微算機系統

實驗三

組別： 14
班級、姓名與學號：四資二　洪晟毅　104590048
四資二　梁皓鈞　104360098

日期： 2016.11.14

1. 實驗內容：

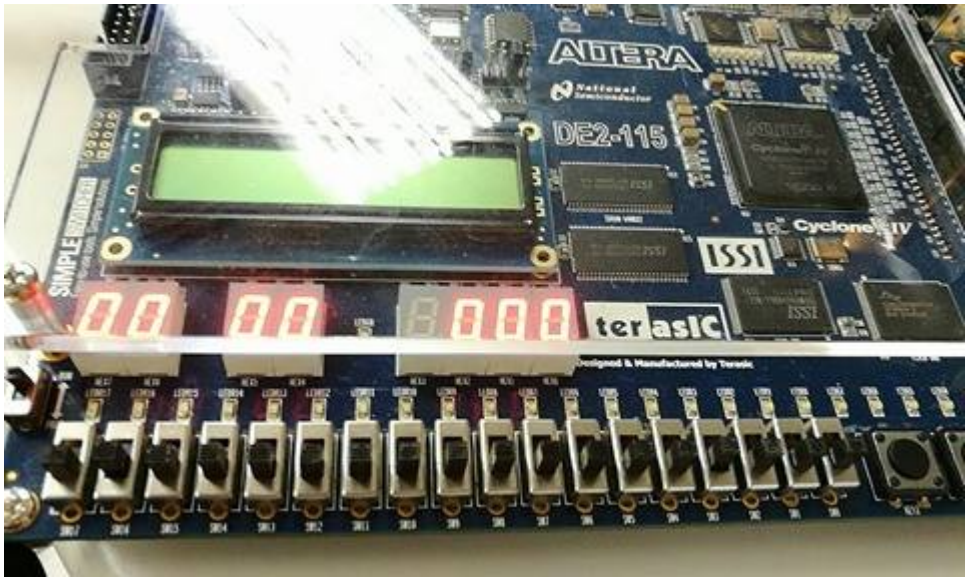
以前兩次實驗的成果作為基礎，透過層遞式設計的以及元件的概念，把實驗一的7-segment 實驗以及實驗二的 Full-Adder 實驗包裝成 Component 並且使用在這一次的 BCD 加法器中。

2. 實驗過程及結果：

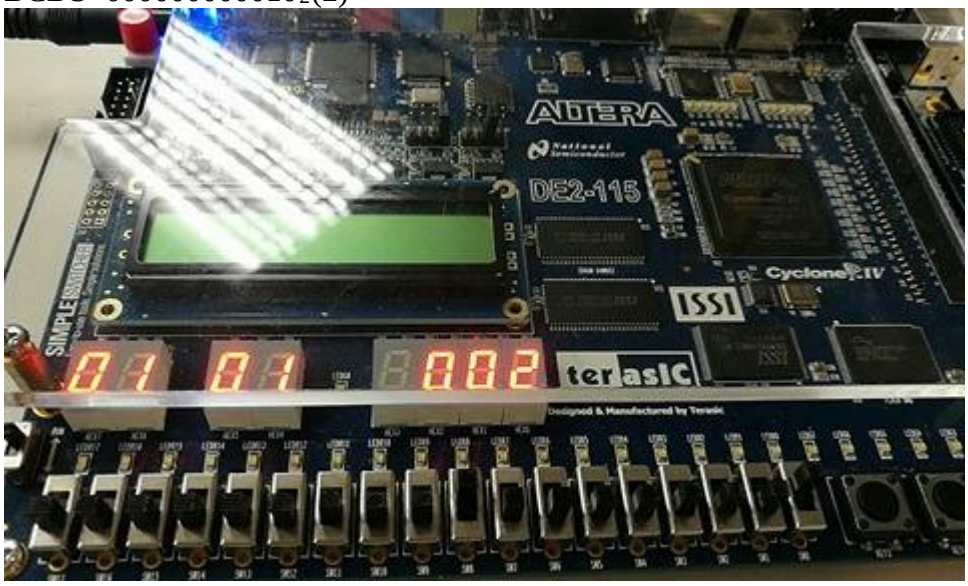
由於梁皓鈞對於 Logic Gate 的概念雖有但欠缺熟悉程度。因此這次實驗的程式碼編寫上主要由洪晟毅作主導，帶領著梁皓鈞去理解 Logic Gate 在這次實驗的用法，示範一次之後交給梁皓鈞嘗試去做其他部份例如包裝 Component。

以下的圖片我們直接以加分題的項目進行示範
註 : ENABLE 控制做加法還是減法，0就是做加法，1就是做減法

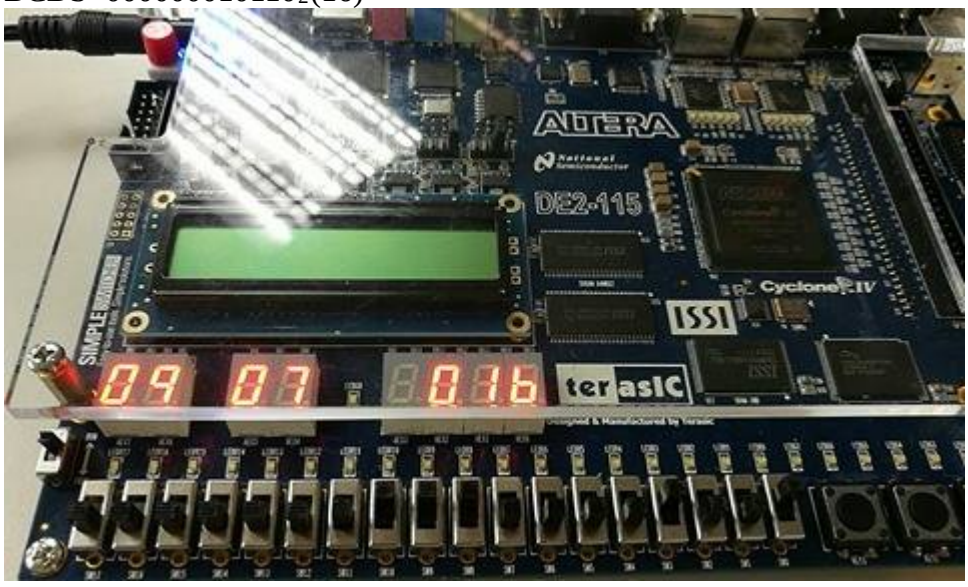
a. 輸入 X=00000000_2，Y=00000000_2，ENABLE=0，可以正確得到 BCDS=000000000000_2



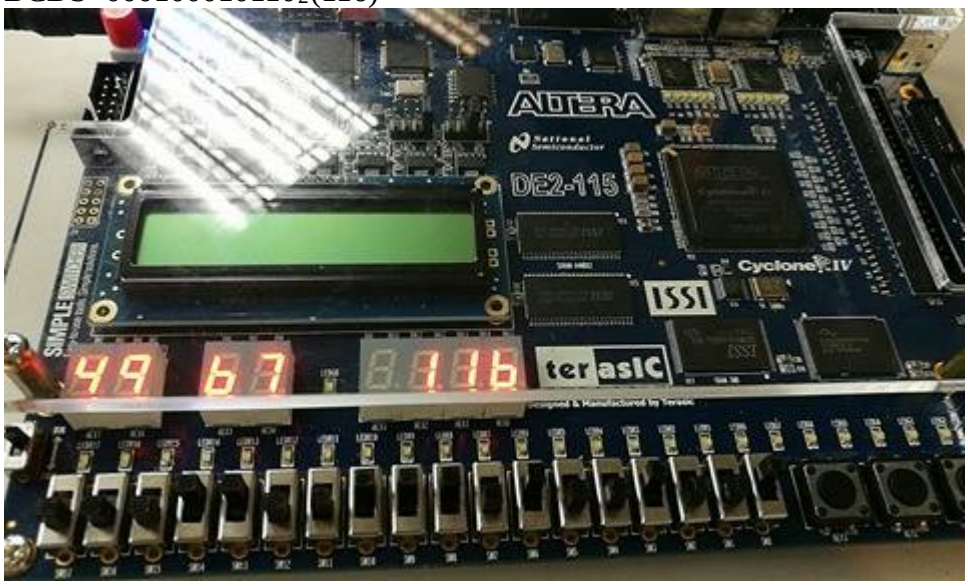
b. 輸入 X=00000001_2，Y=00000001_2，ENABLE=0，可以正確得到 BCDS=000000000010_2(2)



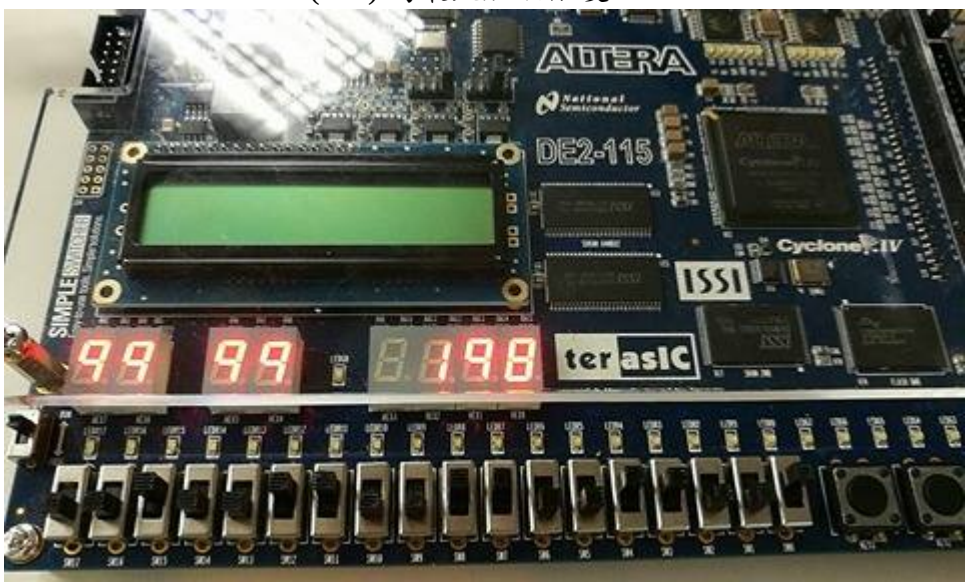
c. 輸入 X=00001001_2，Y=00000111_2，ENABLE=0，可以正確得到 BCDS=000000010110_2(16)



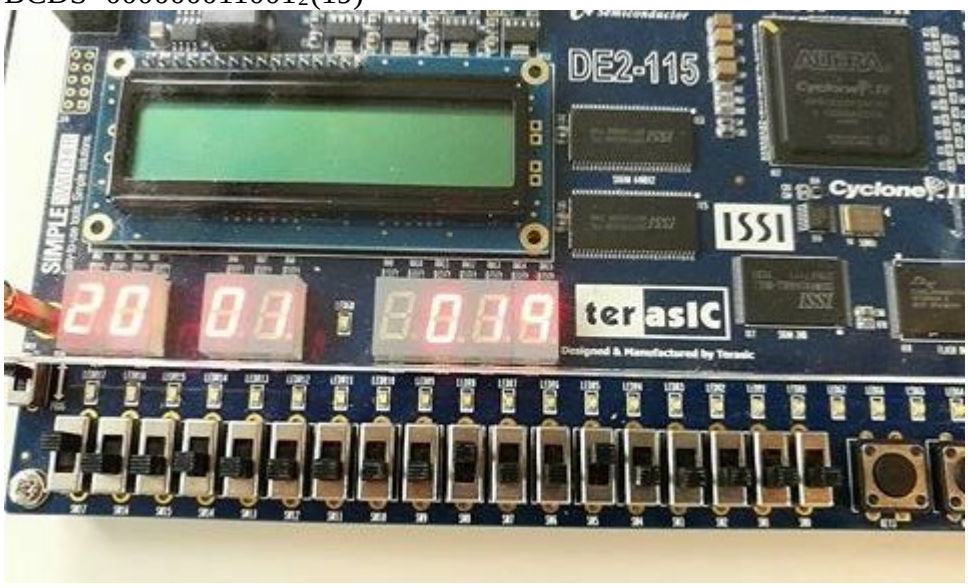
d. 輸入 X=01001001_2，Y=01100111_2，ENABLE=0，可以正確得到 BCDS=000100010110_2(116)



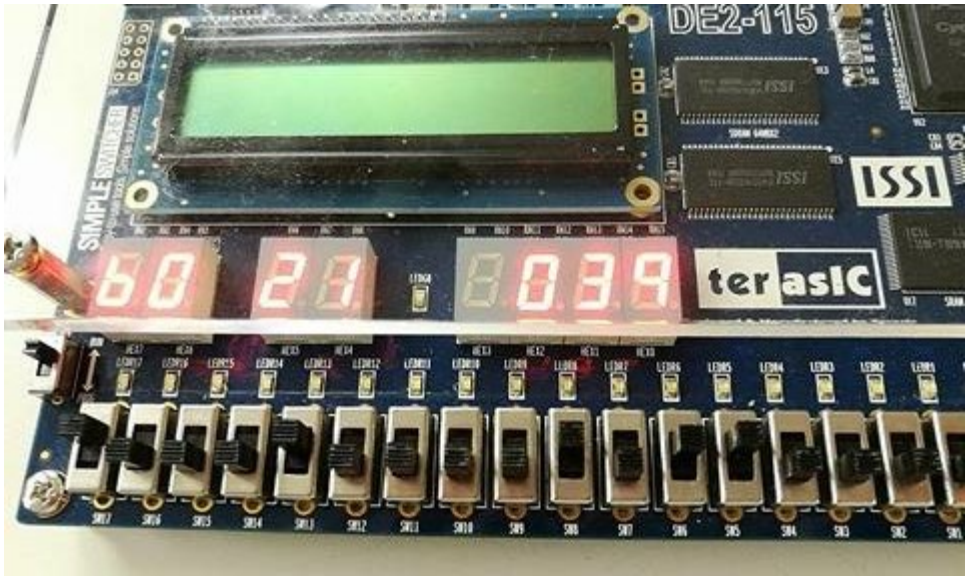
e. 輸入 X=10011001_2，Y=10011001_2，ENABLE=0，可以正確得到 BCDS=000110011000_2(198) 為最大加法限度



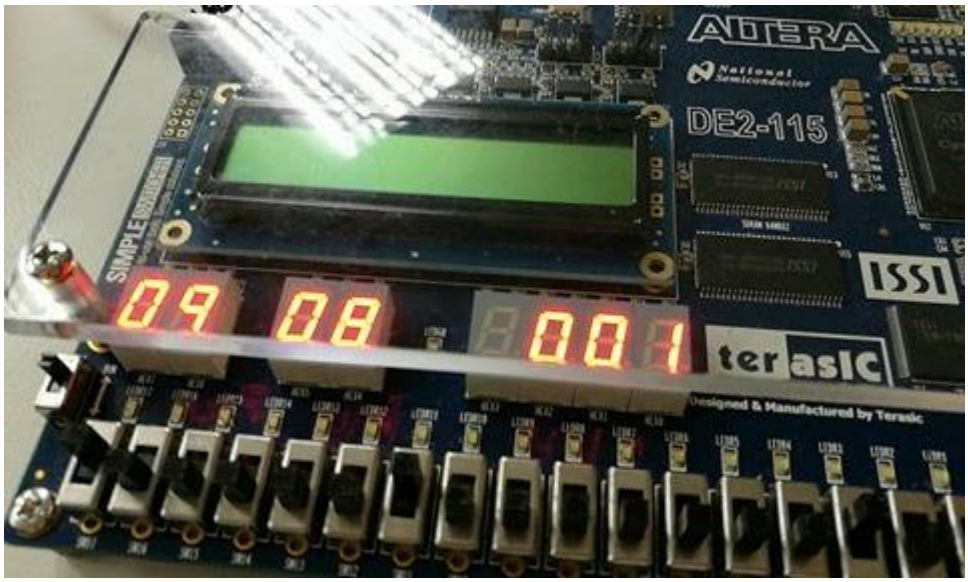
f. 輸入 X=00100000_2，Y=00000001_2，ENABLE=1，可以正確得到 BCDS=000000011001_2(19)



g. 輸入 X=01100000_2，Y=00100001，ENABLE=1，可以正確得到 BCDS=000000111001(39)



h. 輸入 X=00001001，Y=00001000，ENABLE=1，可以正確得到 BCDS=000000000001(1)



程式碼解釋：

7-Segment、FullAdder、FourBitsAdder 在之前的 Project 都已解釋過，故在此僅解釋 BCDAdder 及主電路的部分。

BCDAdder：

```
ENTITY BCDAdder IS
    PORT(
        switch : IN STD_LOGIC;

        x : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
        y : IN STD_LOGIC_VECTOR(3 DOWNTO 0);

        sum : OUT STD_LOGIC_VECTOR(3 DOWNTO 0);

        carry_in : IN STD_LOGIC;
        carry_out : OUT STD_LOGIC;

        sign : OUT STD_LOGIC
    );
END BCDAdder;
```

BCDAdder 的 Entity 設計共有以下幾個 Ports：

Port	Type	Description
switch	STD_LOGIC	切換加減法功能
x	STD_LOGIC_VECTOR(3 DOWNTO 0)	被加(減)數
y	STD_LOGIC_VECTOR(3 DOWNTO 0)	加(減)數
sum	STD_LOGIC_VECTOR(3 DOWNTO 0)	輸出總和
carry_in	STD_LOGIC	進位輸入
carry_out	STD_LOGIC	進位輸出
sigh	STD_LOGIC	輸出結果的正負號

Step 1. 首先直接將輸入的 x 及 y 經過 FourBitsAdder 相加(減)。

```
    add0 : FourBitsAdder PORT MAP (
        x => x,
        y => y,
        sum => sum_temp,
        carry_in => carry_in,
        carry_out => carry_out_temp,
        sign => sub_borrow,
        switch => switch
    );
```

Step 2. 如果是加法，

因為 BCD overflow(即大於9)的關係，必須將結果以+6的方式補回。

```
42      bcd_overflow <= (
43                          NOT switch
44                      ) AND (
45                          carry_out_temp OR (
46                              sum_temp(3) AND (sum_temp(1) OR sum_temp(2))
47                          )
48                      );
49      carry_out <= (NOT switch) AND bcd_overflow;
50
51      handle_bcd_overflow : FourBitsAdder PORT MAP (
52          x => sum_temp,
53          y => '0' & ('1' AND bcd_overflow) & ('1' AND bcd_overflow) & '0',
54          sum => sum_temp2,
55          carry_in => '0',
56          switch => '0'
57      );
58
```

Step 3. 如果是減法，

則必須將負數(2的補數)轉回正數，並以10相減、輸出 sign 結果。

```
60      nagative <= (switch AND sub_borrow);
61
62      do_two_complement : FourBitsAdder PORT MAP (
63          x => '0' & '0' & '0' & '0',
64          y => sum_temp2,
65          sum => sum_temp3,
66          carry_in => '0',
67          switch => nagative
68      );
69
70      sub_by_ten : FourBitsAdder PORT MAP (
71          x =>('1' AND nagative) & '0' & ('1' AND nagative) & '0',
72          y => sum_temp3,
73          sum => sum,
74          carry_in => '0',
75          switch => ('1' AND nagative)
76      );
77
78      sign <= switch AND nagative;
```

Lab3 主程式：

主要電路的部分，只要分別將個位數、十位數串接進 BCDAdder 的 Component 進行計算即可，僅需注意若個位數得出 sign=1則表示個位數相減結果為負數，須有借位的情況發生，此時要再將十位數的結果再減1。

```
25      add_bcd0 : BCDAdder PORT MAP (
26          x => x(3 DOWNTO 0),
27          y => y(3 DOWNTO 0),
28          sum => bcd0_sum,
29          carry_in => '0',
30          carry_out => bcd0_carry,
31          sign => bcd0_borrow,
32            switch => switch
33      );
34
35      add_bcd1 : BCDAdder PORT MAP (
36          x => x(7 DOWNTO 4),
37          y => y(7 DOWNTO 4),
38          sum => bcd1_sum_buffer,
39          carry_in => bcd0_carry,
40          carry_out => bcd1_carry,
41            switch => switch
42      );
43
44      sub_borrow_bcd : BCDAdder PORT MAP (
45          x => bcd1_sum_buffer,
46          y => ('0' & '0' & '0' & ('1' AND bcd0_borrow)),
47          sum => bcd1_sum,
48          carry_in => '0',
49          switch => ('1' AND bcd0_borrow)
50      );
```

3. 實驗心得：

梁皓鈞(104360098)：

這一次實驗是最後一次需要用到 Logic Gates 的實驗。這一次用到很多 Boolean Expression, 而這個也是我的弱項，因此我無法很順利的做出來，幸好有晟毅的幫助。晟毅首先先告訴我大綱，整個程式要怎樣思考，然後示範一次包 Package 跟 Component 給我看。接下來讓我嘗試把其餘的元件包裝起來。

在所有的元件包裝起來之後，晟毅示範給我看接線，最後由我學起來並且把其餘元件也接線接起來。但在 BCD 加法減法部份我真的比較弱，對於數位邏輯的熟悉度不夠，因此主要由晟毅主導做給我看，再從旁不斷地解釋給我聽要怎樣做。但對於 Logic Gate 我始終還是想很慢。

幸好以後不是再使用 Logic Gate，而是使用 Behavior Model 進行 VHDL 編寫。但我還是會花時間想辦法把這一次的實驗熟悉起來。在實驗完結之後，我有把 BCD 加法器部份理解完了，也認為再一次給我我應該也可以做出來，只是有點長時間未必可以很快做好。

加分題部份的 BCD 減法對我來說真的很難，因此主要都是由晟毅教我，我也似懂非懂地理解了。希望之後的 Behavior Model 在編寫時不會再像這樣的問題。

洪晟毅(104590048)：

這次實驗使用的邏輯閘較為複雜一些，皓鈞無法如先前實驗順利地完成，因此此次改由我進行主要程式碼的撰寫。

BCD Adder 的加法器難易度其實不高，僅要注意的是若大於9時須進行加6處理較不易理解。而最初在撰寫時，也因為遺漏一些條件而使加6的處理沒有順利執行，經過一段 Debug 的過程之後才注意到。

對於減法器的部分難度就相對提高不少，大部分同學都是以10的補數作為思考的出發點，但我實際去詢問過後發現，不需要特地使用10的補數也能完成，而且我也認為在2進位的數位電路設計之中，以10的補數作為思考方式有些怪異，所以我堅決使用先前所設計含有加減法功能的 FourBitsAdder 來實現。

過程中經歷不少失敗的挫折，我與隊友甚至浮出只想完成加法器就罷休的念頭。但我們仍努力不懈，充分利用課堂及助教課時間，經過不斷的嘗試及修正後，終於順利完成僅依靠2的補數進行的 BCD 減法器。

4. 組員貢獻度及工作內容：

名字	負責項目內容	貢獻比例	貢獻總和
晟毅	負責主要程式碼撰寫	20%	50%
	報告實驗內容的程式碼解釋	10%	
	報告實驗心得撰寫	20%	
皓鈞	負責實驗結果驗證	10%	50%
	報告實驗內容、實驗過程撰寫	10%	
	報告實驗心得撰寫	25%	
	負責程式程式碼重複實作的地方	5%	
總計		100%	100%