Lab 4 report

107061218 謝霖泳

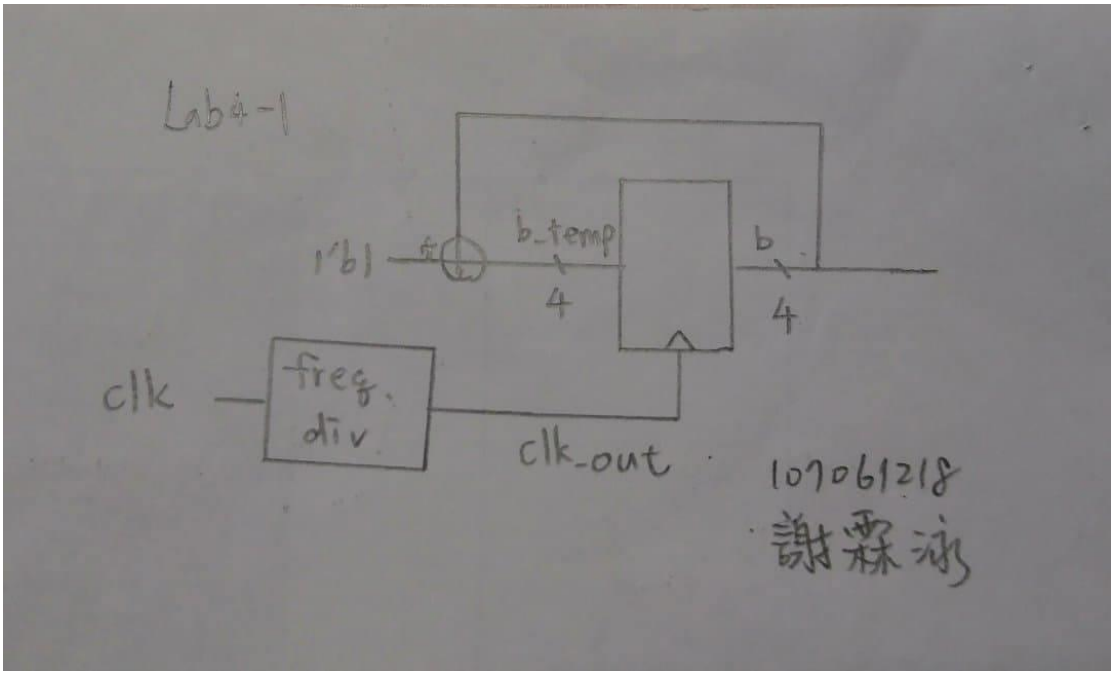
1.

- Design specification

  For a 4-bit synchronous binary up counter:

  Input: clk, rst

  Output: [3:0] b



- Design Implementation

  將 clk 通過之前設計過的除頻器得到 1Hz 的頻率，即上圖中的 clk_out，此 1Hz 的頻率也將做為加法器的 D-FF 的時脈衝動。

  b_temp 經過前面的加法器運算完後就是現在的 b 再加 1'b1，當 clock 的 positive edge trigger 來的時候，b 的值就更新為剛剛已經+1 的 b_temp，達到 4-bit synchronous binary up counter 的效果。

  最後，將這個 4-bit 的 output b 傳到 LED 燈，將使相應的 LED 燈在該 bit 為 1 的時候發亮。

  IO pin assignment:

clk	rst	b [3]	b [2]	b [1]	b [0]
W5	V17	V19	U19	E19	U16

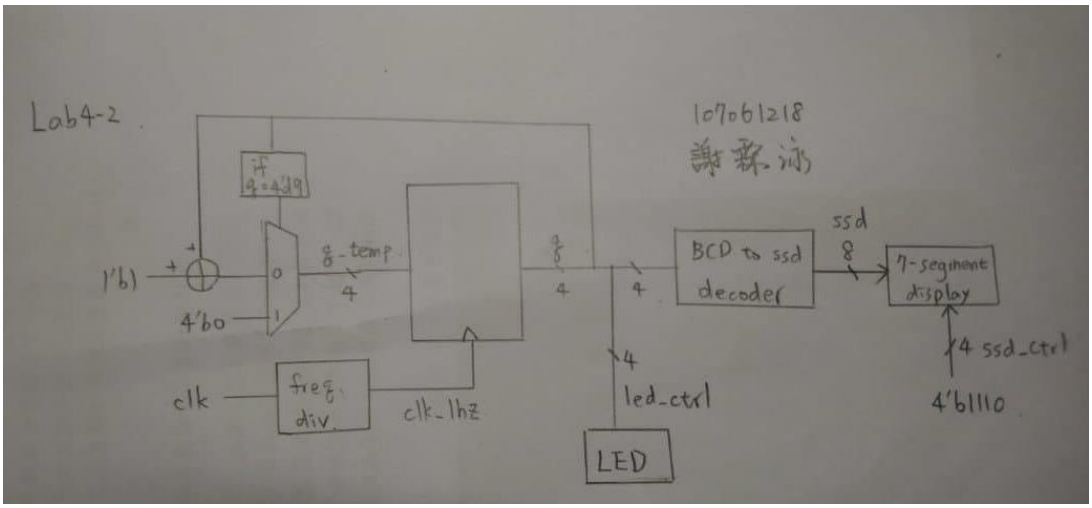
- Discussion

當我們將 rst 由 0 變成 1，也就是將最右邊的開關往上扳，這個 counter 便會開始運作。一開始只亮最右邊的 LED，也就代表著 0001，再來，每秒這個值都會+1，所以，LED 燈也會亮出相對應的數值，到 b = 4'b1111，也就是 4'd15 時，4 個 LED 燈就會全亮。

2.

- Design specification

For a single digit BCD up counter:
Input: clk, rst
Output: [7:0] ssd, ssd_ctrl, led_ctrl



- Design Implementation

將原始的 clk 通過之前設計過的除頻器得到 1Hz 的頻率，即上圖中的 clk_1hz，此 1Hz 的頻率也將做為加法器的 D-FF 的時脈衝動。

加法器的結構與之前大同小異，唯一的不同點在於：因為它是一個 BCD up counter，所以，數值最大不會超過 9。因此，在 q_temp 前須多加 1 個 MUX，其功能為檢查現在的 q 值是否為 4'd9，若不是，則下一個值就為現在的 q 值+1，因此 q_temp 就會選擇前面經由加法器運算完的值；反之，若 q 已經是 9 了，下一個 q 值就必須歸零，因此 q_temp 就要選擇 0。

再來，為了使相應位數的 LED 燈亮起來，所以，需要一個 4-bit 的 output led_ctrl 來控制。因為 LED 燈本身為 high active control，而哪個位數為 1 就要哪個 LED 燈亮起，所以 led_ctrl = q。

至於顯示到七段顯示器的部分，我引用之前實驗設計過的 BCD to 7-segment decoder，將 q 傳入這個 BCD to 7-segment decoder 中，即可得到 8-

bit 的 output，是為[7:0] ssd，將這個 8-bit 的結果傳送到七段顯示器的適當腳位，即可顯示相對應的數值。

此外，因為 BCD 的關係，數值最大為 9，因此在我的設計中，只希望七段顯示器亮最右邊的位數，所以我多加了一個 4-bit 的 input 名為 ssd_ctrl，其值設定為 4'b1110，並將腳位分別設定為七段顯示器的四個位數，因為顯示器本身為 low active control，如此一來，就只會亮最右邊的那一個位數。

IO pin assignment:

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

led_ctrl [3]	led_ctrl [2]	led_ctrl [1]	led_ctrl [0]
V19	U19	E19	U16

ssd_ctrl [3]	ssd_ctrl [2]	ssd_ctrl [1]	ssd_ctrl [0]
W4	V4	U4	U2

clk	rst
W5	V17

- Discussion

  當我們將 rst 由 0 變成 1，也就是將最右邊的開關往上扳，這個 counter 便會開始運作。一開始只亮最右邊的 LED，也就代表著 0001，再來，每秒這個值都會+1，所以，LED 燈也會亮出相對應的數值，於此同時，七段顯示器也會在最右邊那個 digit 顯示出相對應的數值。直到 b = 4'b1001，也就是 4'd9 之後，4 個 LED 燈就都會暗掉，七段顯示器顯示的值也會回到 0。

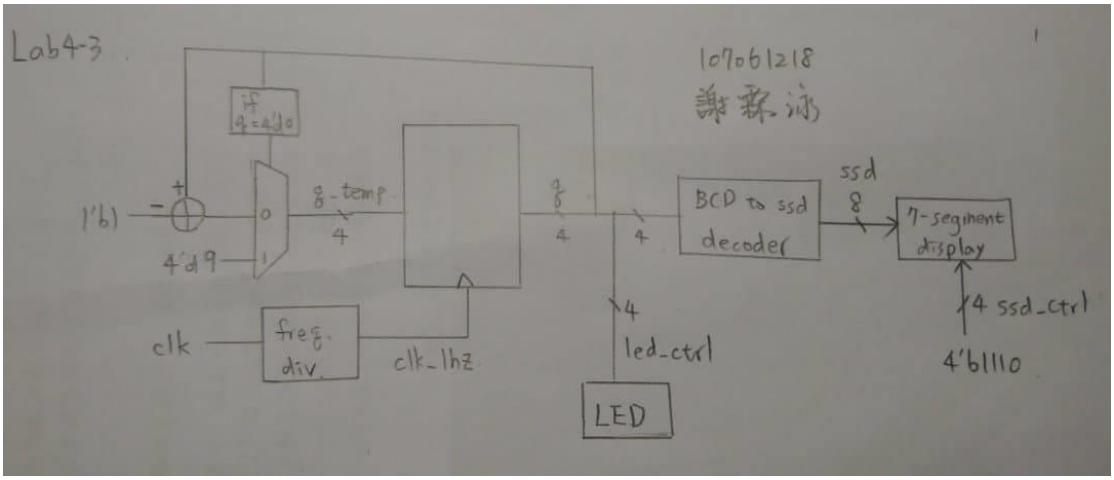
3.

- Design specification

  For a single digit BCD up counter:

  Input: clk, rst

  Output: [7:0] ssd, ssd_ctrl, led_ctrl



- Design Implementation

本題與上一題相似，不同點僅在於上數與下數，也就是 q_temp 前的 MUX 判斷。

將原始的 clk 通過之前設計過的除頻器得到 1Hz 的頻率，即上圖中的 clk_1hz，此 1Hz 的頻率也將做為加法器的 D-FF 的時脈衝動。

減法器的結構與之前大同小異，唯一的不同點在於：因為它是一個 BCD down counter，所以，數值最大不會超過 9，最小不會低於 0。因此，在 q_temp 前須多加 1 個 MUX，其功能為檢查現在的 q 值是否為 4'd0，若不是，則下一個值就為現在的 q 值-1，因此 q_temp 就會選擇前面經由減法器運算完的值；反之，若 q 已經是 0 了，下一個 q 值就必須變為 9，因此 q_temp 就要選擇 4'd9。

再來，為了使相應位數的 LED 燈亮起來，所以，需要一個 4-bit 的 output led_ctrl 來控制。因為 LED 燈本身為 high active control，而哪個位數為 1 就要哪個 LED 燈亮起，所以 led_ctrl = q。

至於顯示到七段顯示器的部分，我引用之前實驗設計過的 BCD to 7-segment decoder，將 q 傳入這個 BCD to 7-segment decoder 中，即可得到 8-bit 的 output，是為[7:0] ssd，將這個 8-bit 的結果傳送到七段顯示器的適當腳位，即可顯示相對應的數值。

此外，因為 BCD 的關係，數值最大為 9，因此在我的設計中，只希望七段顯示器亮最右邊的位數，所以我多加了一個 4-bit 的 input 名為 ssd_ctrl，其值設定為 4'b1110，並將腳位分別設定為七段顯示器的四個位數，因為顯示器本身為 low active control，如此一來，就只會亮最右邊的那

一個位數。

- Discussion

  當我們將 rst 由 0 變成 1，也就是將最右邊的開關往上扳，這個 counter 便會開始運作。一開始會亮最右邊以及最左邊的 LED，也就代表著 1001，再來，每秒這個值都會-1，所以，LED 燈也會亮出相對應的數值，於此同時，七段顯示器也會在最右邊那個 digit 顯示出相對應的數值。直到 b = 4'b0000 之後，4 個 LED 燈就都會變回原始的亮暗暗亮，也就是 9 的意思，七段顯示器顯示的值也會回到 9。

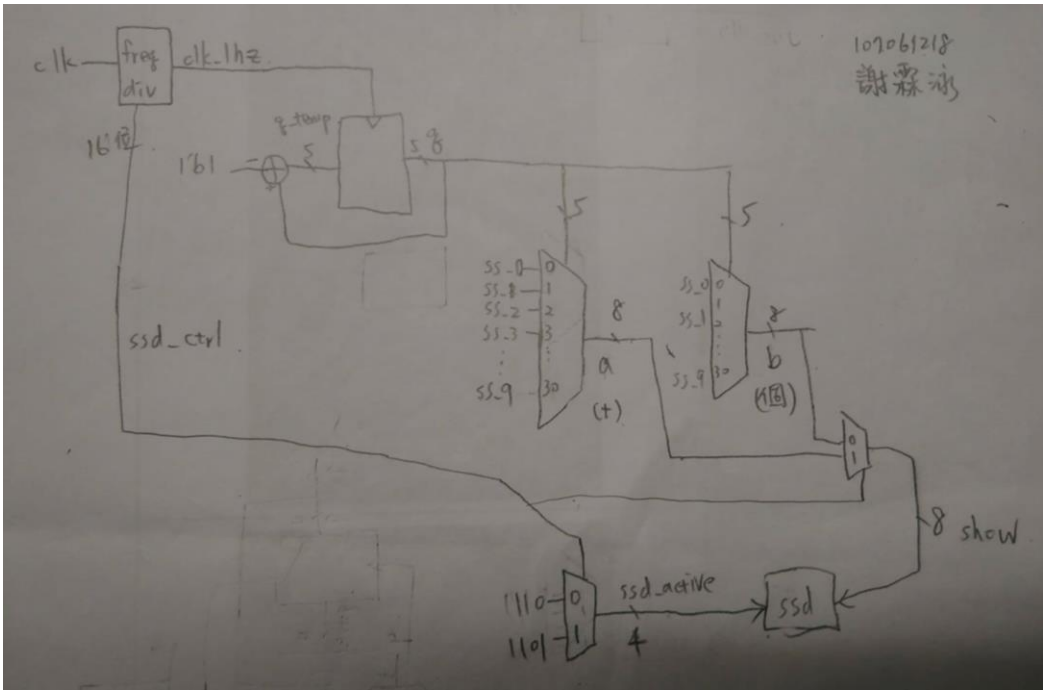
4.

- Design specification

  For a 30 sec count down timer:

  Input: clk, rst

  Output: [7:0] show, [3:0] ssd_active



- Design Implementation

  設置一個 5-bit 的減法器，從 30 開始往下減。該減法器的 clk 為通過除頻器出來的 1Hz 時脈，每經過 1 秒，q 的值就會-1。

  將 q 的值作為右邊兩個 MUX 的 selection，左邊的 MUX 功能為判斷十位數，也就是比如當 q = 5'd2x 時，8-bit 的變數 a 就會選擇 0 經過 ssd decoder 的值，同理，右邊的 MUX 功能為判斷個位數，比如當 q = 5'dx4 的時候，8-bit 的變數 b 就會選擇 4 經過 ssd decoder 的值。

得到十位數和個位數的 ssd decoder 值後，要將之利用視覺暫留顯示到七段顯示器上，因此，引用之前的 27-bit 加法除頻器的第 16 個 bit，名為 ssd_ctrl，作為視覺暫留位數間快速變換的頻率。而 8-bit 變數 show 的功能為控制要顯示的內容，意即 ssd_ctrl = 0 時，MUX 會將 8-bit 變數 show 的結果選擇為個位數的 decoder 值；ssd_ctrl = 1 時，MUX 會將 8-bit 變數 show 的結果選擇為十位數的 decoder 值。

經過前面的操作，我們已經可以快速變換要顯示的內容。再來，這個快速變換的 ssd_ctrl 也將成為另一個 MUX 的 selection，功能為用來選擇要亮的位數。而 4-bit 變數 ssd_active 就是用來控制我要亮哪個位數。當 ssd_ctrl = 0 時，因為經過上段的操作，我們知道這時候亮的內容為個位數，因此 ssd_active = 1110；當 ssd_ctrl = 1 時，因為經過上段的操作，我們知道這時候亮的內容為十位數，因此 ssd_active = 1101。如此一來，便可以在「亮個位數，顯示內容為個位數」與「亮十位數，顯示內容為十位數」之間作及快速的變換而產生視覺暫留，便可製造出倒數計時器的效果。

- Discussion

  當我們將開關打開，這個 timer 便會從 30 秒開始往下數，依序為 30、29、28……03、02、01、00。每秒變動一次，符合我們對於 30 sec count down timer 的要求。

5.conclusion

經過了這次的 lab 我更清楚 clk 的運作方式以及除頻器的使用，也了解到 shift register 與 counter 如何運用在真實的 design 當中。

這次的第二題與第三題大同小異，而且加法器或減法器的觀念在之前就已經教過，只要修改一點點地方就可以了，所以這兩題沒有花我太多時間。反而是最後一題的 30 秒 timer 花費我比較多時間，因為要先把整個 block diagram 都想清楚，再加上 verilog code 除錯的部分，就花掉不少時間。