**Lab 4: Counters and Shift Registers II**

1 Construct a 4-bit synchronous binary down counter (b3b2b1b0) with the 1-Hz clock frequency from lab3 and use 4 LEDs to show the binary values.

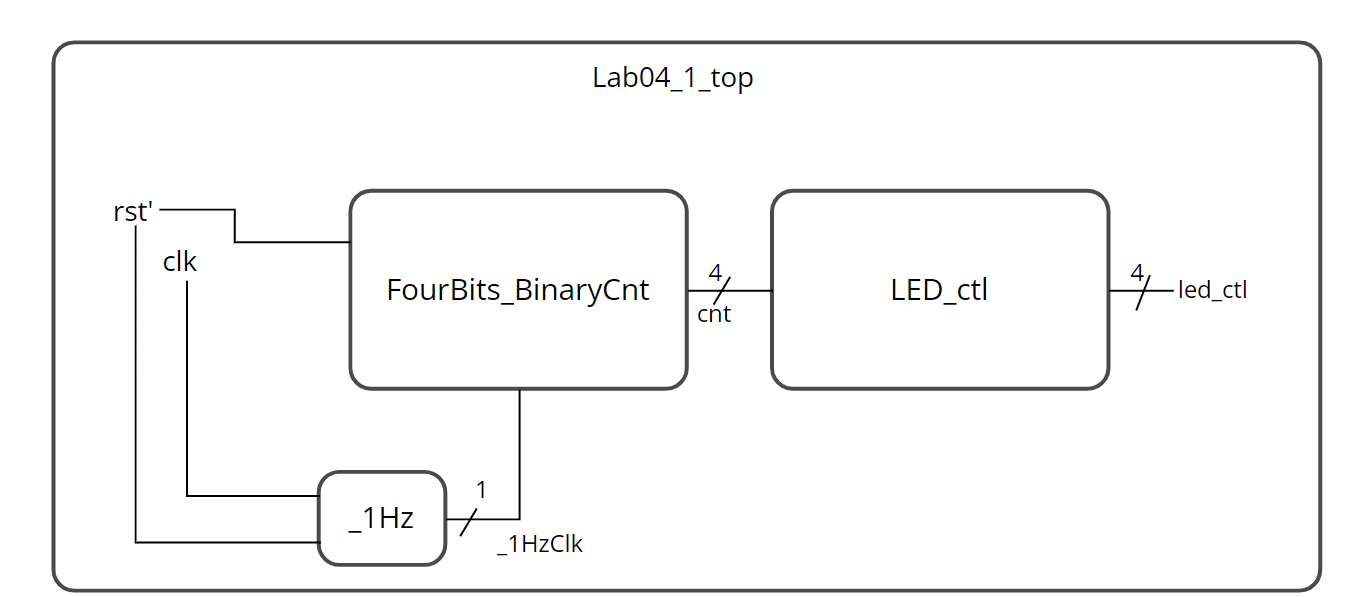
**Design Specification**

IO輸出入設定:

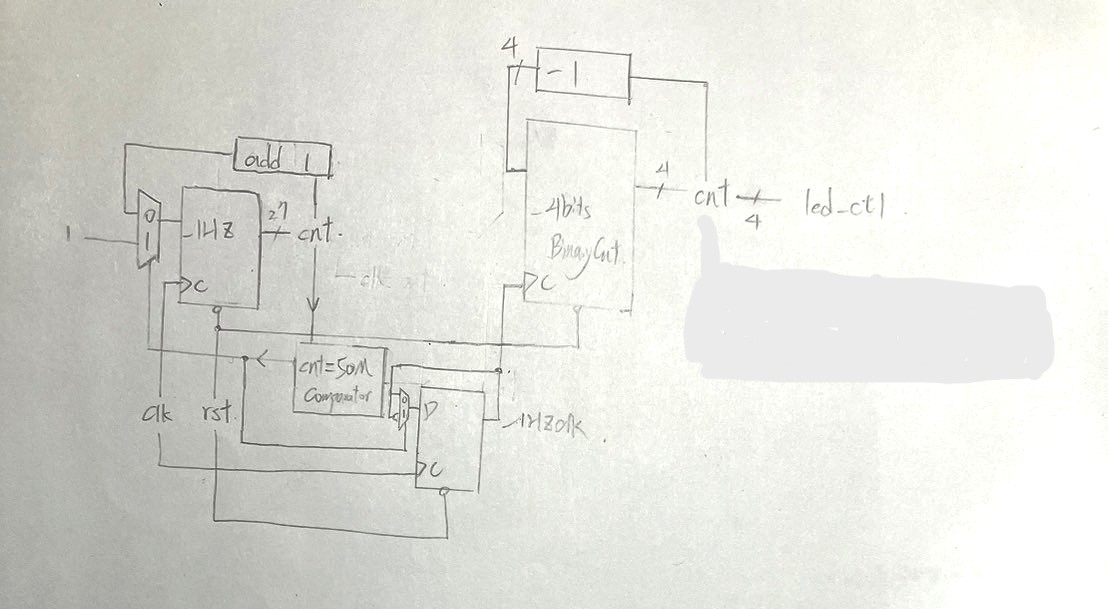
輸入: clk (1 bit), rst (1 bit)

輸出: [3:0]led\_ctl (4 bits)

Logic Block



**Design Implementation**



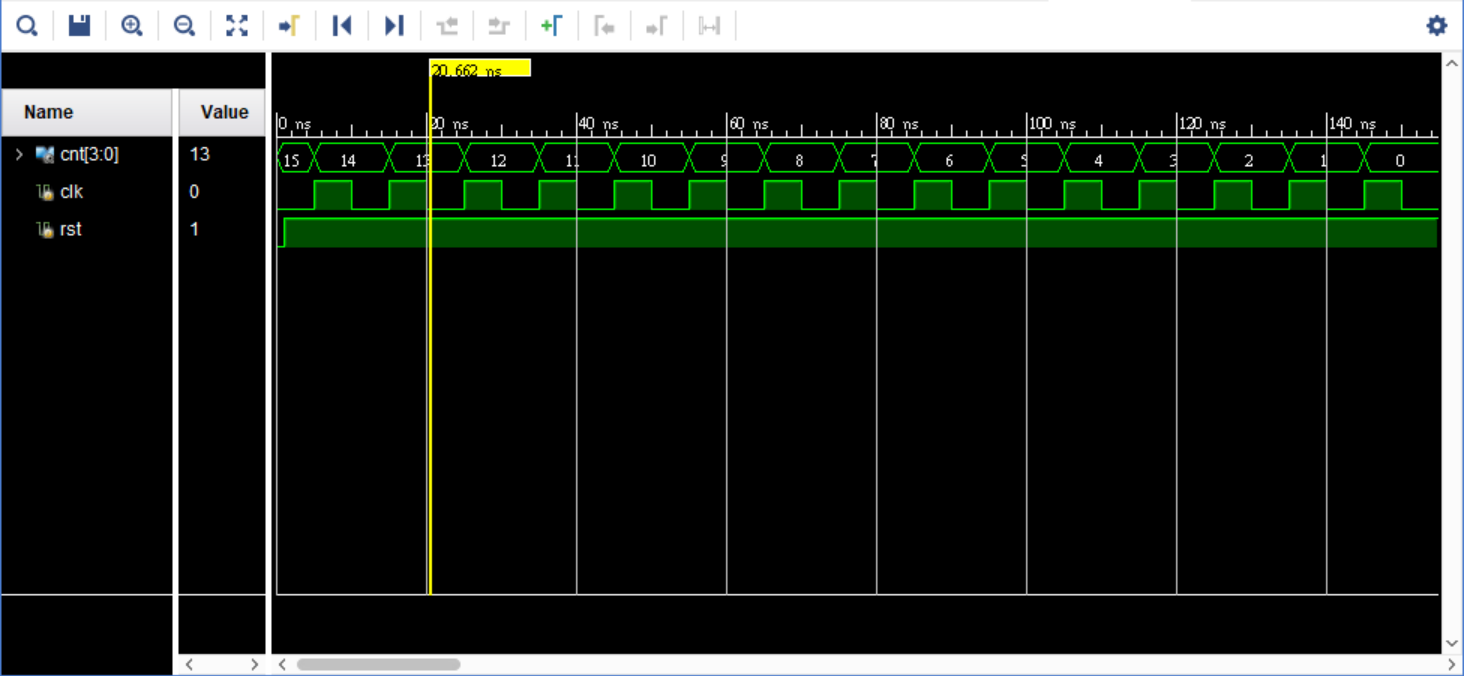
\_1HzClk是\_1Hz模組的訊號，藉由數到50M之後再除2得到，輸出1Hz時脈。

接著再把4bits BinaryCnt接上剛剛產生的時脈，進行4bits遞減。由於是遞減功能，所以初始值設為1111。最後把當前數的數字指定給led\_ctl，控制led燈號的輸出。

Pin assignment

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| IO | clk | rst | led\_ctl3 | led\_ctl2 | led\_ctl1 | led\_ctl0 |
| Pin | W5 | V17 | V19 | U19 | E19 | U16 |

**Discussion**

****

Testbench的呈現顯示出counter是正常運作的。至於\_1Hz和LED\_ctl的部分在實際執行結果沒什麼問題，和上次lab3的實驗幾乎一樣。唯有這次我把50M counter和最後的1bits counter合在一起寫。一開始沒注意到直接寫成{clk\_out, cnt} == 100M的判斷，雖然在實際跑的時候看不出來，但當我再次看程式碼時便覺得怪怪的，因為這樣跟實際的1Hz訊號還是有差，跟我們lab03\_1的27bits的除頻器類似，只是一個近似。所以我後來改用數到50M再接一位bit的除頻方法後便正確許多。

剩下的步驟和上次的Counter類似，只是變成往下數的。

**Conclusion**

在這個章節裡，我學到了

* Down counter的寫法

Down Counter 是一種計數器，從一個初始值開始遞減，並且在每個時鐘周期結束時進行遞減。與 Up Counter 類似，Down Counter 的功能也是非常常見的，特別是在需要倒計時或者逆序計數的應用中。

在 Verilog 中實現 Down Counter 與實現 Up Counter 類似，只需將遞增操作替換為遞減操作即可。通常情況下，Down Counter 的初始值設置為最大值，或是想設定的極限值，接著在進行減法操作即可。

**References**

<https://www.fpga4student.com/2017/03/verilog-code-for-counter-with-testbench.html>

在這個連結裡提到了如何寫Up-Down Counter的寫法，可以當作這兩次lab的總結。

2 Combine the 4-bit synchronous binary down counter from exp1 with a binary-to-seven segment-display decoder (from lab2) to show the binary counting in 7-segment display.

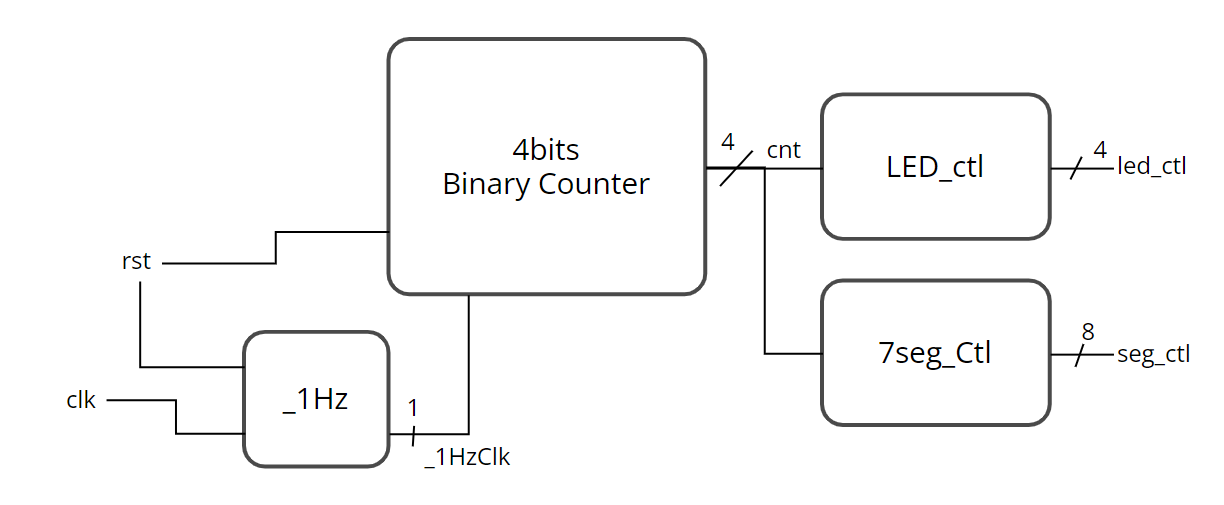
**Design Specification**

IO輸出入設定:

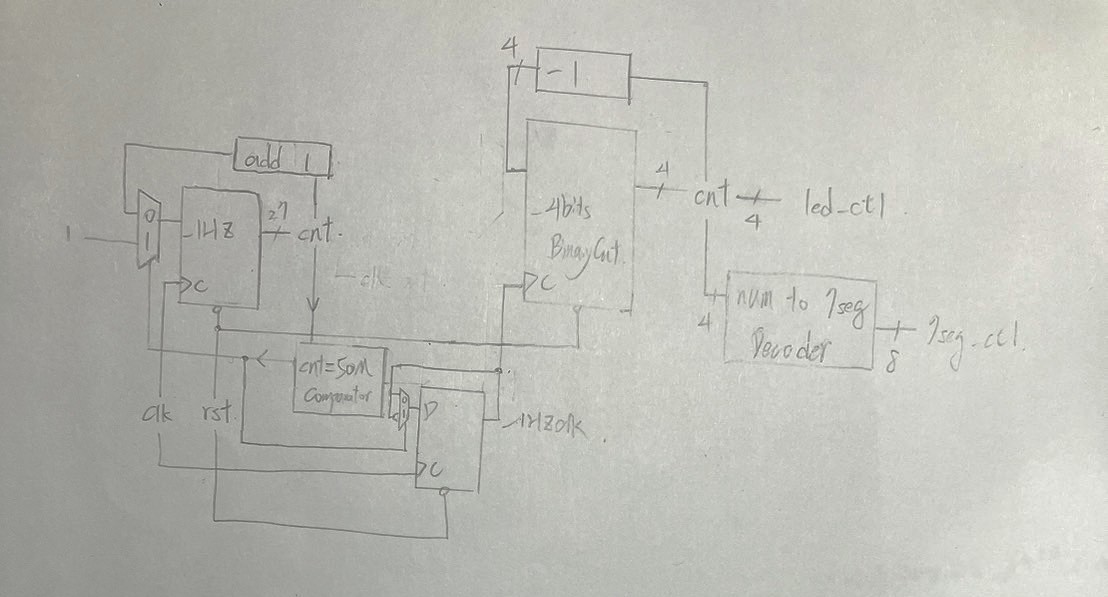
輸入: clk (1 bit), rst(1 bit)

輸出: [7:0]seg\_ctl(8 bits), [3:0]led\_ctl(4 bits)

Logic Block



**Design Implementation**

****

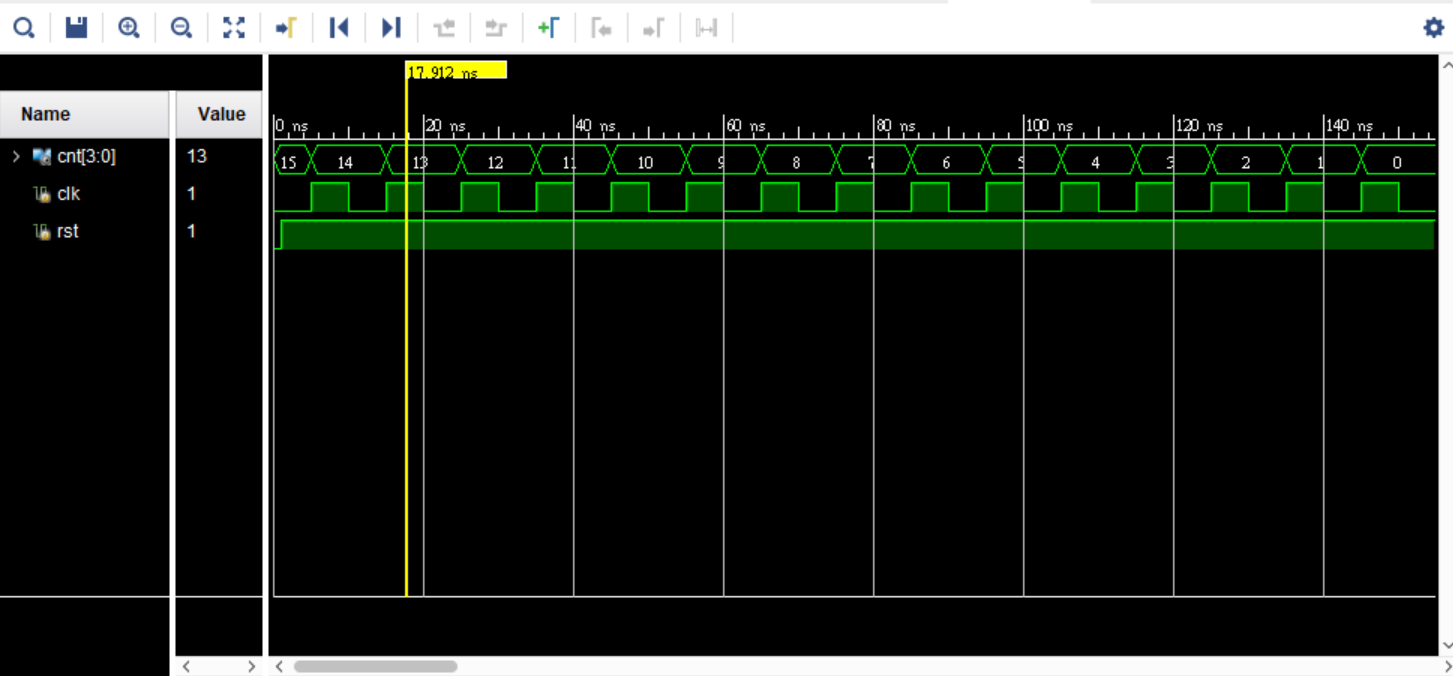
這個題目跟上一題幾乎一樣，只是多了用七段顯示器顯示，所以這裡的logic diagram在後面的部分多拉了一條線接到num to 7seg decoder，而這個decoder我寫在SevenSeg\_ctl裡面，用case去實作，default設為8’b11111110，所以當出現為之錯誤時，七段顯示器數字旁邊的點就會亮起以告訴我們出現錯誤(不過基本上不會出現此情況)。剩下的流程都一樣。

Pin assignment

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| IO | clk | rst | led\_ctl0 | led\_ctl1 | led\_ctl2 | led\_ctl3 | seg\_ctl0 |
| Pin | W5 | V17 | U16 | E19 | U19 | V19 | V7 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| IO | seg\_ctl1 | seg\_ctl2 | seg\_ctl3 | seg\_ctl4 | seg\_ctl5 | seg\_ctl6 | seg\_ctl7 |
| Pin | U7 | V5 | U5 | V8 | U8 | W6 | W7 |

**Discussion**



Testbench的Down counter一樣沒有問題。如上所述，這次只是多了一個七段顯示器的顯示，所以只需要檢查每個數字對應到的顯示是不是正確就好。而結果也經過我檢查無誤。這次用的decoder和lab2所用的方法幾乎一樣，只是變成我先define每個數字的顯示，並在case裡再呼叫這些define而已。另外，這次我沒有寫ssd控制，但結果仍運行正常，因此我認為若沒有特別去設定的話七段顯示器的ssd控制為0000。

**Conclusion**

在這個實驗裡，我學到了

* Down counter和七段顯示器的結合

透過這個實驗，我們可以將 Down Counter 的計數值與七段顯示器進行連接，從而在七段顯示器上顯示出計數的數字。這樣做有助於我們實現一個簡單的計數器或倒計時器，並將其顯示在七段顯示器上。

實現這樣的結合需要理解七段顯示器的工作原理，以及如何將 Down Counter 的計數值轉換為七段顯示器需要的控制信號。通常情況下，我們需要使用一個數字轉換表(decoder)來將每個數字對應到七段顯示器的控制信號，然後根據 Down Counter 的計數值選擇相應的控制信號。

**References**

<https://github.com/tnat93/Up-Down-Counter---7-segment-display>

這裡是我在網路上找到別人的寫法，有趣的是他用兩個seg輸出，不太清楚為何，但我猜測是使用的板子不同導致的

3 Construct a single digit BCD down counter with a 2-Hz clock as the clock frequency and display on the seven-segment display. 3.1 Construct a single digit BCD down counter. 3.2 Construct a BCD-to-seven-segment display decoder. 3.3 Combine the above two together.

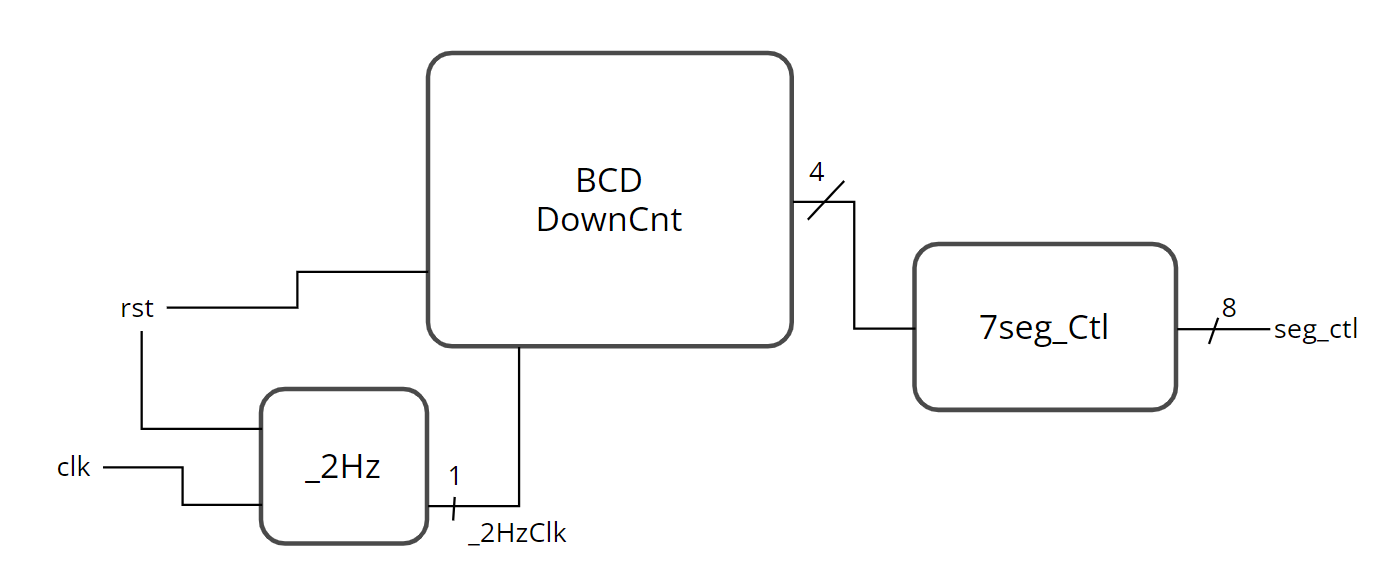
**Design Specification**

IO輸出入設定:

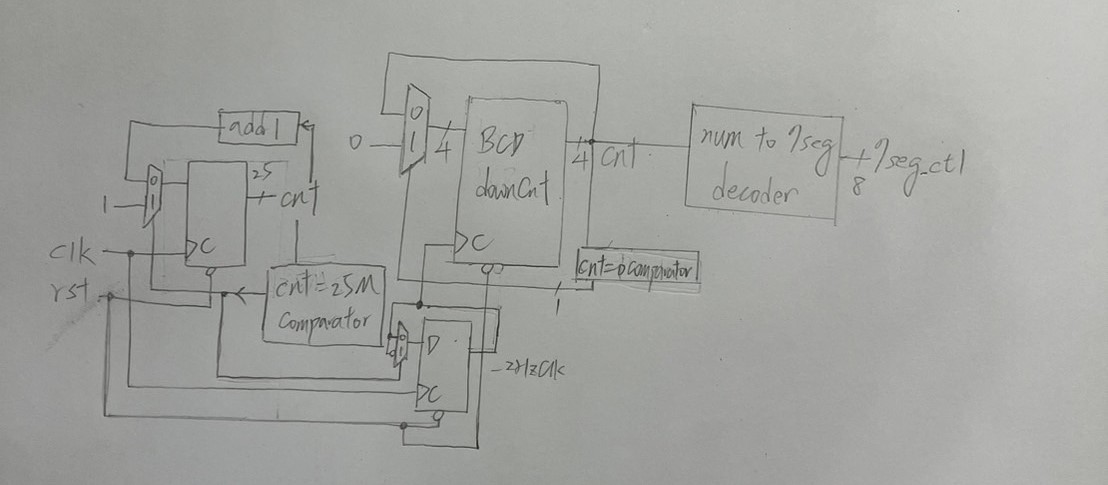
輸入: clk (1 bit), rst(1 bit)

輸出: seg\_ctl(8 bits)

Logic Block



**Design Implementation**

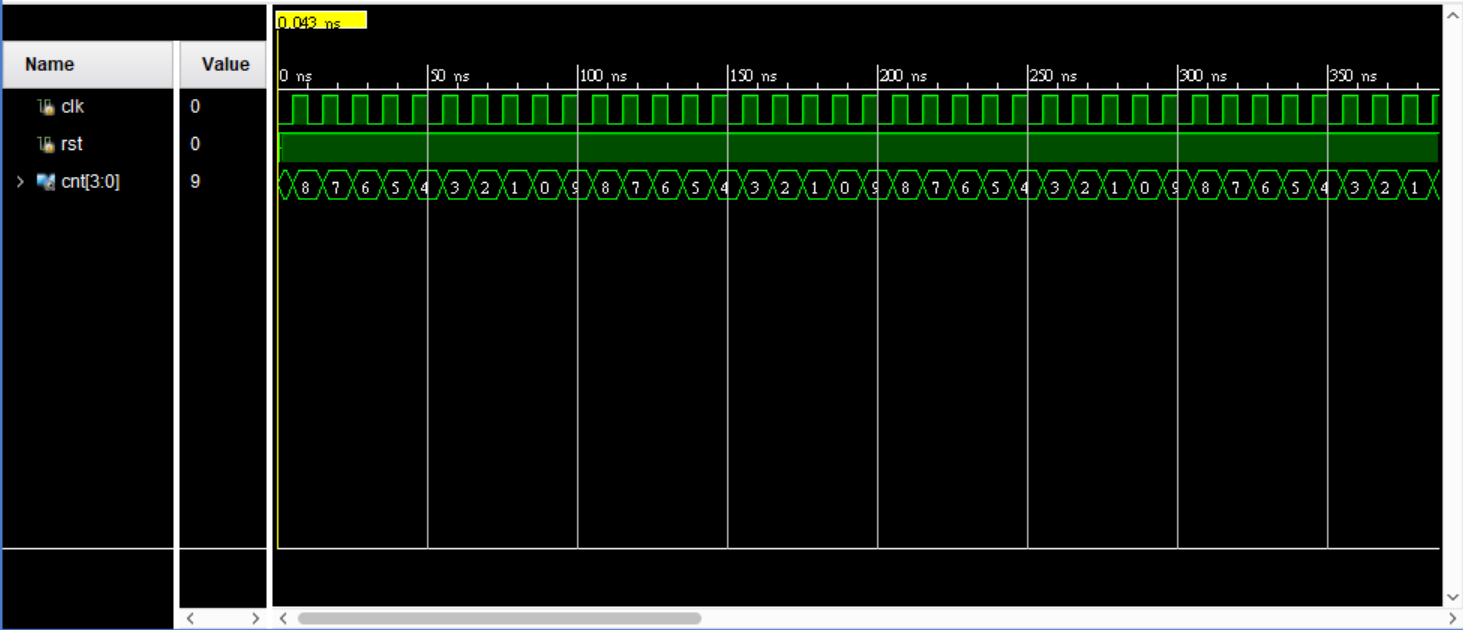


時間頻率由1Hz改為2Hz。因此我把原本數到50M加1bit的除頻器改為數到25M加1bit除頻。而BCD Down counter也跟原本的4 bits down counter相似，只是最大值從1111改為1001。後面的decoder就和原來一樣。

Pin assignment

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| IO | seg\_ctl0 | seg\_ctl1 | seg\_ctl2 | seg\_ctl3 | seg\_ctl4 | seg\_ctl5 | seg\_ctl6 | seg\_ctl7 |
| Pin | V7 | U7 | V5 | U5 | V8 | U8 | W6 | W7 |
| IO | clk | rst |  |  |  |  |  |  |
| Pin | W5 | V17 |  |  |  |  |  |  |

**Discussion**



從testbench中可以看出BCD Down Counter運作正常。實際操作過程中可以明顯地看出頻率快很多，用手機測時間後跑完一個週期大概五秒，跟理論值很接近。BCD to 7Seg的decoder後來我把原本的A~F的設定也刪掉了，因為功能上這些設定不應該出現。同樣地我也沒有去設定ssd的值，所以四個燈都會亮。

在\_2Hz的時脈設定，我原本打算直接數50M作為輸出，但後來不管怎麼用他的時脈永遠都不變，所以就改成25M+1bit除頻。不過之後再經過上網查詢方法後，便知道原來若要用直接數的方法來除頻，那麼最後的時脈輸出也要寫在flipflop裡，這樣出來的結果才是對的。

**Conclusion**

在這個小題裡，我學到了

* BCD down counter
* 2Hz 的時脈產生方法

經過多次的Counter和時脈產生的練習。我愈來愈對這兩個東西感到熟悉，並逐漸掌握更多相關的寫法，這對於許多設計項目都是非常有用的。

**References**

無

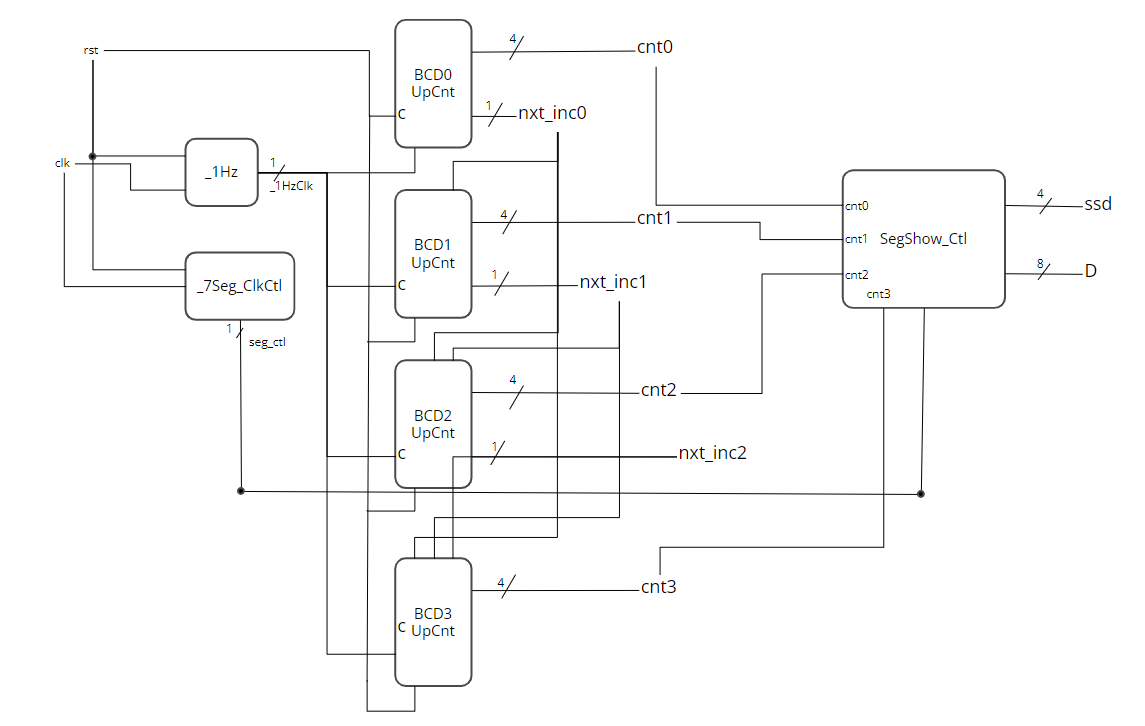
4 Construct a 2-digit BCD up counter with a 1-Hz clock as the clock frequency and display on the seven-segment display.

**Design Specification**

IO輸出入設定:

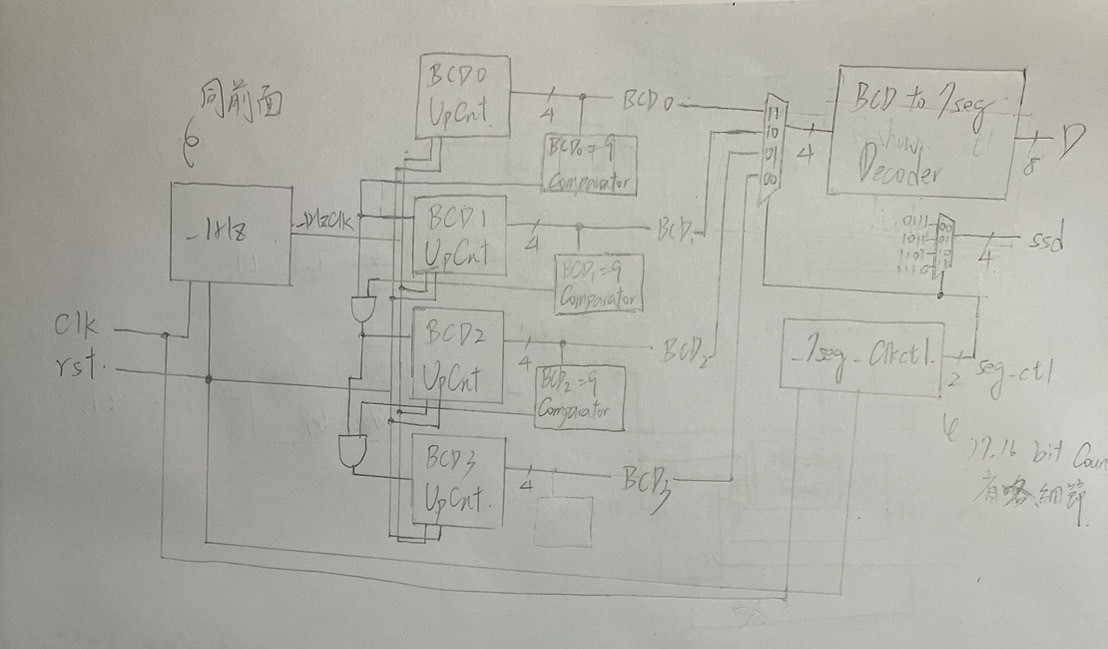
輸入: clk(1 bit), rst(1 bit)

輸出: [3:0]ssd(4 bits), [7:0]D(8 bits)



**Design Implementation**

**Logic diagram**



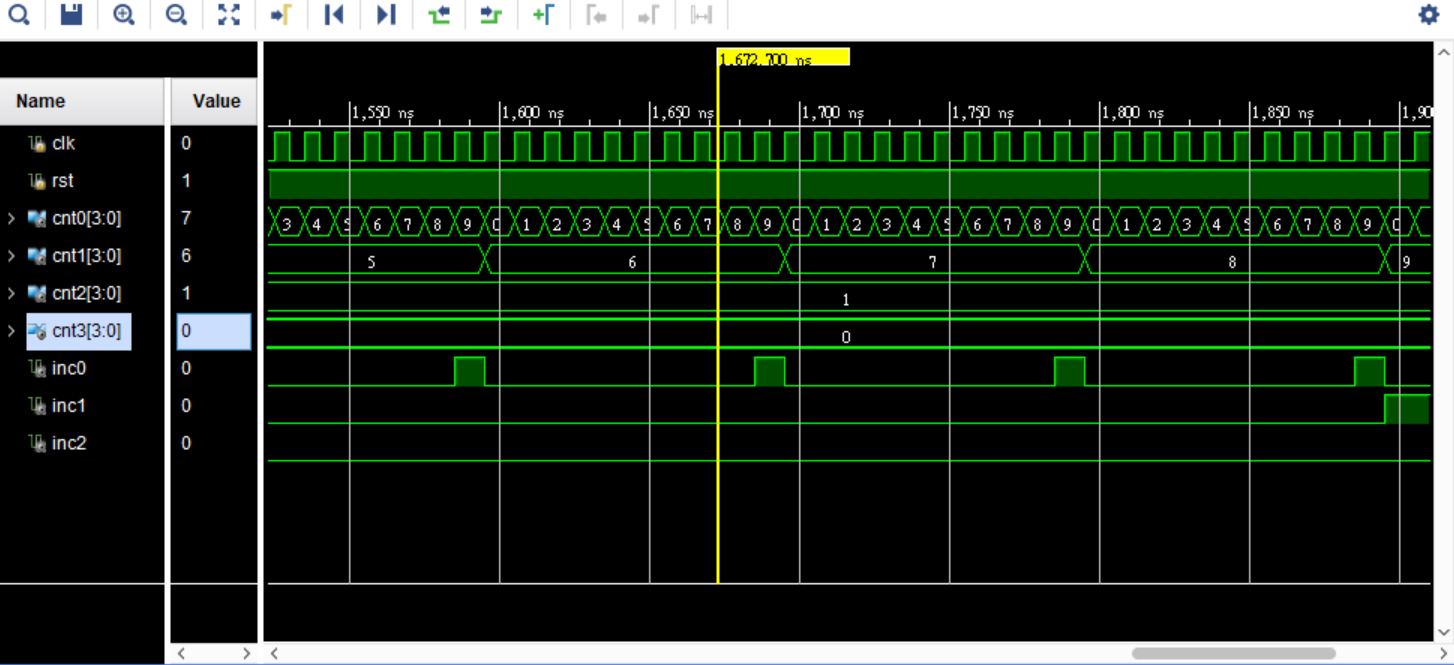
\_1Hz和先前一樣輸出1Hz的時脈。而seg\_ctl是用於控制ssd的時脈(17bits的除頻器)，來決定七段顯示器中四個燈變換的頻率。接著是BCD0 UpCnt~BCD3 UpCnt的配置。BCD0會輸出cnt0代表當前數到的十進位數字，也另外輸出inc0(當cnt0=9時)來控制下一位的變動。而BCD1和BCD0相似，不過多了一個多工器控制當前狀態，其規則如下。

|  |  |
| --- | --- |
| State | func |
| Inc0 = 0 | No change |
| Inc0 = 1, cnt !=9 | Add 1 |
| Inc0 = 1, cnt = 9 | Reset 0 |

一樣，當cnt1=9時，inc1設為1以控制下一位cnt2的變動。依此類推，後面的位數在考慮進位時都要是inc0~inc(n-1)皆為1時才能加一。

最後，SegShow\_Ctl依照seg\_ctl的狀態，決定要輸出哪一位，並利用ssd高頻的控制來達成一次顯示愈個不同數字。

**Discussion**



上面是四個BCD0~BCD3的TB輸出。可以看到所有位數都正常執行，沒有跑出錯誤狀況。不過要數到1000有點久，我就沒放上來了。實際執行的時候也一樣，都正確顯示。在實作過程中，我發現老師ppt裡提供的seg\_ctl方法確實比較好，相比我上次lab2\_5自己設定的頻率，這次的頻率使得七段顯示器顯示得比較自然，不會有一閃一閃的情況。另外一個可以討論的點是進位時機的判斷，我目前的寫法是用always block而裡，是在當cnt=9時設為1，其餘設為0。不過我後來想到應該是要寫在flipflop裡才對，跟cnt重設為0時同時進行。但經過我多次測試後我發現這兩種寫法都可以，Testbench裡也可以看出我這樣的寫法沒問題，inc訊號都有準時的跳出來，也準時地跳回去。

**Conclusion**

在這個實驗裡，我學到了

* BCD Down Counter的寫法
* 複習了七段顯示器一次顯示四個不同數字的寫法

這種四位的BCD Down Counter讓我想到上學期一開始學的4 bits syn. Counter，因為是同步的，所以後一位的都需要前一位的變成1時才能進位。但因為verilog可以直接進行4bits的操作，所以寫程式時可以省略這個步驟。但到了BCD時就又要做這個動作了。

另外就是ssd的頻率控制，上次lab2\_5我用19bits除頻器來寫，所以看七段顯示器時都會覺得他一閃一閃的。這次減少除頻次數，使得觀察七段顯示器時便自然多了。另外跟上次不同的點是，上次是用ring counter來控制ssd，這次用兩位bits來控制ssd，又讓我學會了新的寫法。

**References**

<https://github.com/newajsharif91/Verilog_HDL_Digital-System-Design/tree/main/Counter>

Counter有很多種，除了有序的Up/Down counter，也可以依照自己設定的序列去數。這個連結展示了多種counter，也示範了原始的ripple carry counter如何實作。