| EECS 2070 02 Digital Design Labs 2019  Final Project  FPGA's Band |
| --- |
| 學號：107000115 姓名：林珈卉  學號：107062119 姓名：吳欣祐 |

設計概念

### 功能描述

FPGA's Band 是一個能創作音樂的編曲程式，使用者使用滑鼠點選螢幕上不同的音軌，並對該音軌進行編輯。使用者亦可選擇單獨播放該音軌、並調節音量大小，也可以同時播放全部的音軌所合成的音樂。

我們提供使用者4個音軌，第一個為音色為電音鍵盤樂器的主旋律，占用1個音道；第二個是音色為鋼琴的和絃背景，因為需要3個音高才能組成一個和絃，故占用3個音道；第三個為音色為貝斯的旋律，占用1個音道；第四個為音色為打擊樂器的節奏，占用1個音道。

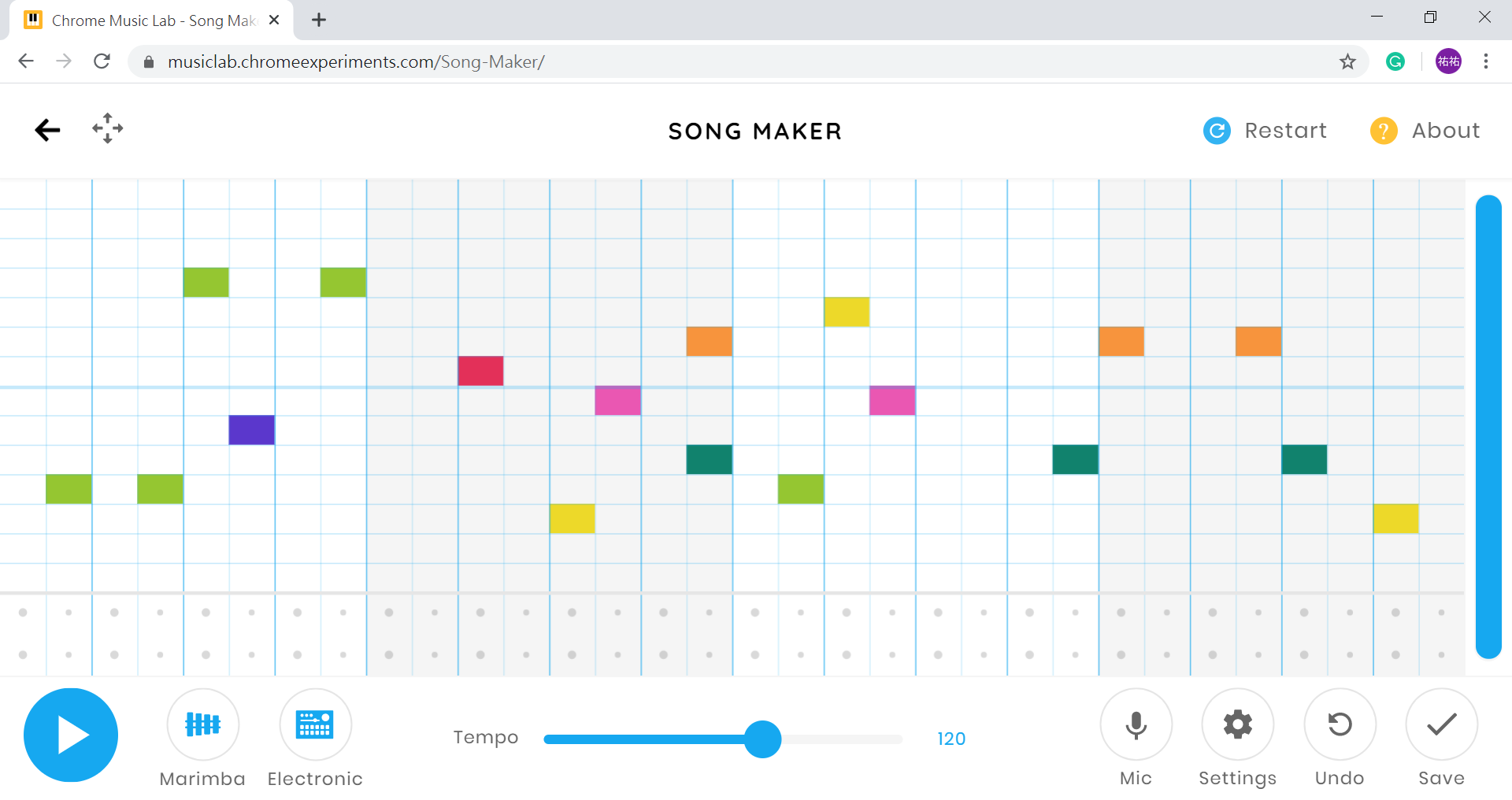
一張含有 室內, 坐, 白色, 黑色 的圖片

自動產生的描述

### 靈感來源

我們的創作靈感來自於 Google 所提供的簡易網頁編曲軟體「CHROME MUSIC LAB: SONG MAKER」，該程式可以編輯四個小節的音樂，可以選擇多種音色，音高範圍則為兩個八度（不含黑鍵部分），節奏也有多種音色，每種音色有兩個音效。

而我們希望使用硬體語言來實作出相似功能的程式，並加上額外的功能。如增加可編輯的音高（含有黑鍵，總共25個音），及增加和絃輸入的功能，讓樂曲可以在和絃架構的基礎下來編輯音樂，還有各種樂器分佈獨立的編輯環境與獨立音軌的播放功能。



架構細節及方塊圖

### 架構細節——畫面

| **畫面** | **功能／畫面** | **細節** |
| --- | --- | --- |
| 主畫面 | 顯示四種樂器  點選可進入該樂器的編輯模式 | 鋼琴：顯示大致旋律音高  貝斯：顯示大致旋律音高  吉他：顯示每小節和絃名稱  打擊：顯示大、小鼓（不同符號） |
| 鋼琴主旋律 | 橫軸為時間軸  縱軸為音高 | 橫軸以八分音符為單位  縱軸總共有兩個八度（25個音） |
| 吉他和絃 | 上方選擇和絃  下方為時間軸 | 以一小節為單位，共四個小節  總共 2\*12 種和絃  （大、小 2 種和絃，12 個根音）  例：C、Cm、C#、C#m |
| 貝斯旋律 | 橫軸為時間軸  縱軸為音高 | 橫軸以八分音符為單位  縱軸總共有兩個八度（25個音） |
| 打擊樂器 | 橫軸為時間軸  縱軸為樂器種類  非獨立畫面，於主畫面編輯 | 橫軸以八分音符為單位  縱軸有兩種音色，大鼓與小鼓 |

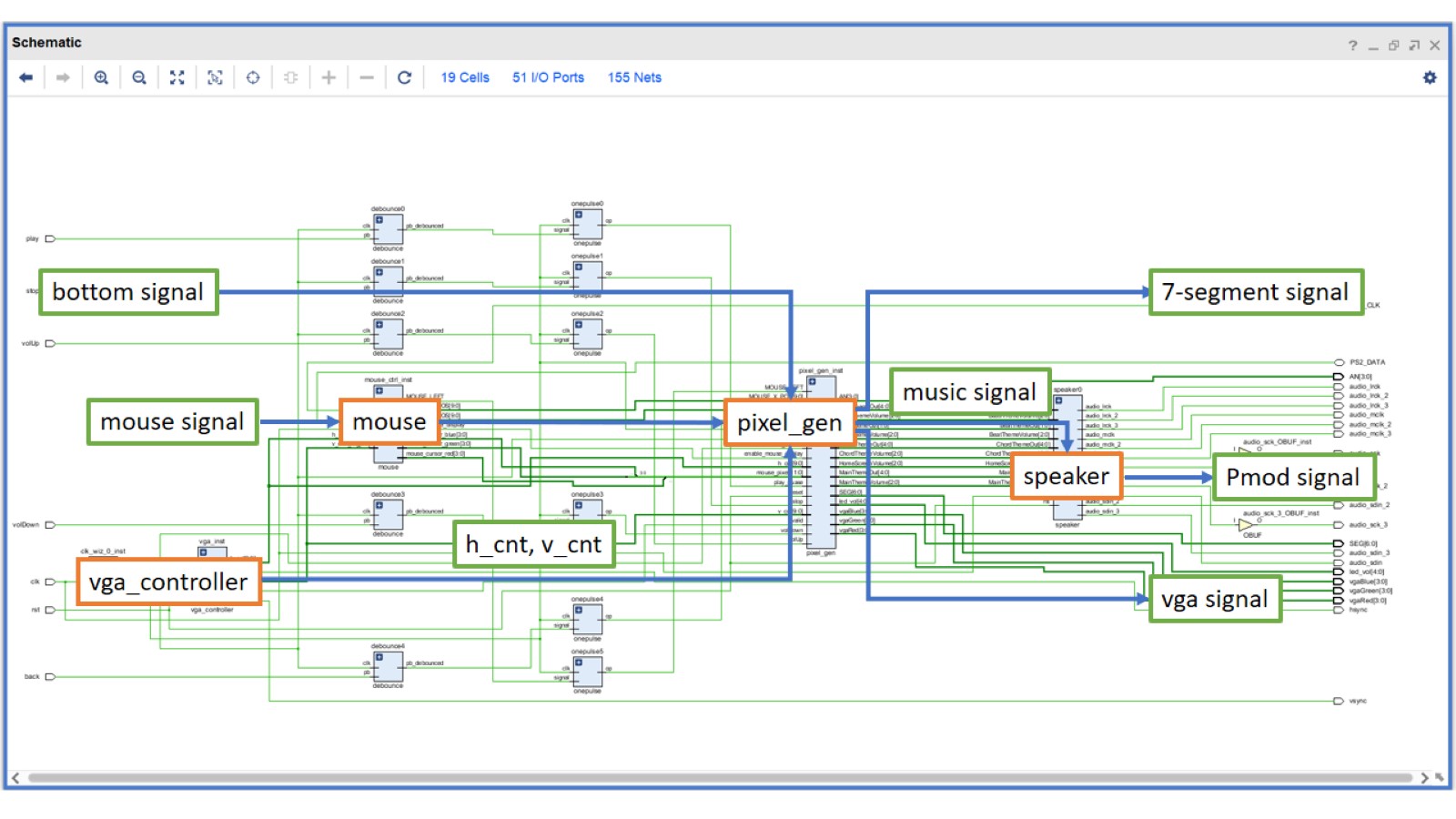
### 

### 

### 架構細節—— IO Devices

| **I/O** | **描述** |
| --- | --- |
| Switch | SW0: rst |
| LED | LED0~LED4: 音量大小顯示 |
| Push button | BTNU:音量增大  BTND:音量減小  BTNL:播放音樂／暫停播放  BTNR:回主畫面  BTNC:停止播放 |
| 7-Segment Display | 顯示當前主旋律音高（例：A4） |
| Mouse | 搭配螢幕，讓使用者輸入音樂、音高 |
| VGA | 輸出螢幕訊號 |
| Audio Amplifier | 播放音樂 |

### 方塊圖



實作方法及難易度說明

| **項目** | **細項** | **實作方法** |
| --- | --- | --- |
| 音色實作 | 音高與頻率 | * 從滑鼠端輸入音高訊號，為音高的編號（主旋律的中央 C 為編號 1，C# 為編號 2 等） * 將編號透過 Note\_Decoder module 轉換成頻率的數值，並傳回 speaker module * Note\_Decoder 如下圖，每個音軌皆有一組 Decoder      * 根據頻率的數值，計算每一個週期的 clk 數量 * 再配合音色，從 note\_gen 輸出訊號給 speaker\_control   實作範例   1. 如中央 A 為 440Hz 2. 每 100000000/440 個 clk 為一個波形週期 3. 主旋律音色，每一個週期的取樣點總共有42個 4. 將取樣點的數值列成陣列（數列命名 w1, w2, w3...） 5. audio\_right 等訊號每經過 (100000000/440)/42 個 clk 會改變至下一個數值（從 w1 變成 w2） |
| 音色波形 | 1. 將該音色的音檔輸入 Sonic Visualiser 程式      1. 取用中段的訊號波形，因前段的波形含有觸鍵的音色，較為混雜，無法使用；後段的音色則逐漸消逝，泛音音色明顯突出        1. 取用一個週期的波形，紀錄每個取樣點的數值      1. 使用 2’s Complement 的格式儲存 16 bit 的數值陣列，下圖為鍵盤樂器的音色，共有 42 個取樣點 |
| 打擊樂器 | 使用 sine 函數，音高從高頻率迅速降至低頻率，製造出打擊樂器的效果。  下圖為大鼓的頻率變化，及兩種鼓在調整 EQ 時，各頻率分布的可用性分析。大鼓頻率的可用範圍約為 20 至 200Hz，小鼓則為 150 至 250Hz。 |
| 和絃訊號 | 輸入此 module 的和絃訊號數值範圍為 0~23，分別代表 24 種和絃，因此我們需要 Note\_Decoder 去把和絃訊號拆解為 3 個單音的頻率訊號。 |
| ADSR 實作 | * 為了製造出更像真實樂器所發出的音色，必須加上「觸鍵」以及「漸出」的效果，而這個部分可以應用到混音學的 ADSR 理論。 * ADSR 包含 Attack、Decay、Sustain 以及 Release，我使用 FSM 來實作這個部分。 * reg [1:0] 變數命名為 ADSR，數值為 2’b00 時為 A state，2’b01 時為 D state，2’b10 時為 S state，2’b11 時為 R state。 * reg 變數 rate 為音量的倍率，用來控制 audio\_left 等變數所輸出的數值。 * 每個 state 的功能   + A state     - next\_rate = rate + 1     - if (rate >= 10) next\_ADSR = D   + D state     - next\_rate = rate - 1     - if (rate <= 7) next\_ADSR = S   + S state     - 使用 counter 來計算 S state 的時間長度     - if (ADSR\_cnt > 20) next\_ADSR = R   + R state     - if (rate > 0) next\_rate = rate - 1 |
| 螢幕顯示 | 圖片顯示 | 我使用 block memory generator 的 IP 來儲存我在project 中所需要用到的圖片，並且透過控制 memory address 來決定要輸出的訊號。 |
| 框線 | 在此 project 中的表格框線都是經由判定 h\_cnt, v\_cnt 的值來決定的，也因此框線並不會消耗記憶體的資源。 |
| 音樂播放 | 我使用了 FSM 來控制音樂播放，state transition graph 如上圖，另外 FSM 也會輸出 play\_cnt 訊號，這個訊號代表當前播放到第幾個 8 分音符，會在 vga 顯示上用到。 |
| 旋律儲存 | 我選擇使用 register 來儲存旋律，原因是因為簡單好實作、且只需要 1 個 clock cycle 便能儲存，也可以直接存取使用。 |

分工

|  | **Modules** | **詳細內容** |
| --- | --- | --- |
| 林珈卉 | speaker  Note\_Decoder  note\_gen  speaker\_control | 實作音色、音高控制  根據音高訊號轉換成頻率以及音色  輸出聲音訊號 |
| 吳欣祐 | Mouse  pixel\_gen  vga  數個 ip | 處理使用者輸入的滑鼠訊號  轉換成音高訊號，傳遞至 speaker module  輸出螢幕畫面的訊號 |

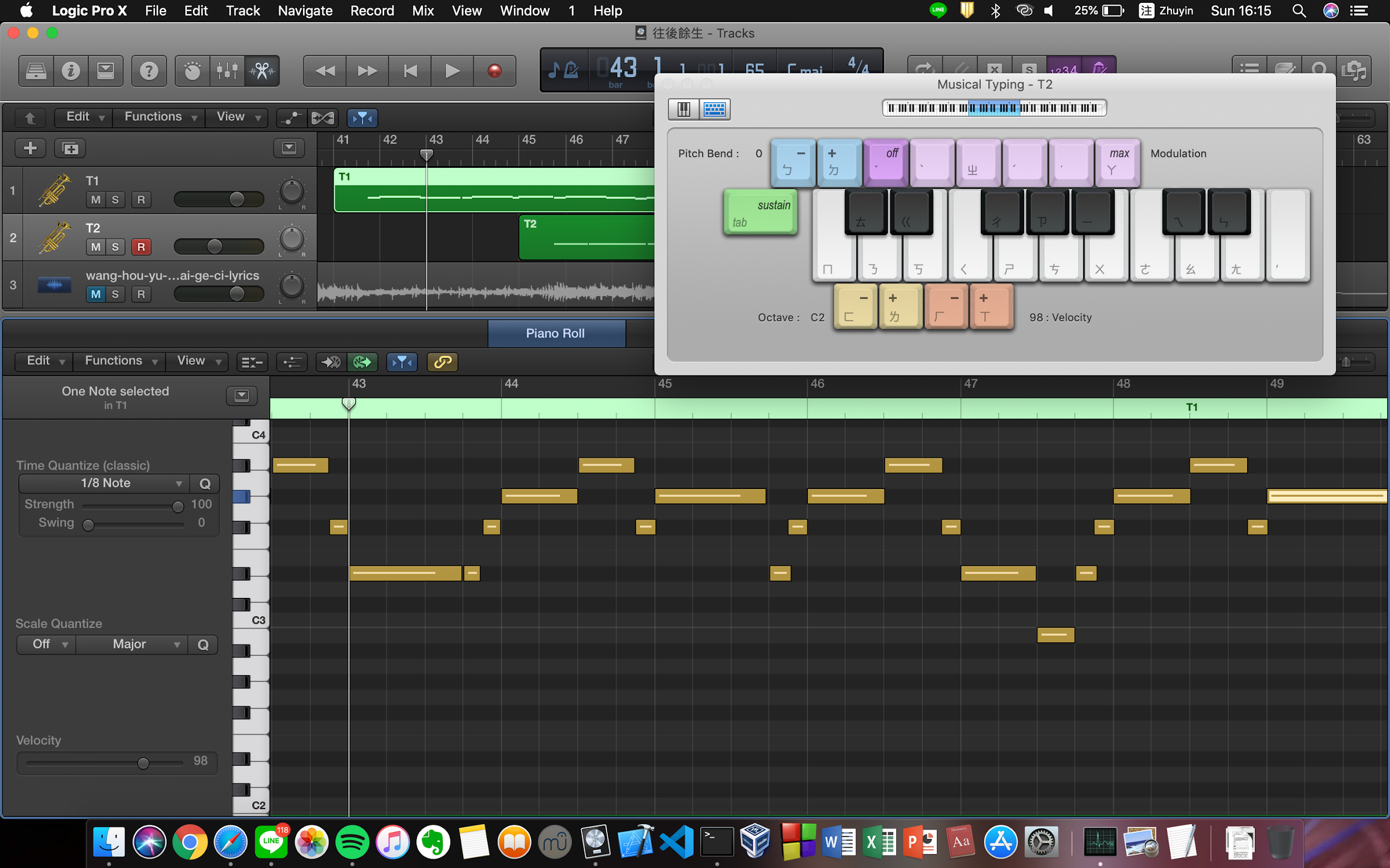
## 

## 困難與解決方法

* 在 pixel\_gen module 中我發現圖片最左側兩個 pixel 無法正常顯示，會有黑點不斷跳動，因為找不到此問題的成因，因此我的解決方法為直接將最左兩個 pixel 設為白色。
* Speaker 實作
  + 在 note\_gen 的 ADSR FSM 中，需要判斷是否有新的音輸入，若是以輸入的音高數值與前者不同來作為判斷依據，就無法連續播放兩個相同的音高。因此需要輸入一個 input 訊號命名為 newpulse，assign 為 (freq != `silence)。使用相同頻率的 clk ，在 FSM 更新的 always block 中判斷若 new\_pulse == 1，next\_ADSR = A，next\_rate = init\_val。
  + 在 note\_gen module 中需要對 audio\_right 等變數賦值，這些變數為 2’s complement 的格式，因此音色的數值陣列也必須為 2’s complement。而數值會根據 rate 乘上倍率，所以 rate 也必須為 2’s complement。也可選擇在 always block 賦值時，變數前面加上 $signed 即可。
  + Speaker module 需要輸出六個音道的訊號，因此要將 master clock、left-right clock、serial clock 及 audio\_sdin 這四個訊號複製為三份，並連接到三個 Pmod 晶片上。

## 心得討論

* 吳欣祐

我覺得 pixel\_gen module 就技術層面來說不會很困難，因為將前面 lab 教的東西靈活運用便能完成，反之讓我比較吃力的是在處理表格和圖片上比較龐雜繁大的工作量。

* 林珈卉

很高興能在這次的 Final Project 中發揮我過去所學的知識，及實作出一個跟我喜歡的領域相關的作品，也謝謝這堂課程、教授、助教們提供這個機會。

過去有些編曲和混音的經驗，使用的是 macOS 的 Logic Pro 軟體。在設計這個 FPGA’s Band 時，除了上面介紹的網頁版編曲程式外，也有參考一些 Logic Pro X 的介面、輸入格式等等。