**Lab 03**

105060016 謝承儒

1. **Lab03-1 : Frequency Divider(原本的倍)**
2. **Design Specification :**
3. Input : f\_crystal //輸入的頻率(100MHz)

Rst //當=1時，使板子暫停運作

1. Output : f\_out //頻率為f\_crystal的1/227倍(接近1Hz)
2. Wire : [26:0] cnt

[26:0] cnt\_tmp //作為一個27-bits的counter

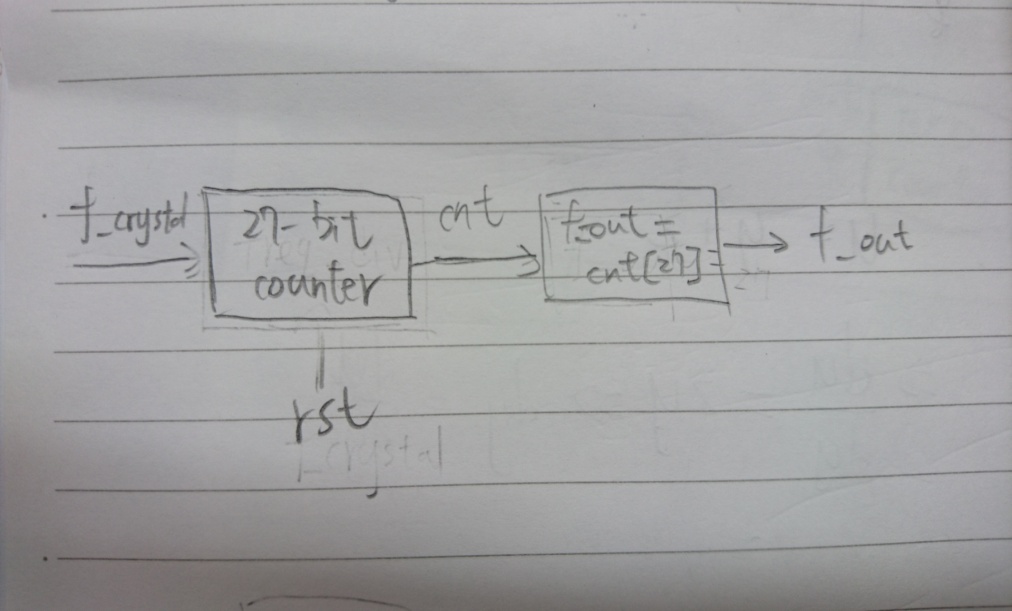


圖1 Lab03-1的區塊圖

1. **Design Implementation :**
2. Logic function :

cnt\_tmp = cnt + 1

f\_out = cnt [27]

1. Logic diagram :

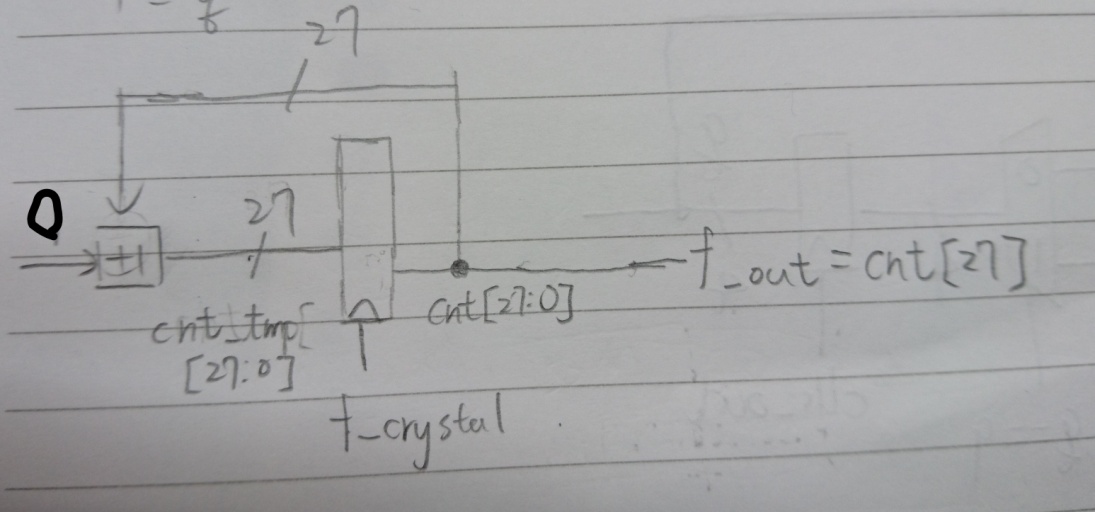


圖2 Lab03-1的邏輯圖

1. Pin assignment :
2. Input :
3. f\_crystal = W5
4. rst = U16
5. Output :
6. f\_out = V17
7. **Discussion :**
8. 思考過程 :

這題是要將輸入的頻率(f\_crystal)降為原先的倍再將其輸出(f\_out)。

我選擇使用一個27-bit的counter(cnt)，只要每次f\_crystal從0變為1

就使cnt加1，而cnt的最高位(cnt[27])在f\_crystal變化226次便會變化，在下一個226次後又回變回來，如此cnt[27]的變化頻率即f\_crystal的倍。因此，cnt[27]便是我們所求的f\_out。

1. 過程中的Bug :

起初並不曉得板子上clk的接腳是W5，而隨便接上一個Switche，導致程式無法編譯。之後回頭看了講義就解決了。

1. **Lab03-2 : Frequency Divider(輸出為1Hz)**
2. **Design Specification :**
   1. Input : f\_crystal //輸入的頻率(100MHz)

rst //當=1時，使板子暫停運作

* 1. Output : f\_out //輸出的頻率(1Hz)
  2. Wire : [26:0] cnt //27-bit的counter裡的值

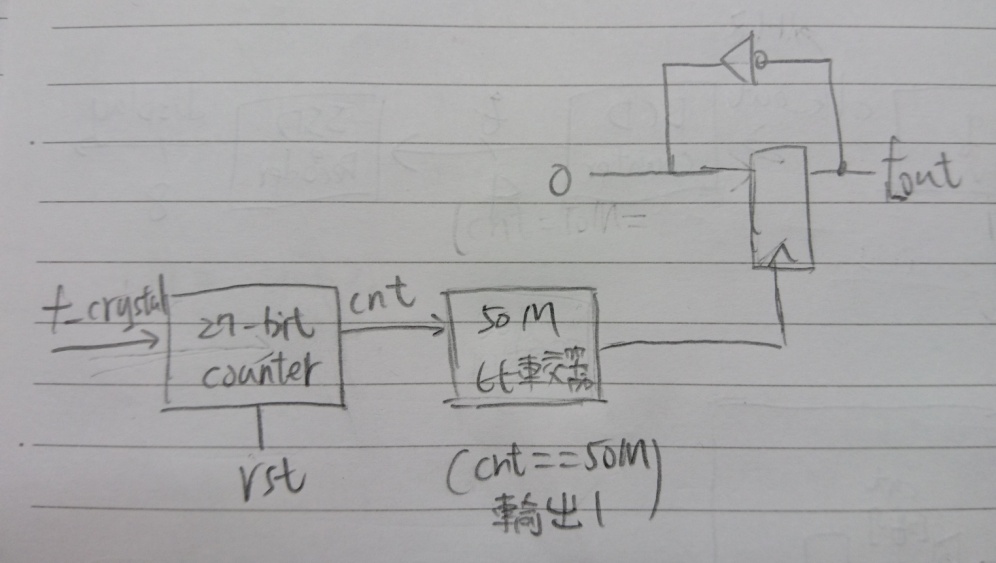


圖3 Lab03-2的區塊圖

1. **Design Implementation :**
   1. Logic diagram :

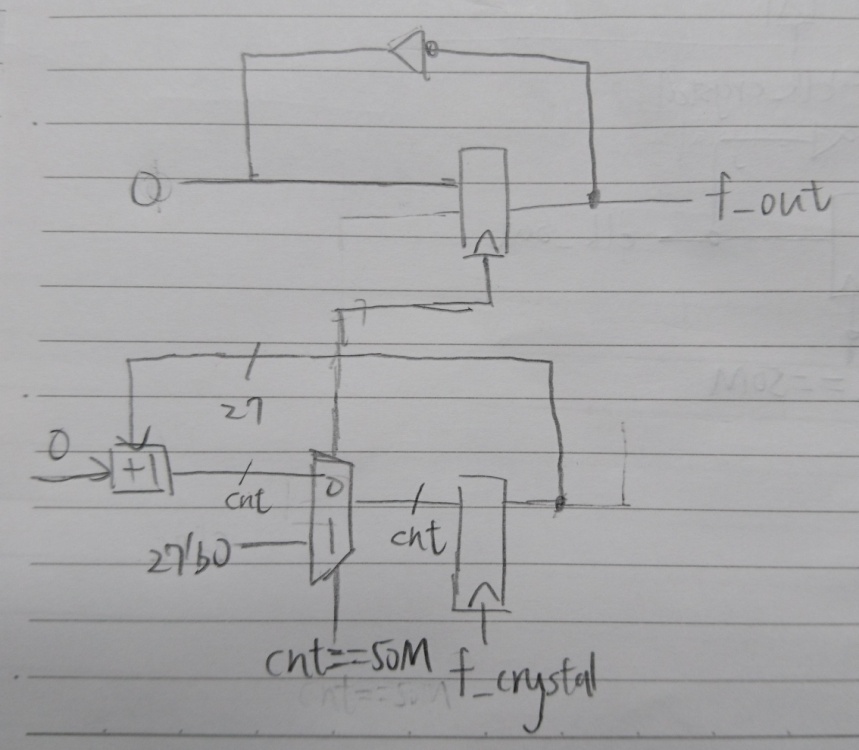


圖4 Lab03-2 的邏輯圖

* 1. Pin assignment :
     1. Input :
        1. f\_crystal : W5
        2. rst : U16
     2. Output :
        1. f\_out : V17

1. **Discussion :**
   1. 思考過程 :

這次要輸出的頻率(f\_out)為1Hz，因為100M沒辦用2的次方達成，沒辦法使用上一個實驗的方法。

因此，改為利用27-bit的counter(cnt)，每個f\_crystal的週期都使cnt加1。當cnt數到50M時，便讓f\_out = ~fout且cnt歸零，如此f\_crystal跑完100M個週期後，f\_out也跑出1個週期，達到1Hz的效果。

* 1. 過程中的Bug :

本來將除頻器和counter兩者的always區塊分開來寫，不過卻編譯錯誤，同學說code裡的變數盡量不要在兩個不同的always裡存值，之後就把兩個always合成一個來寫。雖然之後可以編譯，不過code整體變得十分難看懂。

1. **Lab03-3 : 1Hz的BCD Counter**
2. **Design Specification :**
3. Input : clk\_crystal //板子輸入的頻率(100MHz)

rst //讓板子暫停運作

1. Output : [7:0] display //七段顯示器的顯示碼
2. Wire : clk\_out //用除頻器得到的1Hz clk

[26:0] cnt //27-bit的counter裡的值

[3:0] q //BCD counter裡的值

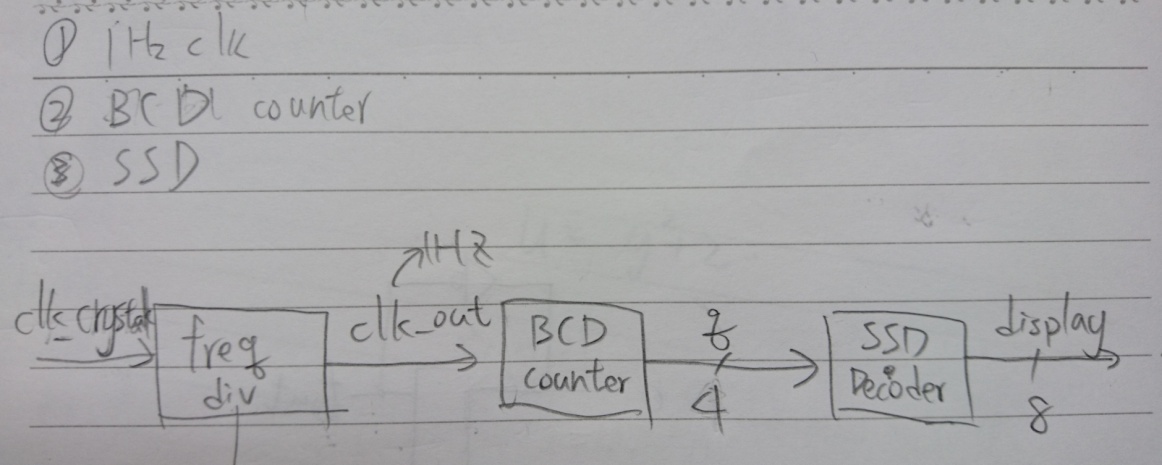


圖5 Lab03-3的區塊圖

1. **Design Implementation :**
2. Logic function :
   * 1. Output :
        1. display[0] = 1
        2. display[1] = q3’q2’q1’ + q3’q2q1q0’
        3. display[2] = q3’q2’q0 + q3’q2’q1 + q3’q1q0
        4. display[3] = q3q2q1’ + q3’q0
        5. display[4] = q3q2 + q3q1 + q2’q1’q0 + q2q1’q0’ + q2q1q0
        6. display[5] = q3q2 + q3q0 + q2’q1q0’
        7. display[6] = q3q2+ q3q1 + q3q0+ q2q1’q0 + q2q1q0’
        8. display[7] = q3’q2’q1’q0 + q3’q2q1’q0’
3. Logic diagram :

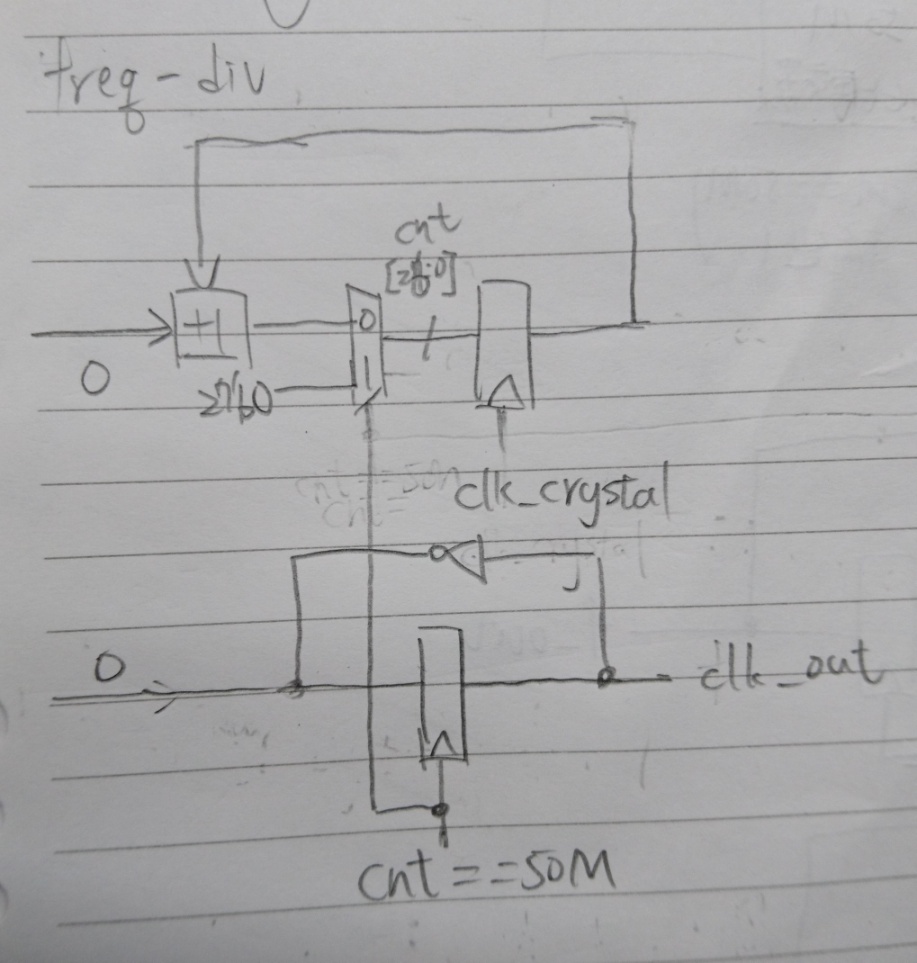


圖6 除頻器的邏輯圖

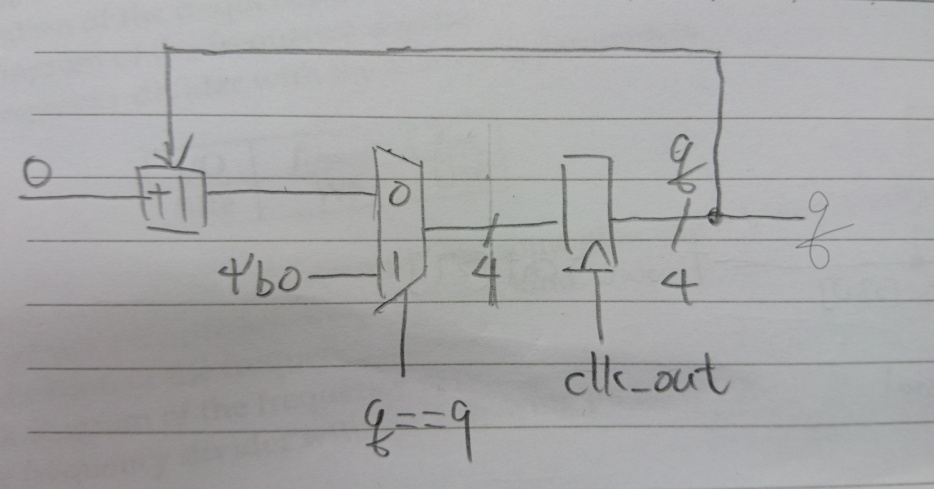


圖7 BCD Counter的邏輯圖

1. Pin assignment :
   * 1. Input :
        1. clk\_crystal = W5
        2. rst = V17
     2. Output :
        1. display[0] = V7
        2. display[1] = U7
        3. display[2] = V5
        4. display[3] = U5
        5. display[4] = V8
        6. display[5] = U8
        7. display[6] = W6
        8. display[7] = W7
2. **Discussion :**
3. 思考過程 :

這次要做的是頻率為1Hz的BCD counter。因此，我們需要三個東西。

1. 1Hz的clk

這可以藉由Lab03-2的實驗結果來達成。用輸入的clk(clk\_crystal)來驅動27-bit的Counter(其值為[26:0]cnt)。在cnt=50M時，讓clk\_out = ~clk\_out且cnt歸零，達到1Hz的效果。

1. BCD counter

將剛剛1Hz的clk\_out輸入，來讓BCD Counter(其值為[3:0]q)運轉。當q=9時便將其歸零，反之則加1，這效果可以利用多工器來達到，code裡則直接使用if即可。

1. 七段顯示器的Decoder

把q輸入進來，經過Decoder解碼成8-bit的顯示碼([7:0]display)，可以利用畫truth table、K-MAP來取得其function，在code裡可以使用case來達到同樣的效果。

1. 遇到的Bug :

起初打開rst時，產生的效果並不是歸零，而是暫停。檢查code後發現當rst打開時，只有停止除頻器運作，也就是暫停clk\_out。除此之外應該還要將q也設為0，在BCD Counter那裏的code改完後便能正常運作了。

1. **Conclusion :**

這次實驗結合前兩次實驗的結果，不過還是遇到很多小bug，耗了很多時間。但看到自己做出來的結果越來越複雜也很有成就感。

1. **Reference :**
   1. 老師給的實驗講義

讓我知道code怎麼打和辨別不同元件的接腳。

* 1. 上學期的邏輯設計講義

讓我知道Counter和除頻器的原理。