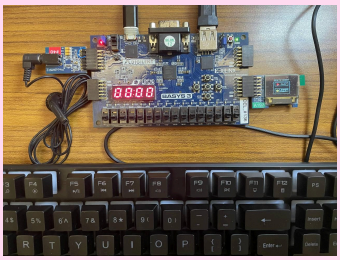
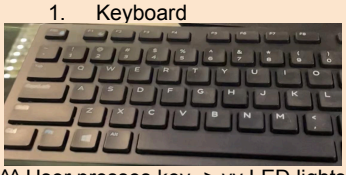
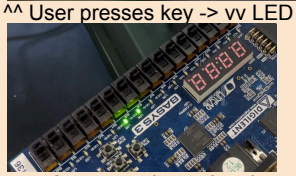
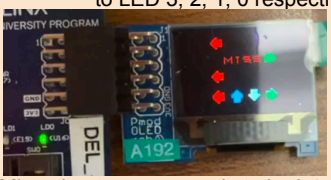
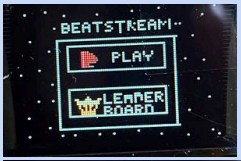
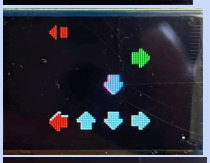
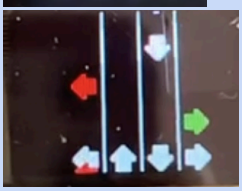
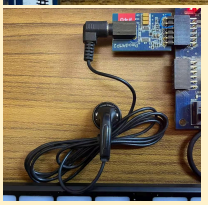
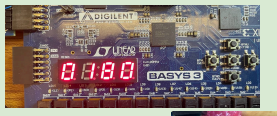
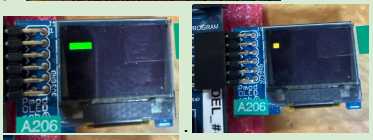
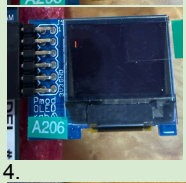
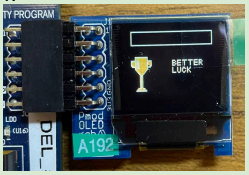
Team Members: Yoshizaki Eichi Lance, Kim Gyurim, Zhong Shu, Saravanan Swetha

PERSONAL AND TEAM IMPROVEMENTS		
Student and Improvement Name	**Improvement Description**	**Images/Photos**
Team "Beatstream"	This project is focused on delivering a piano tiles -esque rhythm game experience. Gameplay is featured around pressing the arrow keys corresponding to individual notes at the intended timing when the notes reach the collection area at the bottom of the screen, signified by the white arrows. The user's score for each note individually will be put into 4 categories: missed (user does not press after 200 ms after the intended timing), bad (between 100 and 150 ms before/after the intended timing), good (within 50 and 100 ms), and perfect (within 50 ms). There are two modes (we may refer to as maps) of difficulty, easy and hard, with the hard mode being designed for the notes to move faster down the screen. The user can also navigate through different screens by using keyboard and push button input.	
Student A: Yoshizaki Eichi Lance Improvement: Game Logic: Mapping scheme for easy map creation, Note grading, Cheating Prevention, Screen navigation	1. *Keyboard Input* **Inputs**: (PS2Clk, PS2Data) letters, space, enter, arrow, lshift keys The first task was to implement the keyboard input and associate key presses to LED **output**s. This module was then used for generating input signals to other modules. When the key is pressed, the corresponding signal will be set to 1 (LED ON) and 0 (LED OFF) otherwise. 2. *Note Grading* **Inputs**: up, down, left, right keys and notes map Given a unified notes map scheme, the student coded a module that automises note grading based on any given notes map and **outputs** the *perfect*, *good*, *bad*, and *miss* signals. These are 4 bit values (1 bit per lane), and each bit gets a short pulse when a note is deemed to be in that grading category. Additionally, **output** *p, g, b, m* signals represent the signal when any of the 4 lanes deems a note as that category (perfect, good, bad, miss). 3. Prevent 'cheating' **Inputs**: key press status **Output**: pulse signal representing shortened key press A Clever user may think of 'beating the system' by holding the arrow keys for a prolonged period and gaining score easily. To mark these as missed notes, a shortened key press signal is created so that the user only hits the note when the positive edge of the key press aligns with the note. 4. *Screen Selector* **Inputs**: same as keyboard input from part 1, as well as an endgame and pause flag **Output**: Selected screen number The student also implemented **reset** and **pause** functionality for the gameplay portion given keyboard input from the user. **Instructions for Game Navigation**: From the main menu, user presses up key to navigate to the difficulty menu, then selects up or down key to choose which map to play. The user may press the down key in the main menu to navigate to the loading/icon screen. From the game screen, user presses p key to bring up the pause menu. From pause menu, if user press left arrow key he/she will be brought back to the main menu (quit option), if user presses enter key the game will navigate back to difficulty menu (retry option), if user presses left or p key the game will resume. The end screen displays at the end of gameplay automatically. Modules: keypress.v, PS2Receiver.v, key_status.v, elongate.v, short_press.v, play_timer.v, note_grader.v, decomposer.v, screen_selector.v	1. Keyboard  ^^ User presses key -> vv LED lights up  2. p, g, b, m signals correspond to LED 3, 2, 1, 0 respectively  Missed note corresponds to both 'miss' display on-screen as well as LED 0.

Team Members: Yoshizaki Eichi Lance, Kim Gyurim, Zhong Shu, Saravanan Swetha

Student B: Kim Gyurim Improvement: Game Display (Easy, Hard), Menu Screen, Pause Screen, Difficulty Menu	**1. Menu Screen** **Inputs:** CLOCK, btnC, btnD **Outputs:** JC When the bitstream is first run, the game menu pops up on the oled display. The user is given two options to choose from, the top being “Play” which leads to the game difficulty screen, while the bottom option leaderboard is the option to navigate to the endscreen/leaderboard screen. The user is prompted to press up key on the keyboard, to navigate to the menu difficulty page to play the game. **2. Difficulty Menu** **Inputs:** CLOCK **Outputs:** JC After pressing play, before the game starts, the user is given a selection of the two game modes, easy and hard. It is displayed on the OLED board. When the user presses ‘up’ key, it prompts the user to the **‘easy’** game mode, when the user presses ‘down’ key, it prompts the user to the **‘hard’** game mode. **3. Game/Easy Hard Map** **Inputs:** CLOCK, reset, pause, perfect, good, bad, miss, missedflag **Outputs:** JC A rhythm game where arrows spawn from the top to the stationary arrows at the bottom. Arrows either go **off** screen or **disappears** on the white part depending on if the note is **missed**. Compared to the ‘easy’ map, ‘hard’ map has **more** arrows and the falling arrow speed is **faster**. **4. Pause Screen** **Inputs:** CLOCK **Outputs:** JC In game screen, when the user presses p, the **pause** screen **pops** up, and the user is given three options, **‘quit’, ‘retry’, and ‘cont’**. If the user presses, left arrow key, the user gets sent to the difficulty menu. If the user presses, enter key, it **restarts** the game and the game resets from the beginning. If the user presses the right key, the game **unpauses** and the arrows continue from where it was left off.	     
Student C: Zhong Shu Improvement: On-screen Feedback Display, Background Music	**1. On-screen Feedback Display** **Inputs:** note-grading signals/keyboard inputs **Outputs:** JC ports When a note is hit or missed, visual feedback that displays "Perfect", "Good", "Bad" or "Miss" in different colours above the notes is given at the centre of the screen, according to the note-grading signals, while ensuring that the visual feedback is immediate and clear to the player. **2. Background Music** **Inputs:** - **Outputs:** audio output Using Pmod AMP2 and external earpiece to play the background music in the game. The music lasts for 1 minute in ‘Easy’ mode and 30 seconds in ‘Hard’ mode. Players can pause the music if they need a break, seamlessly continue from where they left off, or reset the music to the beginning if they restart the game.	 

Team Members: Yoshizaki Eichi Lance, Kim Gyurim, Zhong Shu, Saravanan Swetha

Student D:Saravanan Swetha Improvement: Loading game screen, display points system, survival bar, end game screen/leaderboard	1. **Loading game screen:** **Inputs:** CLOCK, keyboard signals **Outputs:** JC When it enters the game page, **"Enter name 1"** is prompted on the screen where the player uses the keyboard to type in the name. The keyboard signals are then converted to display letters in the OLED. When the user clicks **'enter'**, there are 6 display icons shown on the OLED and prompting the user to select one of them. The pushbuttons are used to replicate the selection process of a keyboard, for example: if btnR is pressed only the heart icon is displayed and if btnL is pressed only the tree icon is displayed so on and so forth. 2. **Display points system:** **Inputs:** CLOCK, keyboard signals: btnD(perfect),btnR(good),btnL(bad),btnU(miss) **Output:** anode, segments The maximum points that the user can attain is 9999. 'Perfect' note: 50 points (btnD) 'Good' note: 30 points (btnR) 'Bad' note: 10 points (btnL) 'Miss' note: 0 points (btnU) In the actual game, the notes are classified based on the timing the user takes to hit the note. For my part I used the push buttons on the basys3 board to resemble the notes. The total points increment based on the respective push buttons pressed and are displayed in the 7-segment display of the board. 3. **Survival bar:** **Inputs:** CLOCK,keyboard signals: btnC(miss) **Output:** JC The health level of the survival bar starts from 100 and decrements every time the user **'misses**' the note. For my individual part btnC corresponds to **'miss'** note. If the health level is below 40, the bar turns **orange** and if the health level is below 20, the bar turns **red** signifying to the player. Once the survival bar hits 0, the game ends and the pause screen is displayed. 4. **End game screen/leaderboard:** **Inputs:** CLOCK **Output:** JC Displayed automatically at the end of gameplay.	1.  .2.  3.   4.  .

Team Members: Yoshizaki Eichi Lance, Kim Gyurim, Zhong Shu, Saravanan Swetha

References and Feedback

Yoshizaki Eichi Lance

- Keyboard Input

Digilent provides a very basic overview of how the usb keyboard input is converted into PS/2-style scan codes using a microcontroller installed on the Basys 3 board, and how to interact with this type of data. See the 'keyboard' section: https://digilent.com/reference/programmable-logic/basys-3/reference-manual. A more in-depth explanation regarding the PS/2 interface can be found here: https://www.burtonsys.com/ps2_chapweske.htm. PS/2 scan codes can be found here: https://www.vetra.com/scancodes.html

Note: the microcontroller on the Basys 3 board seems to convert some of the key inputs to set 2 scan codes while others to set 3 scan codes.

Open source code: Digilent also has published demo code to read keyboard scan codes using a serial terminal application. link: https://digilent.com/reference/programmable-logic/basys-3/demos/keyboard#:~:text=The%20user%20presses%20a%20key%20on%20the%20keyboard%2C,key%20with%20PS%2F2%20code%20XX%20has%20been%20released. I used their code to read the scan codes on the lab computer and determine which keys are interpreted using set 2 scan codes and which ones are set 3 scan codes. I modified their module 'PS2Receiver.v' to also read input from special keys (these keys have an extra E0 sequence contained in the make code but the original code only reads 8 bit make codes). ChatGPT was used to understand the original code as there was close to no documentation on how the code functioned, especially what the different flags in the code mean and how they are used

- Note Grading

Multidimensional arrays in verilog were a new topic for me so I used the following resources as aiding reference on how to handle them:
https://www.javatpoint.com/verilog-arrays
https://stackoverflow.com/questions/28623819/how-to-flatten-array-in-verilog

Kim Gyurim

- Pixel Oled Design
  I had to first map all the display pages onto this website first, which would help me determine the specific pixel constraints for each drawing, then i would have to hardcode all the pixels onto the code.https://www.pixilart.com/helohewowow/gallery

Zhong Shu

- On-screen Display Feedback
  Inpired from a post on Digilent Forum
  https://forum.digilent.com/topic/4731-pmod-oled-how-to-send-char-to-display/
  Referenced ASCII table https://www.asciitable.com/

- Background Music
  Diligent open source code provides a simple demo of connection of AMP2 and how to make simple sound.
  https://digilent.com/reference/_media/reference/pmod/pmodamp2/amp2_library_example.zip

Saravanan Swetha

- Pixel Oled Design
  For coding out the display icons and game screens, I had to initially draw the display out onto the pixelart.com website which allowed me to determine the specific pixel coordinates and the size of each display. Therefore this then made it easier for me to code them out in Verilog.
  https://www.pixilart.com/goldencherryis/gallery