Lab2: FPGA Emulation

107061112 王昊文

Experiments

1. Emulate exp1 in lab1 (a full adder ***s*** + ***cout***=***x*** + ***y*** + ***cin***) with the following parameters.

I/O	*x*	*y*	*cin*	*s*	*cout*
LOC	V17	V16	W16	U16	E19

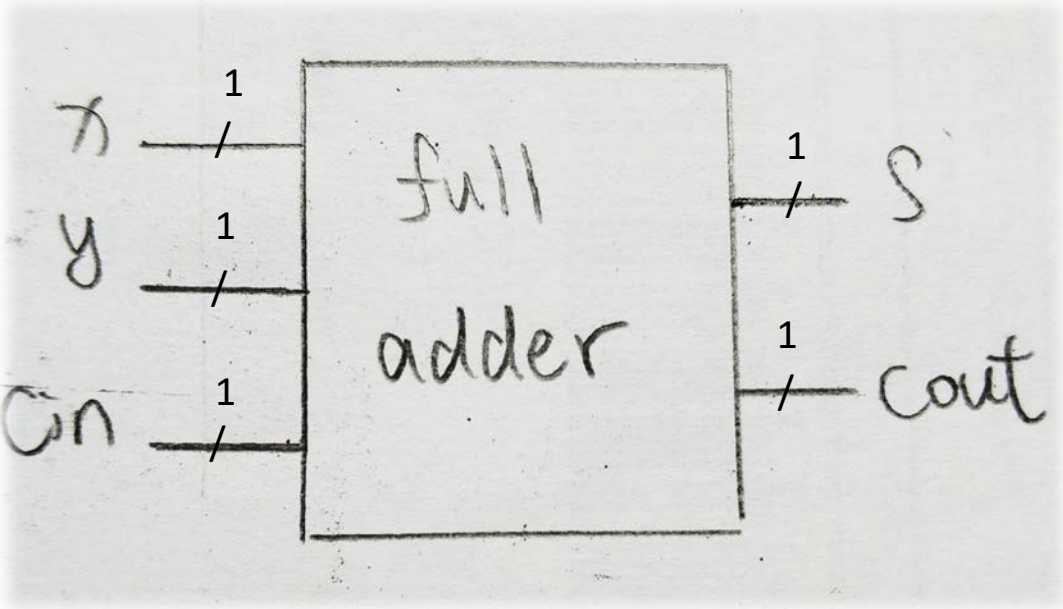
Design Specification

✓ I/O

Input x, y, cin

Output s, cout

✓ Logic Block:



輸入三個input，通過full adder邏輯得到兩個輸出。

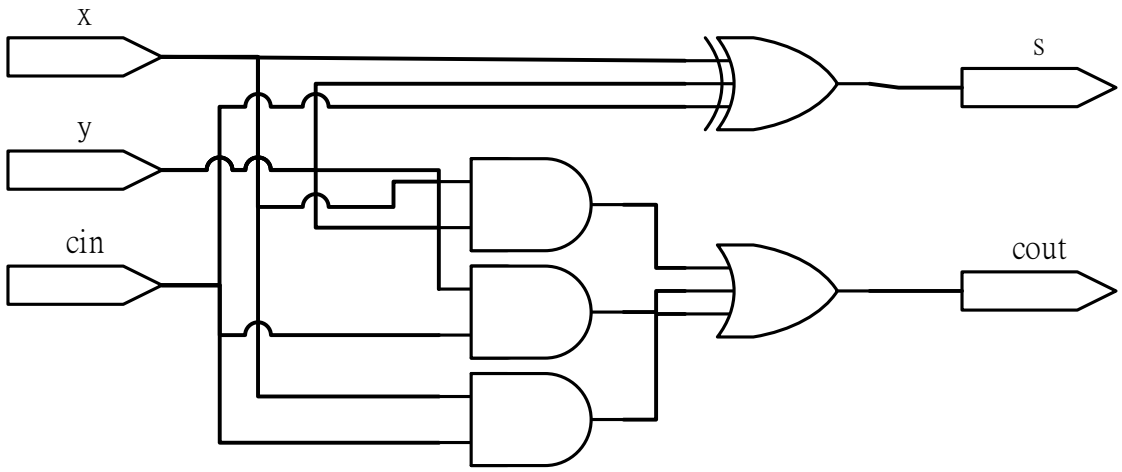
Design Implementation

✓ Logic Function:

cout = (x &cin) | (x & y) | (cin & x)

s = x ^ y ^ cin

✓ Logic Diagram:



✓ IO Pin

Input			Output	
x	*y*	*cin*	*cout*	*s*
V17	V16	W16	E19	U19

Discussion

這一題算是讓我們體驗看看接Pin的感覺，第一次找按鍵找了蠻久的。基本上只要邏輯沒有出錯，Pin符合題目要求基本上就不會出錯。

2. Derive a BCD (***i***[3:0]) to 7-segment display decoder (***D_ssd***[7:0]), and also use four LEDs (***d***[3:0]) to monitor the 4-bit BCD number. (Other values of ***i*** outside the range will show F).

Design Specification

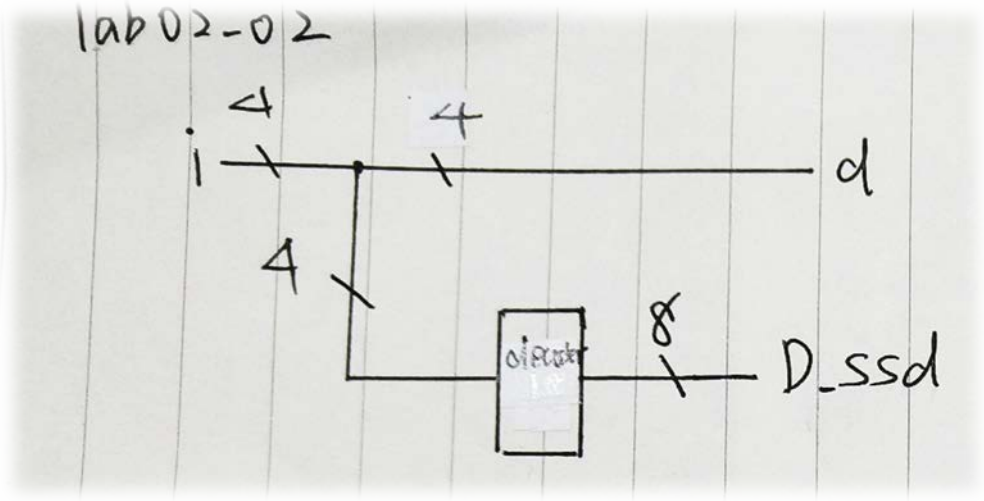
✓ I/O

Input [3:0]i // BCD input

Output [3:0]d // LED output

Output [7:0]D_ssd // Seven Segment

✓ Logic Block:



輸入i當作Pin輸入d直接連結到輸出。剩餘輸入前往decoder，選擇想要輸出的數字。

Design Implementation

✓ Logic Function:

d = i

we define the 7 segments, then for the case of i, we assign the segments to D_ssd.

For cases bigger than 9, we assign F to D_ssd.

```
case (i)
        4'd0: D_ssd = `SS_0;
        4'd1: D_ssd = `SS_1;
        4'd2: D_ssd = `SS_2;
        4'd3: D_ssd = `SS_3;
        4'd4: D_ssd = `SS_4;
        4'd5: D_ssd = `SS_5;
        4'd6: D_ssd = `SS_6;
        4'd7: D_ssd = `SS_7;
        4'd8: D_ssd = `SS_8;
        4'd9: D_ssd = `SS_9;
        default: D_ssd = `SS_f;
```

✓ IO Pin

input [3:0]i			
i[0]	i[1]	i[2]	i[3]
V17	V16	W16	W17

output [3:0]d			
d[0]	d[1]	d[2]	d[3]
U16	E19	U19	V19

output [7:0]D_ssd							
D_ssd[0]	D_ssd[1]	D_ssd[2]	D_ssd[3]	D_ssd[4]	D_ssd[5]	D_ssd[6]	D_ssd[7]
V7	U7	V5	U5	V8	U8	W6	W7

Discussion

其實我期待這一刻真的很久！讓LED亮起來是沒什麼感覺，但是看到能讓SSD亮起自己想要的數字感覺真的很好。這個實驗中，剛開始一直無法顯示自己想要的字，後來發現是接PIN的時候接反，導致無法顯示正確的數字。還有Define的部分設定數字的亮暗也容易設定錯誤。

3 Derive a binary (***i***[3:0], 0-9, a, b, c, d, e, f) to 7-segment display decoder (***D***[7:0]), and alsouse four LEDs (***d***[3:0]) to monitor the 4-bit binary number.

Design Specification

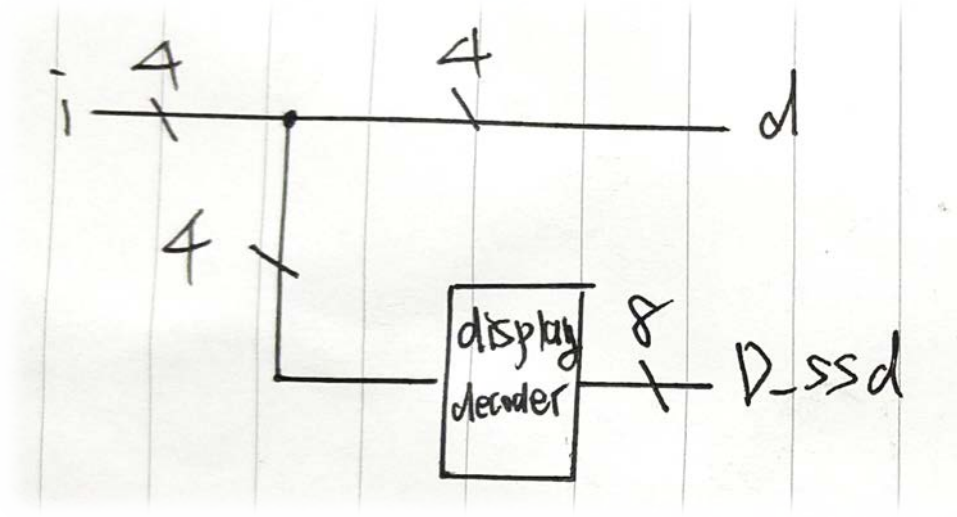
✓ I/O

Input [3:0]i // BCD input

Output [3:0]d // LED output

Output [7:0]D_ssd // Seven Segment

✓ Logic Block:



邏輯與前一題類似。

Design Implementation

✓ Logic

Similar to the previous experiment, the only difference is for 10~15 we have to assign the corresponding alphabet.

✓ IO Pin

input [3:0]i			
i[0]	i[1]	i[2]	i[3]
V17	V16	W16	W17

output [3:0]d			
d[0]	d[1]	d[2]	d[3]
U16	E19	U19	V19

output [7:0]D_ssd							
D_ssd[0]	D_ssd[1]	D_ssd[2]	D_ssd[3]	D_ssd[4]	D_ssd[5]	D_ssd[6]	D_ssd[7]
V7	U7	V5	U5	V8	U8	W6	W7

Discussion

這一題跟上一題相當的類似，只是必須額外補上 a~f 的 define.

4 (Bonus) Design a combinational circuit that compares two 4-bit unsigned numbers A and B to see whether A is greater than B. The circuit has one output X such that X = 0 if A ≤ B and X = 1 if A > B. (let A[3:0], B[3:0] be controlled by 8 DIP switches, the binary numbers are displayed on 8 LEDs. The result X is on another LED.)

Design Specification

✓ I/O

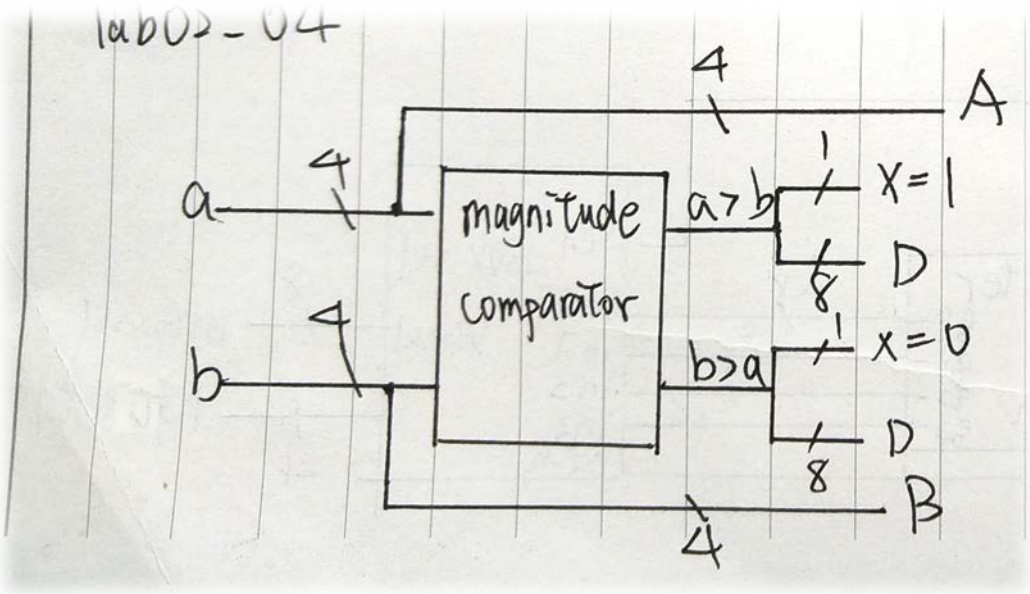
Input [3:0]A, B // input number

Output [3:0]a, b // LED output of the number

Output X // If A > B, then, X = 1

Output [7:0]D // Seven Segment

✓ Logic Block:



a,b input 直接接到A, B output，然後a, b進入comparator比較，然後輸出大的值。

Design Implementation

✓ Logic

A = a // assign input to output

B = b

If A > B, X = 1, display A,

Else X = 0, display B

✓ IO Pin

input [3:0]A			
A[0]	A[1]	A[2]	A[3]
W15	V15	W14	W13

input [3:0]B			
B[0]	B[1]	B[2]	B[3]
V17	V16	W16	W17

output [3:0]a			
a[0]	a[1]	a[2]	a[3]
W18	U15	U14	V14

output [3:0]b			
b[0]	b[1]	b[2]	b[3]
U16	E19	U19	V19

output [7:0]D							
D [0]	D [1]	D [2]	D [3]	D [4]	D [5]	D [6]	D [7]
V7	U7	V5	U5	V8	U8	W6	W7

Output X -> L1

Discussion

這一題Bonus基本上，只要前面Segment Display會做，這一題也不成問題。題目只規定在LED上顯示大小關係，但我接受老師的建議把比較大的數字顯示在ssd上。想法也很簡單，在大小比較完之後透過MUX選擇想要顯示的數字即可。

Conclusion

這次的lab難度有小小提升，但基本上上學期邏輯設計好好學的話應該都不難。這次的實驗接線的部分讓我開始思考了很多延伸的功能，紅綠燈等等或許並不是很難做到的。