Lab 8

110060027 朱豐蔚

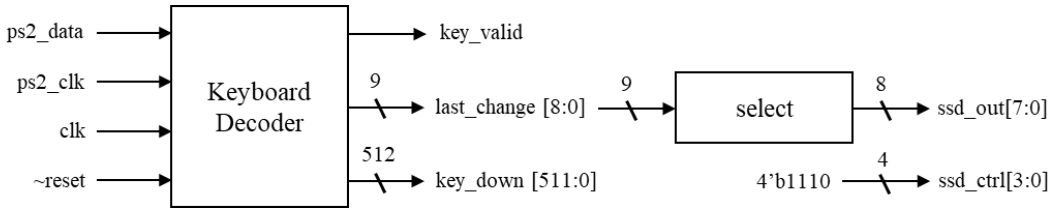
I. exp_1 (key board with display function)

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

I/O	clk	reset	ps2_ data	ps2_ clk
LOC	W5	V17	B19	C17

I/O	ssd_ out[7]	ssd_ out[6]	ssd_ out[5]	ssd_ out[4]	ssd_ out[3]	ssd_ out[2]	ssd_ out[1]	ssd_ out[0]
LOC	W7	W6	U8	V8	U5	V5	U7	V7

I/O	ssd_ ctrl[3]	ssd_ ctrl[2]	ssd_ ctrl[1]	ssd_ ctrl[0]
LOC	W4	V4	U4	U2

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。

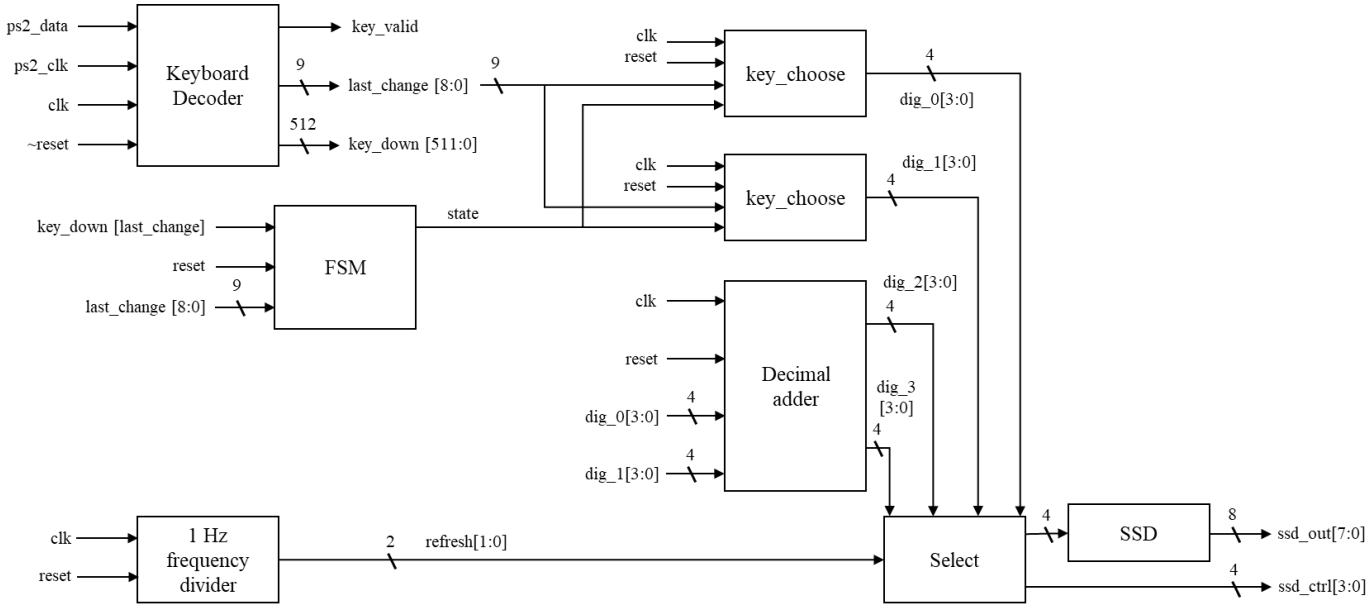
II. exp_2 (single digit decimal adder with keyboard input function)

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:

I/O	clk	reset	ps2_ data	ps2_ clk
LOC	W5	V17	B19	C17

I/O	ssd_ out[7]	ssd_ out[6]	ssd_ out[5]	ssd_ out[4]	ssd_ out[3]	ssd_ out[2]	ssd_ out[1]	ssd_ out[0]
LOC	W7	W6	U8	V8	U5	V5	U7	V7

I/O	ssd_ ctrl[3]	ssd_ ctrl[2]	ssd_ ctrl[1]	ssd_ ctrl[0]
LOC	W4	V4	U4	U2

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。

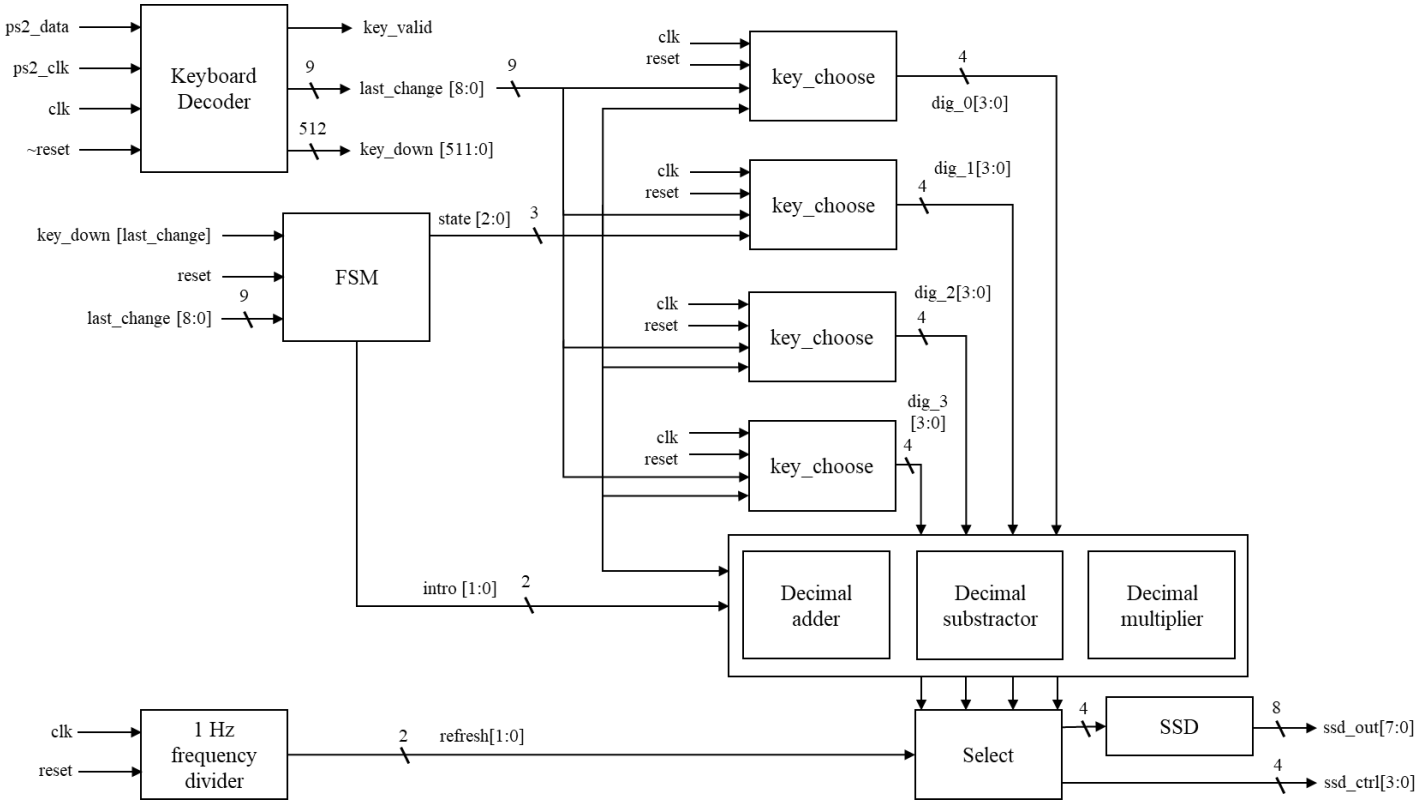
III. exp_3 (two-digit decimal adder/subtractor/multiplier)

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:

I/O	clk	reset	ps2_ data	ps2_ clk
LOC	W5	V17	B19	C17

I/O	ssd_ out[7]	ssd_ out[6]	ssd_ out[5]	ssd_ out[4]	ssd_ out[3]	ssd_ out[2]	ssd_ out[1]	ssd_ out[0]
LOC	W7	W6	U8	V8	U5	V5	U7	V7

I/O	ssd_ ctrl[3]	ssd_ ctrl[2]	ssd_ ctrl[1]	ssd_ ctrl[0]
LOC	W4	V4	U4	U2

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。

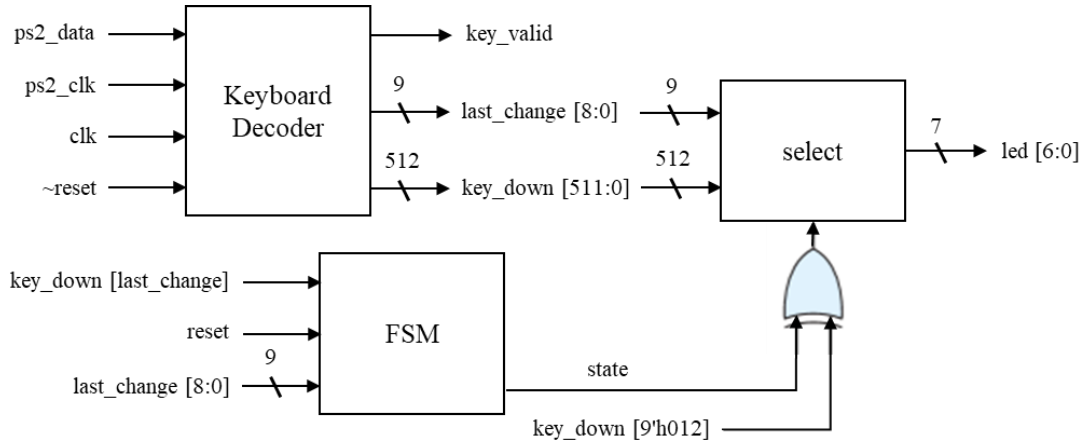
IV. exp_4 (keyboard with caps function)

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:

I/O	clk	reset	ps2_ data	ps2_ clk	state
LOC	W5	V17	B19	C17	L1

I/O	led[6]	led[5]	led[4]	led[3]	led[2]	led[1]	led[0]
LOC	U14	U15	W18	V19	U19	E19	U16

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)