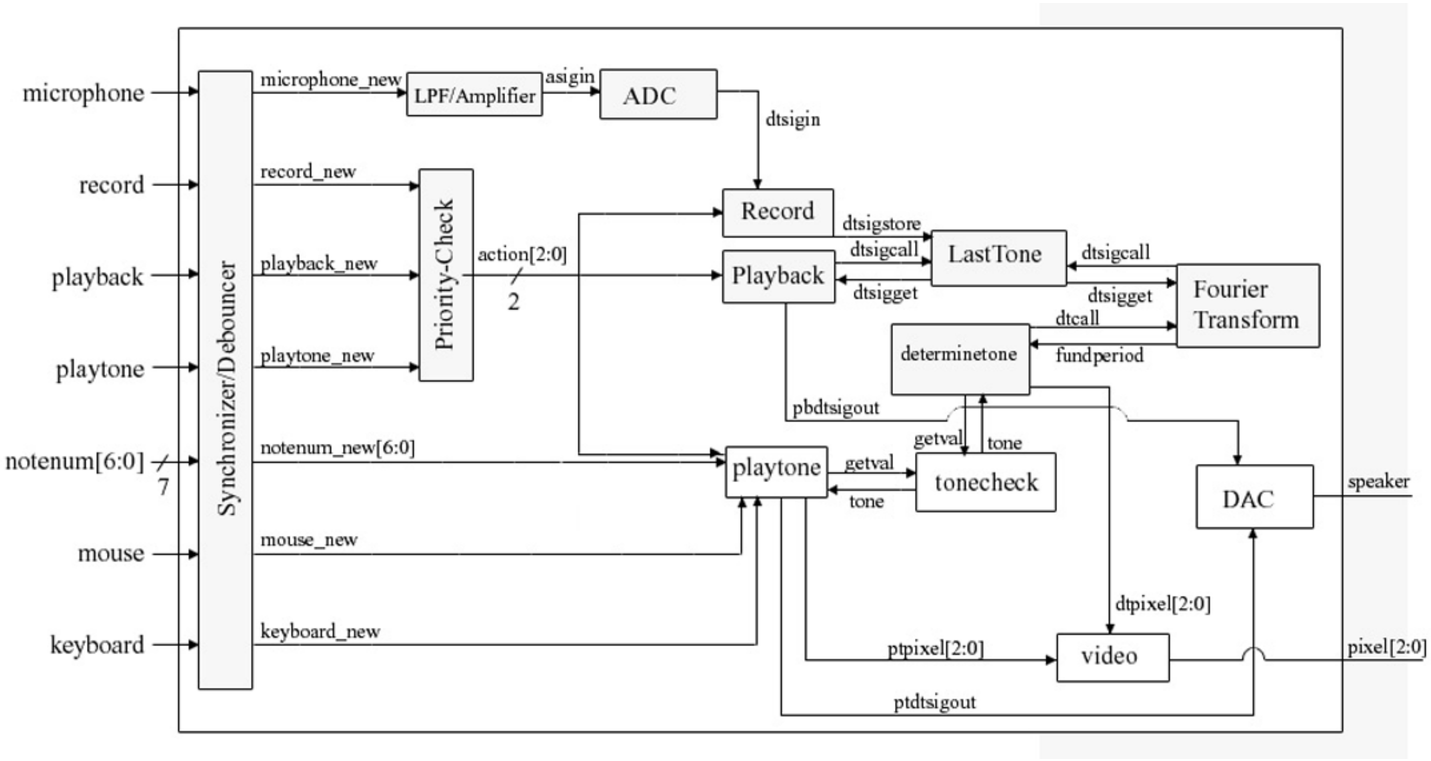
PROJECT RESPONSIBILITIES

Linda

DT module, FT module, Playback module, Record module, RAM
Linda is doing the top half of the block diagram (subsystems 3,4,5), except for the LPF amplifier and AD converter.

Roshni

Video Subsystem, ADC, DAC, LPF/Amplifier, Playtone module, ROM
Roshni is doing the bottom half of the block diagram (subsystems 1,2,3,6,7), along with the LPF amplifier and AD converter.


microphone
record
playback
playtone
notenum[6:0]
7
mouse
keyboard
Synchronizer/Debouncer
microphone_new
record_new
playback_new
playtone_new
notenum_new[6:0]
mouse_new
keyboard_new
LPF/Amplifier
asigin
ADC
dtsigin
Priority-Check
action[2:0]
2
Record
Playback
playtone
dtsigstore
dtsigcall
dtsigget
LastTone
dtsigcall
dtsigget
Fourier Transform
dtcall
fundperiod
determinetone
pbdtsigout
getval
tone
getval
tone
tonecheck
DAC
speaker
dtpixel[2:0]
ptpixel[2:0]
video
pixel[2:0]
ptdtsigout