Sound Module:

Note Detector – The note detector should use an FFT to decode which note the user is playing, and output the information in a way the rest of the game can understand. The final output is four three bit numbers which represent which note the user is playing on each string:

Things to demonstrate.

1. FFT / note_storage . Demonstrate working FFT Module. On logic analyzer display the bin number and magnitude square of the Fourier Transform.
2. Note Logic – Demonstrate the note logic keeping a running max of the desired sets of bins over one full sweep of the FFT. Display on the logic analyzer the current max for every set of bins, observe as it changes
3. Note Logic – Demonstrate the note logic does not keep values that are blow a certain threshold. Note on the logic analyzer how the values of the output are zero when no notes are played.
4. Demonstrate working note detector by turning on the logic analyzer, playing a set of notes, and seeing those fret numbers output by the note detector.
5. If time allows, demonstrate enough robustness and speed that you can play lots of notes in a row, and the module will not get confused.

Song Module – The song module is responsible for playing a song from the flash memory to the AC97 chip. Every time the AC97 chip asserts a ready signal it must deliver the next piece of PCM data from the Flash Memory.

Things to demonstrate:

1. Output the PMCM data to the logic analyzer, note that the next piece of PCM data was put in the flash memory becomes the output every time the ready signal is enabled. The module should repeat each 8-bit PCM word 8 times, since the song will be down sampled by that factor.
    1. If time allows, this module should perform 8 step linear interpolation over successive samples.
2. Working module, load a song onto the AC97 chip, put on the headphones and listen to a song being played.

Song Storage – The song storage module is what will allow us to store a full song the lab kit. It must be able to take a song as a coe file with PCM data in it. Initialize BRAM to those values, and write all the data to the Flash.

Things to demonstrate:

1. Ability to take a file (binary PCM data) and convert it to a .coe file that can be loaded onto the BRAM. (this part is done with software, but its still necessary, so it is included as a check item)
2. Demonstrate the contents of a .coe file are in fact written to the FLASH memory by outputting the contents of the Flash memory to the logic analyzer, and observing that they match the contents of the .coe file.
3. Demonstrate working module by playing a full song off the flash!

FSM – The FSM is responsible for keeping track of the state of the game and switching between states. States of the game are Title Screen, Game Over, Play Mode, Pause. It needs to output which state is currently in as a two bit number (mapping to the order given above), and an enable that is high in state

1 and low in all other states. It must stitch between states when the proper control signals are asserted.

Things to demonstrate:

1. On logic analyzer, display the current state and the enable bit. Observe the following behavior:

State	***Next State***	***Pause/Button***	***Reset***
0	0	0	0
0	0	0	1
0	1	1	0
0	0	1	1
1	1	0	0
1	0	0	1
1	3	1	0
1	0	1	1
2	2	0	0
2	0	0	1
2	0	1	0
2	0	1	1
3	3	0	0
3	0	0	1
3	1	1	0
3	0	1	1

Saving the song template – Template Module: holds the actual notes of the song.

- On the logic analyzer: see that the BRAM holds the notes to the song (stored as an 8 bit number, the first 5 representing the note, the next 3 representing the duration)
- On logic analyzer: see that the output from the BRAM is decoded correctly into a 12 bit number representing the note of the
- On logic analyzer: see that the output from the BRAM was held for the proper amount of next_note cycles according to the table in the project report.
- On logic analyzer: see that song starts from beginning on reset, and only changes to next_note input when enable is high.

Video Display – Video Module – Takes input from the template module, the FSM, and the Game Logic module and displays the game. Also tells Game Logic module when the user should be playing a note and what score they will get for playing it.

- Video module outputs a next_note signal to the template module every 8th note. On a song at 120bpm that is every .25 seconds

- (If time permits): have an adjustable speed setting to game. Use switches to set a speed that the game can be played at, between 50% and 150% its original speed.
- On monitor: Display different colored sprites for notes. See a sprite exists on screen with the right x, y pixel location and color according to the input to the sprite.
- On monitor: storing future notes/displaying future sprites. See that future sprites scroll across the screen at the same rate as the song(i.e. cross the "play now" marker at the on beat with the song)
- On monitor: displaying the score. Given a certain input score from the game logic, the video module displays that score on screen.
- Using logic analyzer: note point value: the video module outputs a score(3 bits) that increases as the sprite approaches the "play now" marker and decreases as the sprite moves away from the "play now" marker (the output score will be delayed a small amount of time to account for the delay in the circuitry for accurate scoring).
- Using logic analyzer: see that the note output by the video module, note_expected, is the actual note that the user should be playing.
- (if time permits) display how well the user is playing (eg. Good, bad, perfect), dictated by the game logic module.
- On monitor: see that fret board and "play now" marker.
- On monitor: see title screen, play screen, pause screen, and game over screen, with the game_mode input from the FSM(2 bits).

<u>Game Logic</u> – Game logic module – compares the values from the Note Detector module and the video module and keeps track of the total score. Adds score from video module if the two signals are equal. only adds the first score that is equal(if any are equal) and does not add scores after that until the next eighth note (given by video module)

- Logic analyzer: module keeps track of the accumulated score.
- Logic analyzer: module adds input note point value from video module to the total score if the two input note values are the same.
- Logic analyzer: module only adds the score input on the first clock cycle when the two input note values are equal (every eight note).