**Lab 8 110060027朱豐蔚**

1. **exp\_1 (key board with display function)**

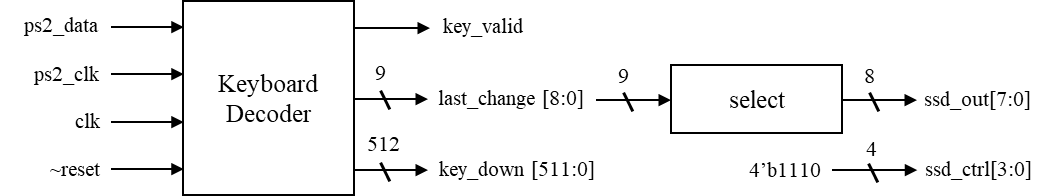
**Design Specification**

Input: clk, reset.

Inout: ps2\_data, ps2\_clk.

Output: [3:0] ssd\_ctrl, [7:0] ssd\_out.

**Design Implementation**

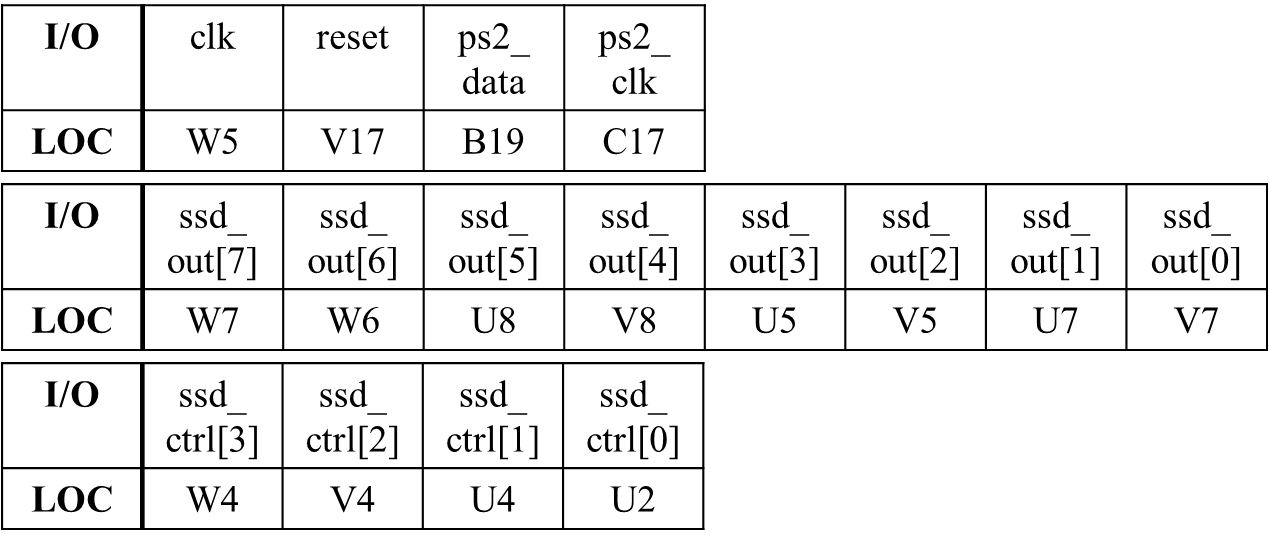
 Block diagram:

Design flow:

使用附上的KeyboardDecoder對ps2\_data和ps2\_clk訊號進行編譯，轉變為供後續使用的key\_valid、last\_change、key\_down訊號。

在exp\_1中，我只使用到last\_change訊號，在select模組中，我使用case的寫法，當last\_change為題目所求的0 ~ 9、A、S、M時，分別顯示出各自代表的符號。此外，我將ssd\_ctrl永遠assign為4’b1110，目的是讓SSD中僅有第四位顯示所求。

注意：在exp\_1中，我定義的A為上橫線符號、S為中橫線符號、M為下橫線符號。

I/O pins assignment

**Discussion:**

在一開始，我在使用KeyboardDecoder模組上遇到了不少問題，第一是我不小心將ps2\_data和ps2\_clk的inout打成input，進而一直出現Error的訊息。第二是我的reset習慣上是使用負邏輯，但附上的KeyboardDecoder模組卻是使用正邏輯，這點讓我的鍵盤一直沒反應，也是找了許久才發現這個Bug。

我認為exp\_1的難度非常剛好，沒有太多複雜的功能，讓我們能把注意完全放於KeyboardDecoder模組上，進而了解如何去操作與使用模組，以面對後續的其他題目。

1. **exp\_2 (single digit decimal adder with keyboard input function)**

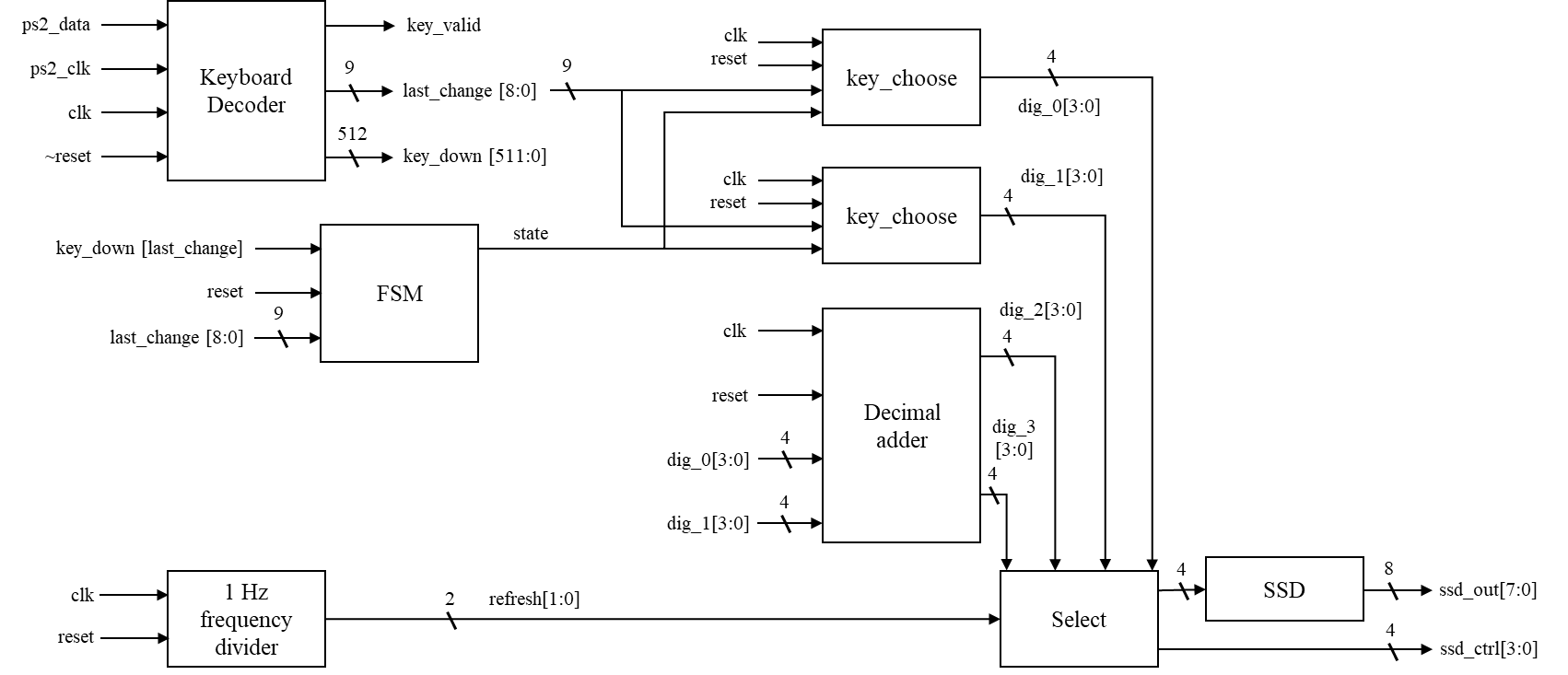
**Design Specification**

Input: clk, reset.

Inout: ps2\_data, ps2\_clk.

Output: [3:0] ssd\_ctrl, [7:0] ssd\_out.

**Design Implementation**

 Block diagram:

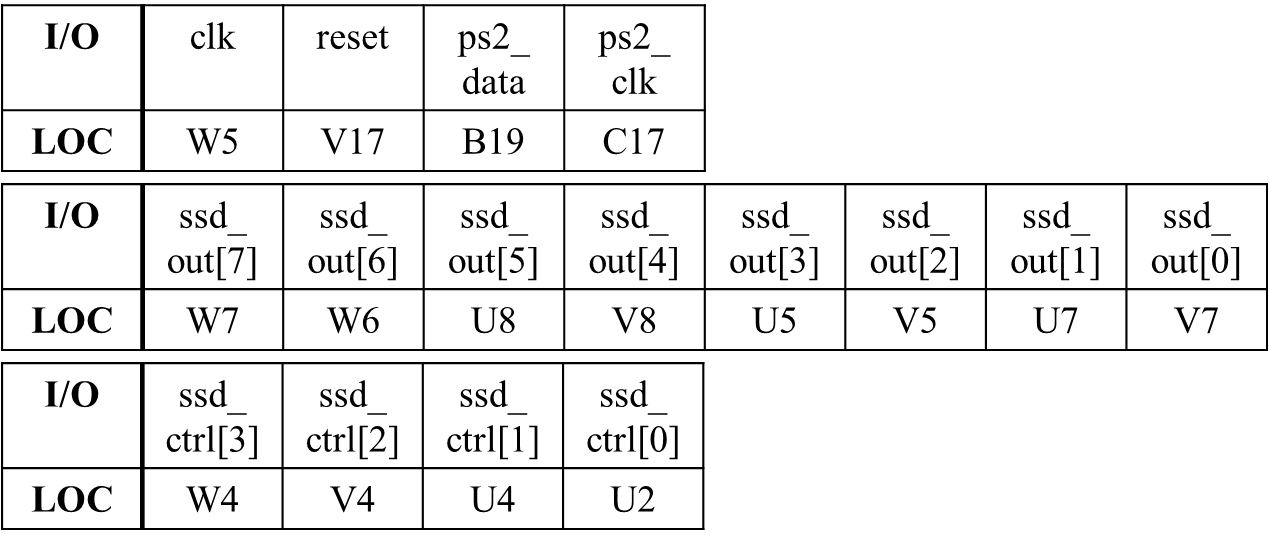
Design flow:

同exp\_1，使用附上的KeyboardDecoder對ps2\_data和ps2\_clk訊號進行編譯，轉變為供後續使用的key\_valid、last\_change、key\_down訊號。

在FSM當中分為2個state，分別代表輸入要給第一位數字還是第二位數字，當state為0時，讓key\_choose去抓取第一位數字；而當state為1時，讓key\_choose去抓取第二位數字。注意：初始下state為0，並使用enter鍵作為state的切換。

使用decimal adder模組對第一位和第二位數字作加法，並輸出第三位與第四位數字，在exp\_2中，我設計的decimal adder是只要前兩位數字發生改變就會跟著改變的。

最後如同前幾次的Lab，我使用1hz frequency divider去產生refresh訊號，去更新SSD的顯示，使四個數字可以各自被顯示出來。

I/O pins assignment :

**Discussion:**

在exp\_2中，我比較沒有遇到太多的問題，最大的挑戰應該就是如何撰寫key\_choose模組去抓取所需的數字，我最後決定使用類似exp\_1中case的寫法，針對last\_change，當其為0 ~ 9的數字時，就讓該位子的數字顯示出來，並進行後續的加法運算。

此外，decimal adder的設計讓我回想上學期在上邏輯設計中有教過，當和大於9，就要再額外加6，讓數字能夠呈現十進位的模式。

在exp\_2中，不但使用了以往學過的知識，還進一步需要自己的思考與發想，並且在完成的時候，會有滿滿的成就感。

1. **exp\_3 (two-digit decimal adder/subtractor/multiplier)**

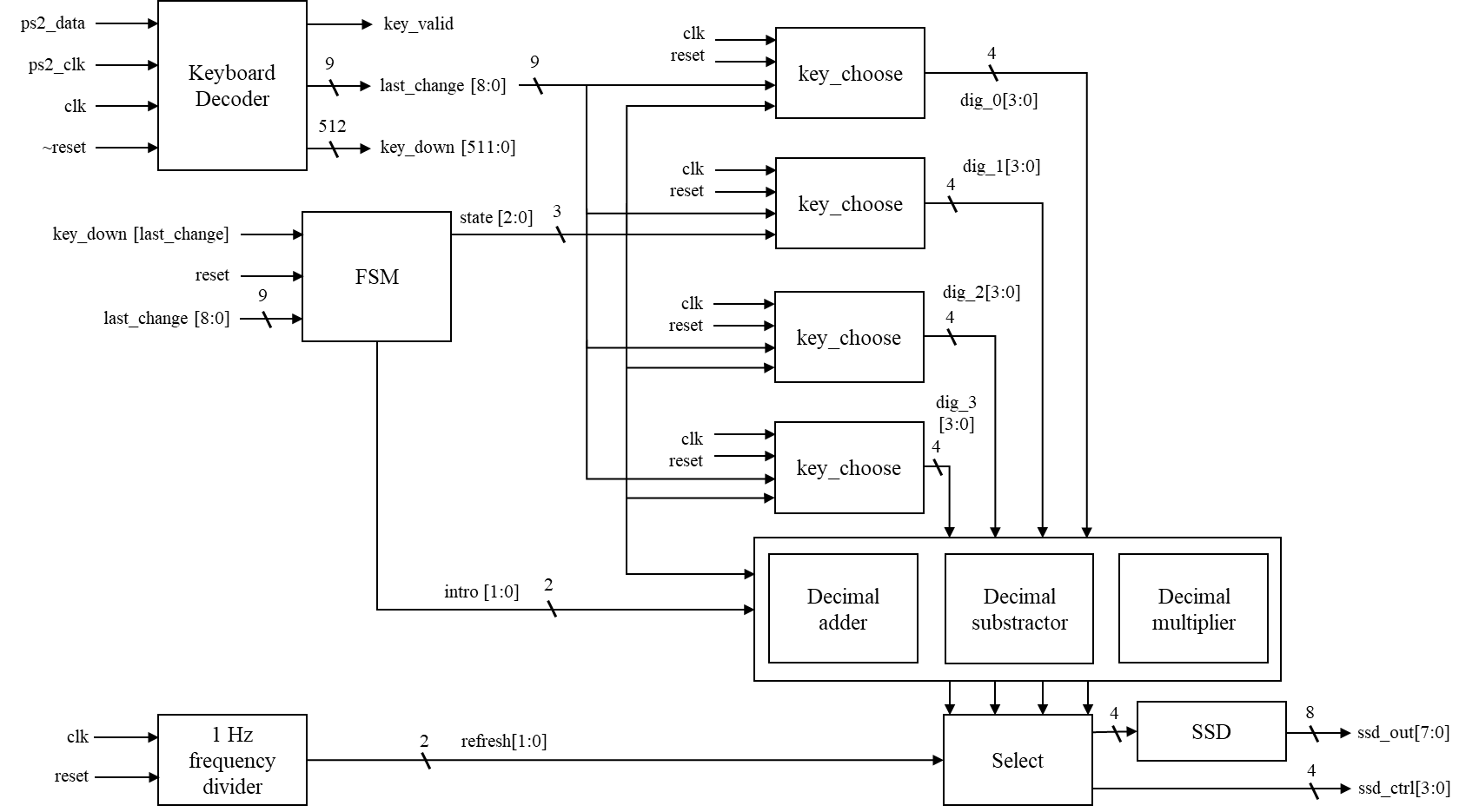
**Design Specification**

Input: clk, reset.

Inout: ps2\_data, ps2\_clk.

Output: [3:0] ssd\_ctrl, [7:0] ssd\_out.

**Design Implementation**

 Block diagram:

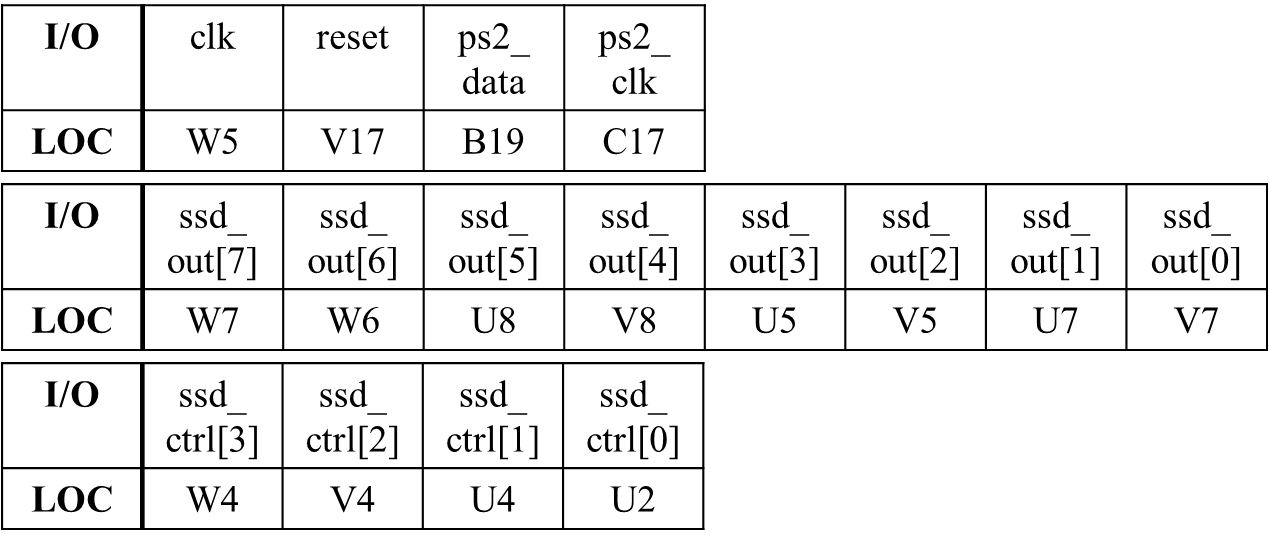
Design flow:

同exp\_1，使用附上的KeyboardDecoder對ps2\_data和ps2\_clk訊號進行編譯，轉變為供後續使用的key\_valid、last\_change、key\_down訊號。

FSM當中分為7個state，當state為0時，讓key\_choose去抓取第一位數字；當state為1時，讓key\_choose去抓取第二位數字；當state為2時，讓FSM去抓取要進行加減乘的哪一項，並輸出為intro訊號供後續使用；當state為3時，讓key\_choose去抓取第三位數字；當state為4時，讓key\_choose去抓取第四位數字；當state為5時，讓FSM去等待enter鍵的訊號，並保持著原本四個數字的顯示；當state為7時，讓SSD進行對應的計算結果顯示。

在計算的部分，加法器、減法器、乘法器都會永遠運作，intro訊號只會影響呈現的結果，這樣的設計能夠省去還要對各個運算方塊做enable訊號的過程。我透過always的寫法，去選擇出真正需要被顯示的結果為何，再輸出到後續的select模組中。

最後如同前幾次的Lab，我使用1hz frequency divider去產生refresh訊號，去更新SSD的顯示，使四個數字可以各自被顯示出來。

I/O pins assignment :

**Discussion:**

在本次實驗，我覺得最困難的部分就是減法中正負號的判別，我最後選擇透過多重if、else的寫法，對各種位元彼此之間的大小關係進行列舉，並選出之中為負的情形，將負號的訊號傳出，讓後續的選擇能夠判斷出是正還是負。

1. **exp\_4 (keyboard with caps function)**

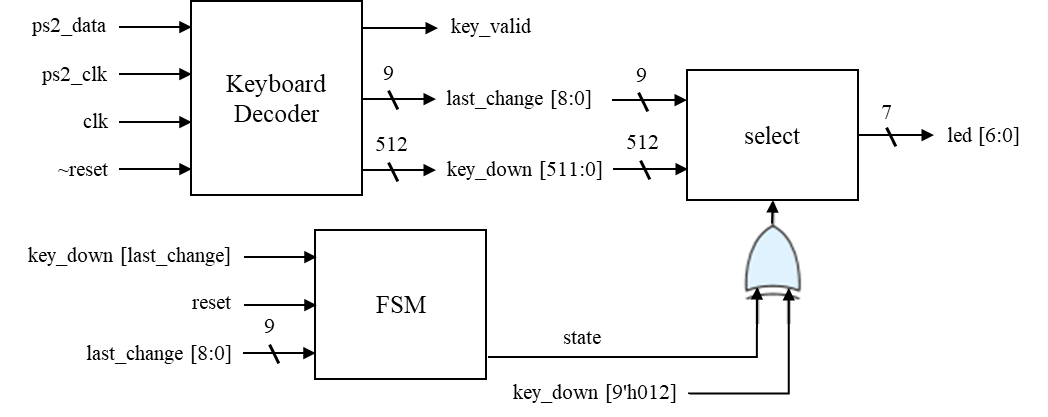
**Design Specification**

Input: clk, reset.

Inout: ps2\_data, ps2\_clk.

Output: [6:0] led, state.

**Design Implementation**

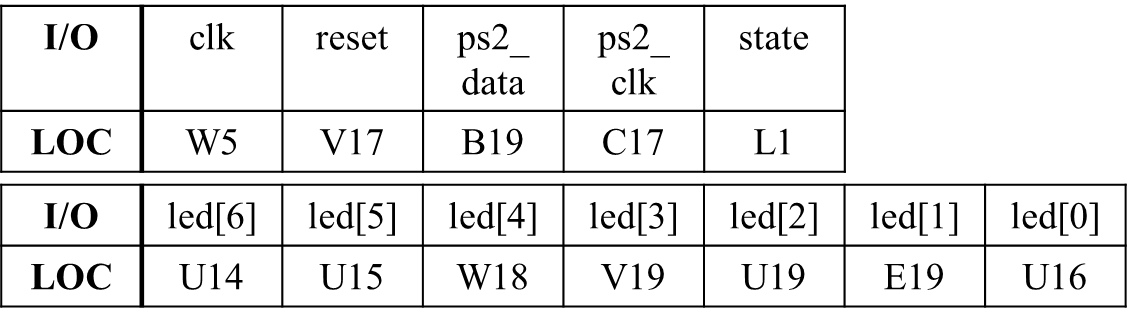
 Block diagram:

Design flow:

同exp\_1，使用附上的KeyboardDecoder對ps2\_data和ps2\_clk訊號進行編譯，轉變為供後續使用的key\_valid、last\_change、key\_down訊號。

在FSM當中分為2個state，分別代表現在是cap打開狀態還是關閉狀態。我透過key\_down [last\_change]當作clk，去判斷last\_change是否為cap鍵訊號，若是，就使FSM的state切換狀態。

將state與keydown [9’012] (shift鍵)訊號做XOR，即能夠達到題目所求的大小寫需求，最終將此訊號丟入select中，去針對按下的A到Z鍵做各自大小寫的ASCII顯示。

I/O pins assignment :

**Discussion:**

在本次實驗中我們學到了複合鍵的撰寫與應用，事實上，在日常生活的鍵盤應用中，複合鍵是被大量使用的，不管是大小寫切換、標點符號，還是玩遊戲時的快捷鍵等，這次的實驗很好的幫助我去思考且了解複合鍵內部的深奧與廣大，甚至同樣的鍵位還會有優先度的差別，這無疑都大大增加了在設計時的複雜度與困難度。

**======================================================================**

**Conclusion**

回顧Lab 7與Lab 8，在前者，是透過外接模板，去達成輸出的效果，而後者則是使用USB，達成輸入的效果。在現實中的種種應用中，往往輸入輸出是相輔相成的，一個完善的工程裝置，抑或是電子操控設計，都是輸入與輸出並存的實際應用。

老師也說過了，以往是會有將鍵盤輸入與音效輸出兩者統整在一起的Lab，我還是覺得這學期Lab中沒有能夠做出類似的設計是一件很可惜的事情，除了錯過能學習到統整輸入輸出在一塊的能力，更可惜的是錯失了能夠親眼看見自己的的雙手透過按鍵盤，就能讓耳機發出聲音，這件超有成就感的體驗，只能待final project中再去實現這個目標。

**References**

鍵盤IP: (from 第八週課堂網頁)

鍵盤接腳代號查詢: (from 第八週上課講義)