**Lab 4 110060027朱豐蔚**

1. **exp\_1(1 Hz binary down counter)**

**Design Specification**

Input: clk, reset.

Output: b[3:0].

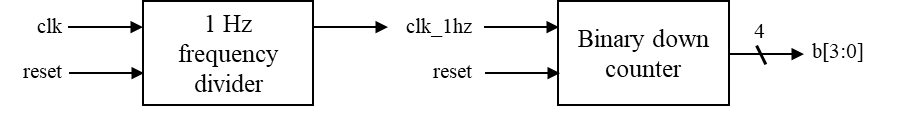
Wire: clk\_1hz.

**Design Implementation**

Specification of the frequency divider:

使用一數50M次的counter來對原本100M Hz的clk進行除頻，使輸出為1 Hz的訊號clk\_1hz，即達成題目所求。

Block diagram:

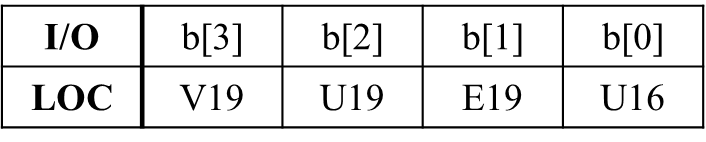
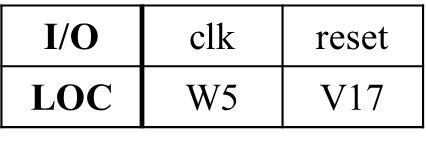


Design flow:

考慮down counter，我使用always的寫法，初始值設為b = 0，並使b\_next永遠都為b – 1，而當clk的正緣訊號產生時，就讓b的值等於b\_next，即可達成down counter的設計。

I/O pins assignment

Input: Output:



**Discussion:**

本實驗是考驗我們對block互相串接的熟悉程度，使用在Lab3製作的1Hz除頻器，並將在Lab3中大量使用的up counter改造為down counter，透過兩者的串接，就可以達到題目所要求的效果。

1. **exp\_2 (1 Hz binary down counter with SSD)**

**Design Specification**

Input: clk, reset.

Output: q[7:0].

Wire: clk\_1hz, b[3:0].

**Design Implementation**

Specification of the frequency divider:

使用一數50M次的counter來對原本100M Hz的clk進行除頻，使輸出為1 Hz的訊號clk\_1hz，即達成題目所求。

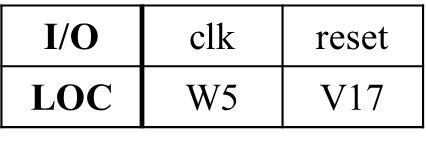
 Block diagram:

Design flow:

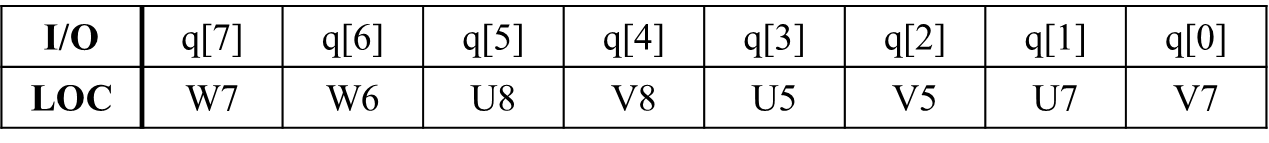
同exp\_1中的除頻器與down counter，可以直接拿來做使用，但為了要使結果能夠呈現於FPGA板上，我們還需要引用Lab2-2中的SSD模組，來讓4-bit的binary訊號轉為7段顯示器的8-bit顯示輸入。

I/O pins assignment

Input:



Output:



**Discussion:**

在exp\_1中，我們使用4個燈去做輸出，雖然已經很習慣binary訊號的讀取，但一 加上了exp\_2的SSD，更能讓我們一眼就看出數字的變化有沒有符合我們的預期，而這樣的顯示也讓我們在隨著clk一秒一秒數的過程更有成就感。

1. **exp\_3 (0.5Hz BCD down counter with SSD)**

**Design Specification**

Input: clk, reset.

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

Wire: clk\_0p5hz, b[3:0].

**Design Implementation**

Specification of the frequency divider:

使用一數100M次的counter來對原本100M Hz的clk進行除頻，使輸出為0.5 Hz的訊號clk\_0p5hz，即達成題目所求。

 Block diagram:

Design flow:

在此題中，相較於exp\_2，頻率變為0.5 Hz、counter變為BCD counter、SSD顯示的數字也變為只能輸出single digit。

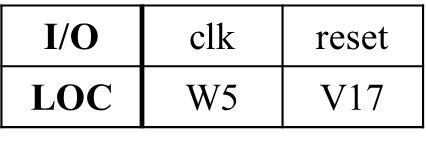
針對0.5 Hz，我們只需要將原本數50M次的counter改為100M即可。

而針對BCD counter，我考慮b\_next的值會隨著b值有所不同，若b現在為0，則b\_next就應該為9而不是b - 1 = 15，在其他情況下，b\_next都為b – 1。而一樣，初始值b = 0，當正緣的clk訊號輸入時，就使b = b\_next。

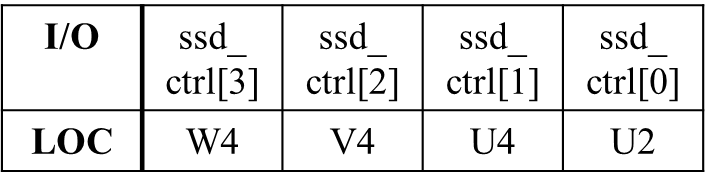
最後考慮single digit的輸出問題，我讓ssd\_ctrl永遠為1110，即SSD上的數字永遠只有第四位會顯示，即可達成輸出single digit的要求。

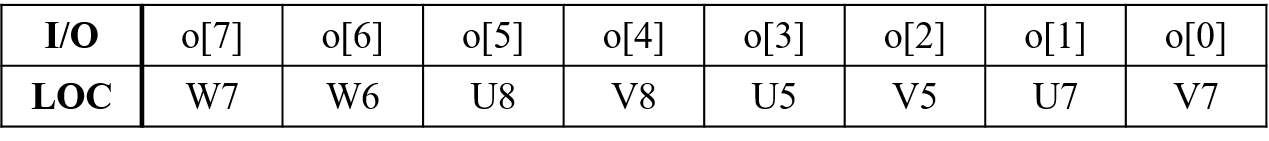
I/O pins assignment

Input:



Output:





**Discussion:**

我認為exp\_3是exp\_4的前哨站，不管是在single digit的寫法上，還是BCD counter的要求，都為我們進行exp\_4打下很好的基礎，也方便我們後續在進行exp\_4時能夠更快上手。

1. **exp\_4 (1 Hz BCD 2-digit up counter with SSD)**

**Design Specification**

Input: clk, reset.

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

Wire: clk\_1hz, dig\_0[3:0], dig\_1[3:0], dig\_choose[3:0], refresh, borrow.

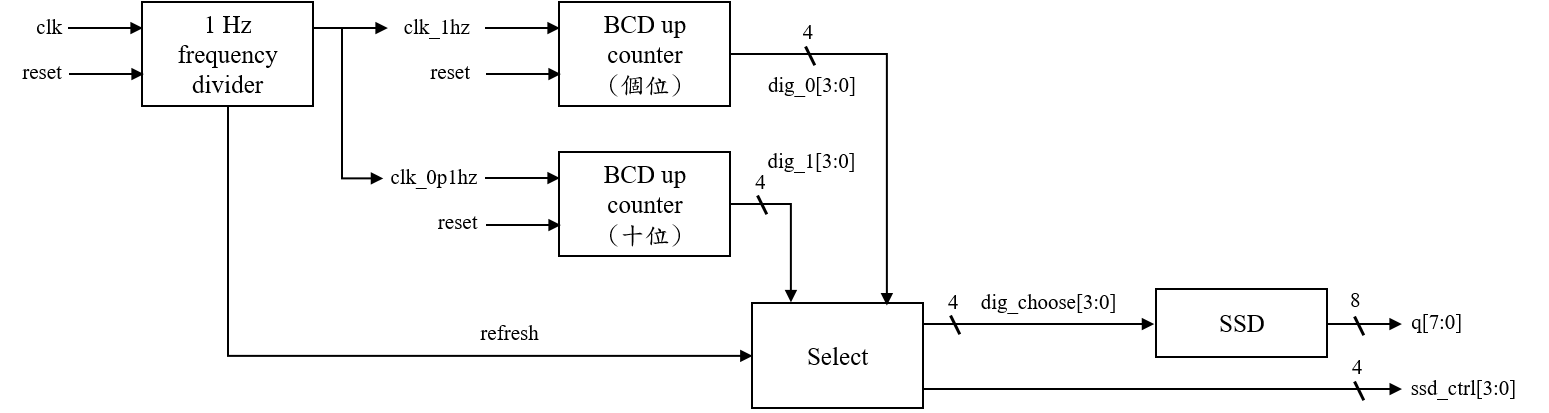
**Design Implementation**

Specification of the frequency divider:

使用一數50M次的counter來對原本100M Hz的clk進行除頻，使輸出為1 Hz的訊號clk\_1hz，即達成題目所求。

此外，assign counter的第19位元作為refresh，以提供後續select的運作。

(若取太後面的位元，會導致模糊閃爍；若取太前面的位元，會導致視覺暫留不明顯)

 Block diagram:

Design flow:

在此題中，相較於exp\_3，要解決3項問題，第一是BCD down counter改為BCD up counter，第二是雙位元彼此間的進位問題，第三則是select的編寫。

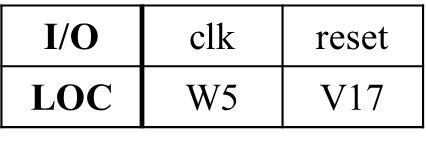
針對第一項問題，我考慮b\_next的值會隨著b值有所不同，若b現在為9，則b\_next就應該為0而不是b + 1 = 10，在其他情況下，b\_next都為b + 1。而一樣，初始值b = 0，當正緣的clk訊號輸入時，就使b = b\_next。

而針對第二項問題，我使用了一個數5次的counter，來將clk\_1hz轉變成clk\_0p1hz的訊號，而這個新產生的訊號就是給處理十位數的BCD counter的clk作為使用，使每數10秒就會讓十位數加一，達到進位的效果。

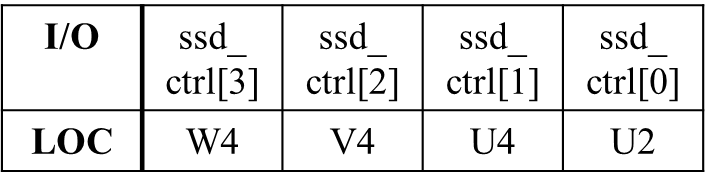
最後則是select的問題，這次的select相較於Lab3-5較為簡單，因為這次只需要選擇兩項即可。而同樣的，我使用case的寫法，當refresh為0時，引入dig\_choose = dig\_0，且ssd\_ctrl = 1110，此時最右側的燈會更新為個位數字顯示；同樣的，當refresh為1時，引入dig\_choose = dig\_1，且ssd\_ctrl = 1101，此時右側第二個燈會更新為十位數字顯示。而由於refresh的訊號變動夠快，視覺暫留的影響下，就會讓SSD看起來為兩位數字同時獨立顯現，達成題目的要求。

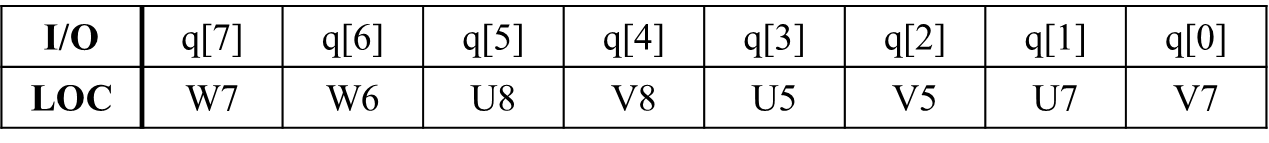
I/O pins assignment

Input:



Output:





**Discussion:**

我認為exp\_4的題目是前面四次Lab題目的綜合體，用到了Lab2的SSD顯示、用到了Lab3的除頻器與SSD獨立顯示方式、更用到了這次Lab就已經有了的counter寫法。

而在過程中，困擾我最久的就是全新出現的進位問題，由於是第一次出現，讓我在思考如何進位時遇到了很大的挑戰，我首先設想了兩種進位方式，第一種方式是當個位數為9時去enable處理十位數的BCD counter。而第二種正如上述，使用0.1hz的clk作為十位數的進位驅動，但後來經過考慮，我發現不管是在reset的設定，還是clk本身正緣驅動的影響，都會影響到最後0.1 Hz可能會與1 Hz處在不相同的clk上，而導致可能會出現延遲顯示的問題，故在往後的Lab中，我應該都會選擇以第一種作為進位的設定，來確保進位的獨立與穩定性。

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

**Conclusion**

在這次Lab，最讓我印象深刻的題目就是exp\_4，如果我上面所說，exp\_4有點像是前面4週所有所學的綜合體題目，雖然在counter的編寫上沒有變得更難、SSD的更新顯示也比Lab3-5簡單，但當同時要處理那麼大量不同的block去做功能上的串聯與設計，還是會感到複雜與困難。

而我最大的心得就是，未來在處理這樣會使用到許多block的邏輯電路設計時，一定要在一開始就設想好可能會發生的問題與解決方案，最重要的是，一定要有設計圖之類的去輔助思考，或者是在真正撰寫module時有個參考，這樣才能知道每個module需要多少input、output，甚至在一開始的宣告就能把那些線路是wire是reg都區分開來，雖然這樣的前處理會花上不少時間，但卻能大大省去之後debug的迷茫與未知。

**References**

7段顯示器的輸入接口查詢: (from Basys3 Reference Manual)

counter編撰方式: (from 第四週上課講義)

7段顯示器的更新方法: (from 第四週上課講義)