Group #37 Proposal Presentation

Taleb Hirani, Vincent Chow, Liam Cooper, AJ Dhillon, Jonathan Pegues

Problem Statement:

Our team has received a request to design and implement a SCOMP peripheral that can generate continuous 50% duty cycle square waves within 2% accuracy of a range of frequencies.

Our Team’s Proposal

To design a peripheral that plays notes for programmed songs and allows the user to record and playback a series of notes.

DE10-Lite Implementation - Switches

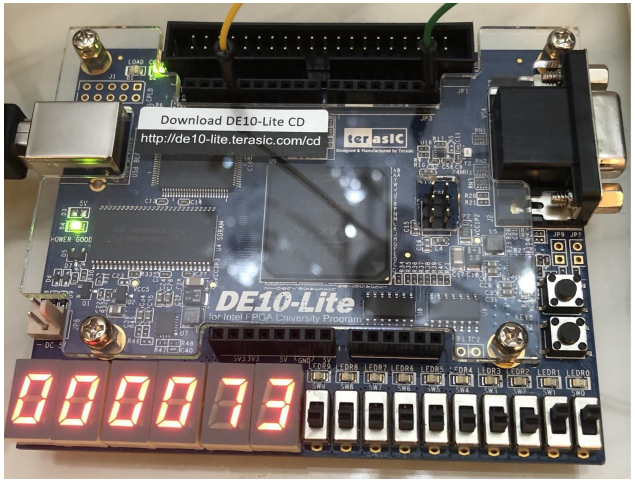
- The switches will be used to record notes the user wants and toggle between playback and recording mode
- Switches 0-5 will be used to represent a binary number from 0 to 61, and this value will map to a unique note to be played
- The remaining switches will enable playback of recorded notes or a preprogrammed song

DE10-Lite Implementation - 7-seg display

- Use of the 7-seg display to display the frequency that is currently being played
- Allows user to know what note is being recorded before saving it to their song
- Input to switches will map to a frequency that will be outputted to the 7-segment display

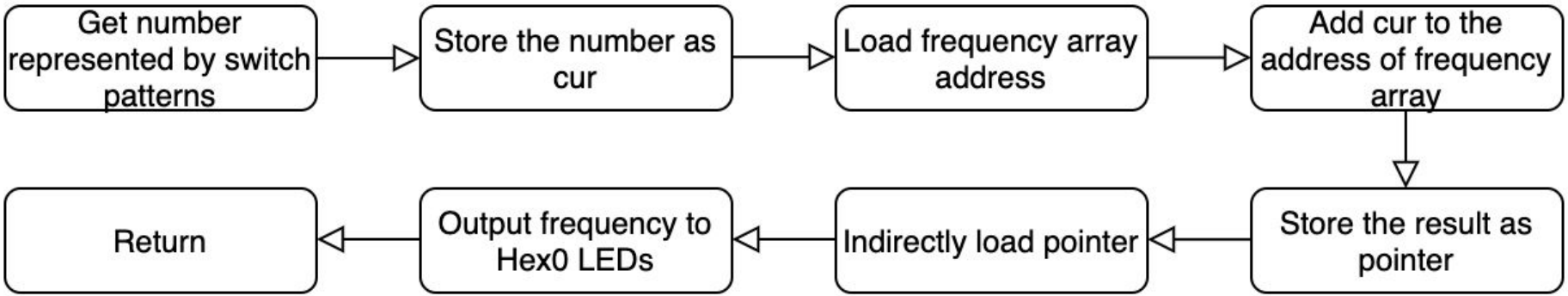
Displaying Frequencies

- Displays frequency of note being played on Hex0 LEDs.



Displaying Frequencies - Approach

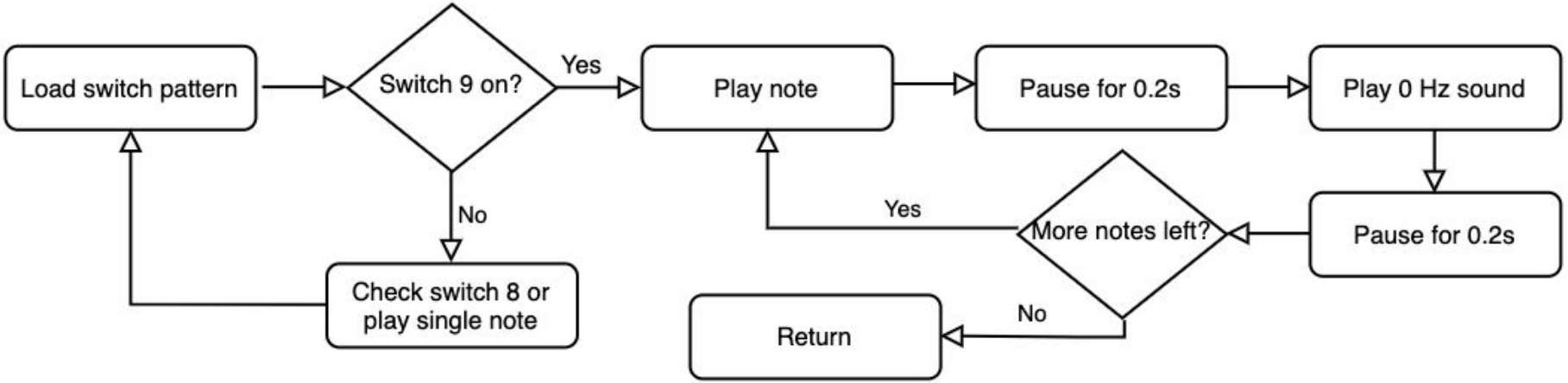
- Uses array to store frequencies of notes and uses indirect addressing for access.



Programmed Song

- Program songs using different frequencies
- Played when switch 9 is switched
- Twinkle Twinkle Little Star

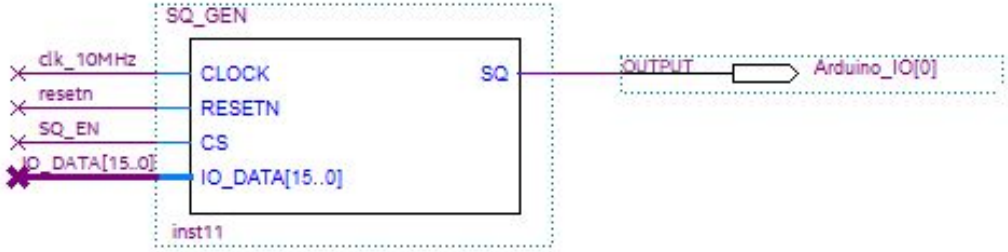
Programmed Song - Approach



Design Specifications

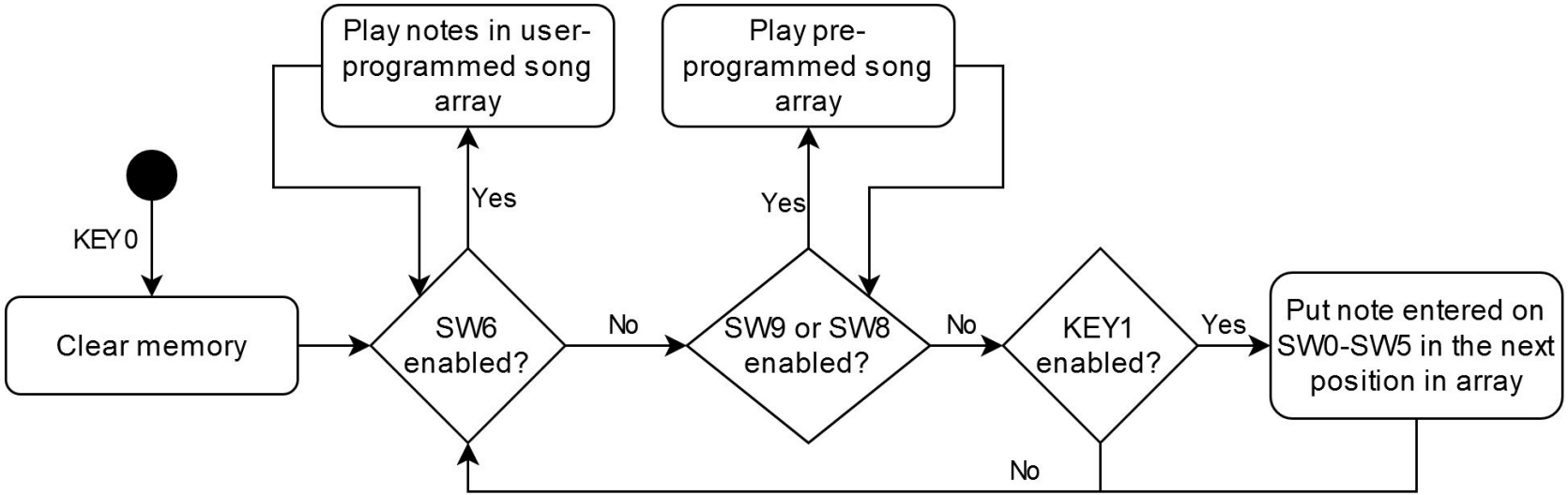
- Each note within 2% of their true frequency
- Notes should play every 0.2 seconds
- Storage of at least 64 notes
- At least 2 sample songs readily playable
- Display frequency being played on 7-segment displays

Peripheral Approach



- 6-bit binary number input to IO_DATA selects note
- 10MHz clock
- IO_DATA converted to index in a frequency array
- Counter incremented to 5MHz divided by the chosen frequency minus 1, then counter is reset and the output goes from high to low or low to high

Software Approach



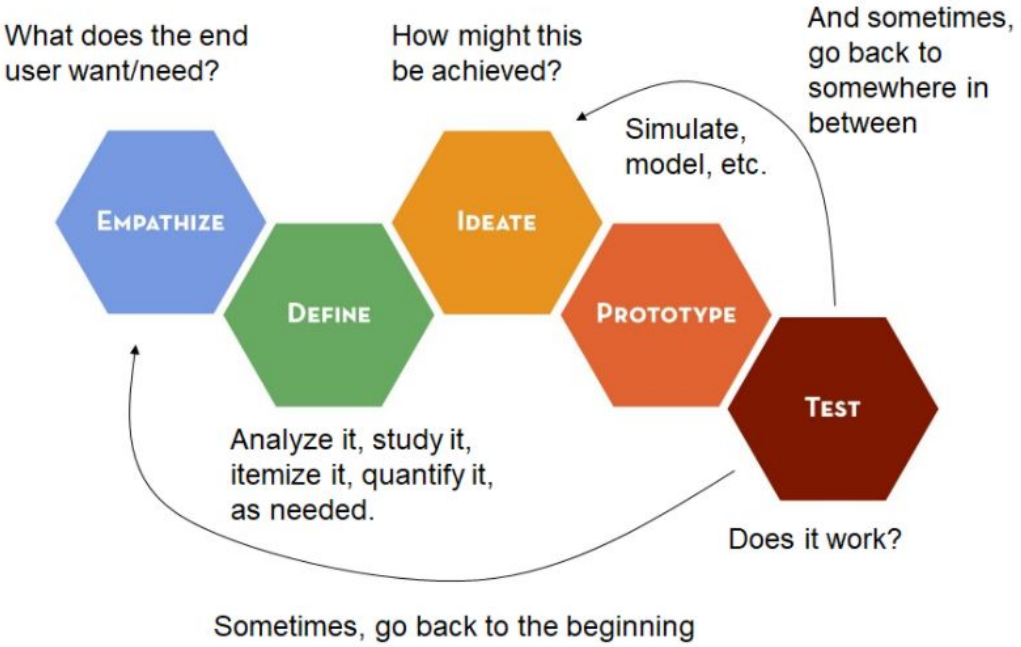
Testing and Verification

- Highest frequency played = 2093Hz
- 5MHz / 2093 - 1 = 2387
- Peripheral counts to 2387 before switching
- 5MHz / 2387 = 2094
- % error between 2093 and 2094 = 0.048%
- Tested experimentally with frequency detecting phone app

Demonstration Plan

- Play pre-programmed songs
- Demo programming a song, playing back the song, and resetting the system
- Use frequency detecting app to compare what is displayed versus what the speaker is playing

Design Thinking Process



Project Timeline

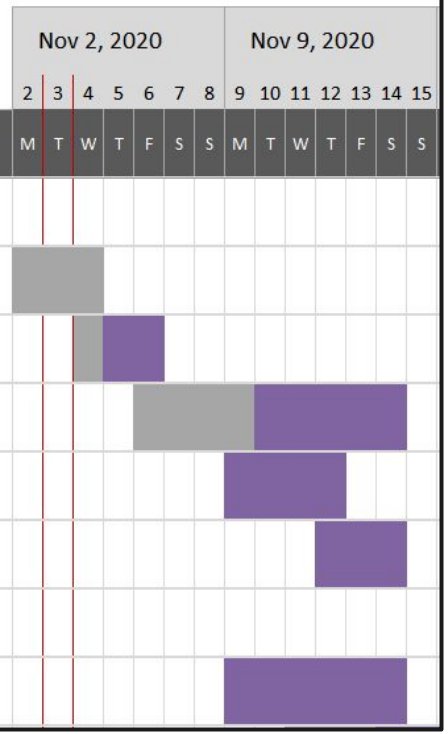
Music Player and Recorder

Project Team #37

Project Start: Sun, 11/1/2020

Display Week: 1

TASK	ASSIGNED TO	PROGRESS	START	END
Project Design Tasks				
Rehearse Proposal	Full Team	100%	11/1/20	11/4/20
Program Another Song	Full Team	35%	11/4/20	11/6/20
Implement Recording	Vincent, Liam, Taleb	50%	11/6/20	11/14/20
Prepare Demo	AJ and Johnathan	0%	11/9/20	11/12/20
Rehearse Demo	Full Team	0%	11/12/20	11/14/20
Troubleshooting				
Debug/Find Design Flaw	Full Team	0%	11/9/20	11/14/20



Conclusion - Main Goals

- Create a peripheral to generate continuous square waves
  - This has already been done, we are currently working on an efficient way to store pre recorded and live recorded songs in assembly
- Create assembly code so a user can record notes and play them back on the peripheral we create

Conclusion - Upcoming Deadlines

- 11/6 Add a few more songs
  - By having a larger sample size of songs, we can be sure that our peripheral is working
- 11/14 Implement recording and playback feature
  - Allotted a large period of time to this portion to ensure enough time for debugging if issues are found
- 11/15 Have a finished product and begin rehearsing for demo