**微算機系統**

**小組專案報告**

實驗二

組別： 18

班級、姓名與學號：

資工二 蕭耕宏 110590005

資工二 楊榮鈞 110590034

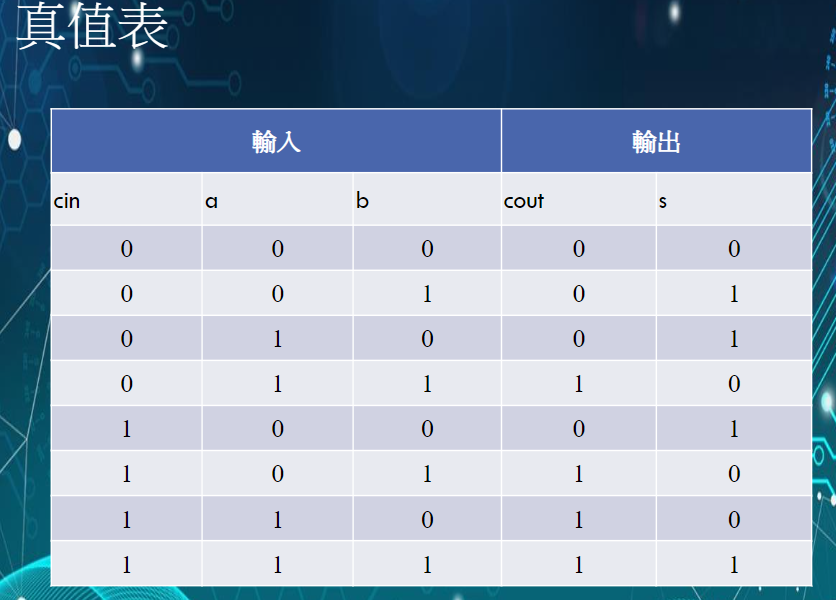
日期： 2022.10.03

1. 實驗內容：

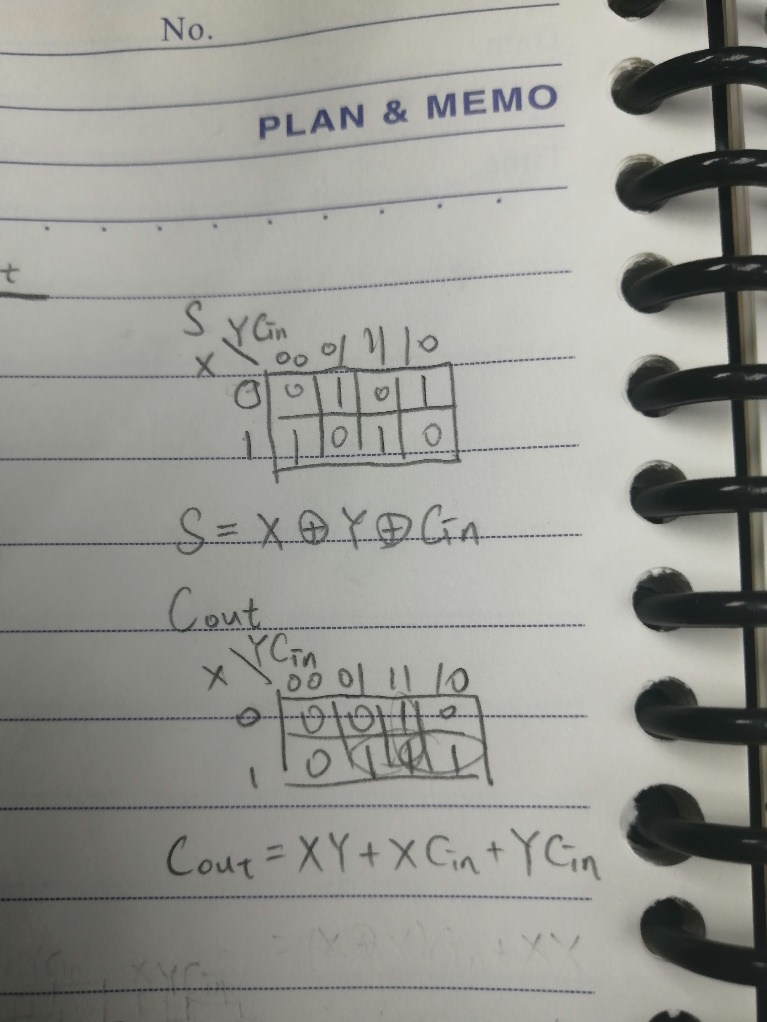
利用component、package及port map，使用1 bit的全加器做出一個8 bit的加法器和減法器，輸出用七段顯示器並且需要偵測overflow(用LED顯示)。

1. 實驗過程及結果：

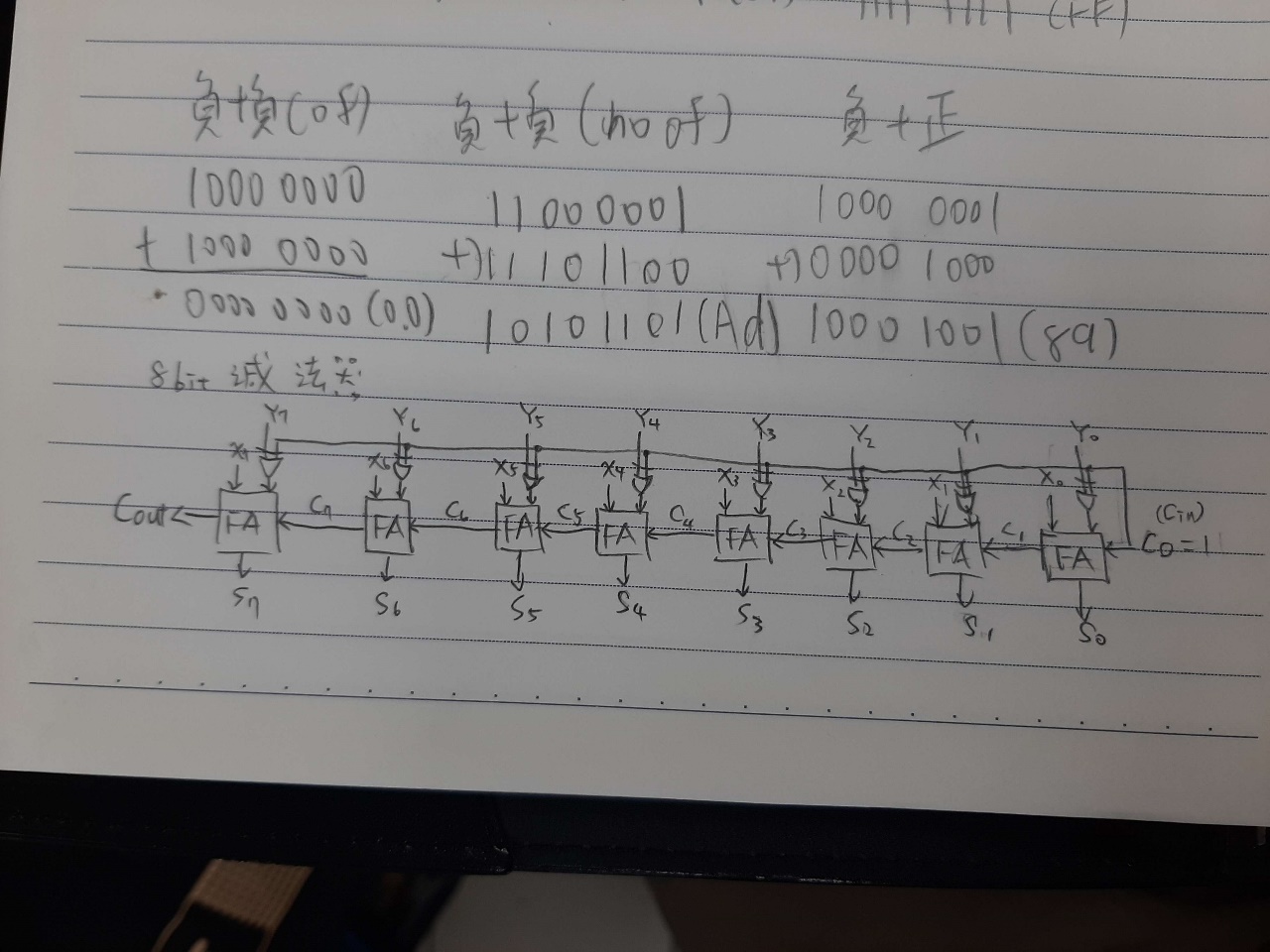
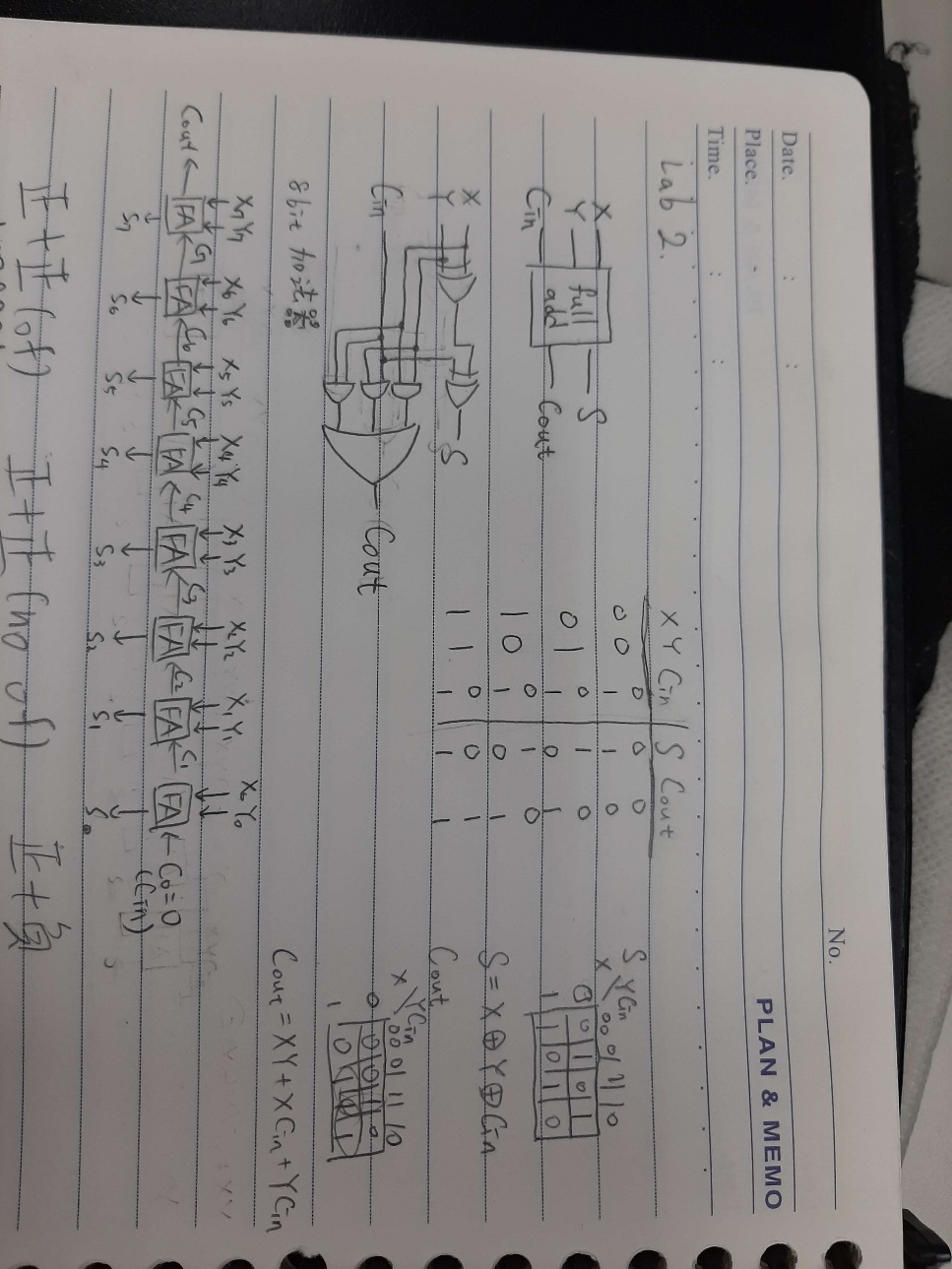
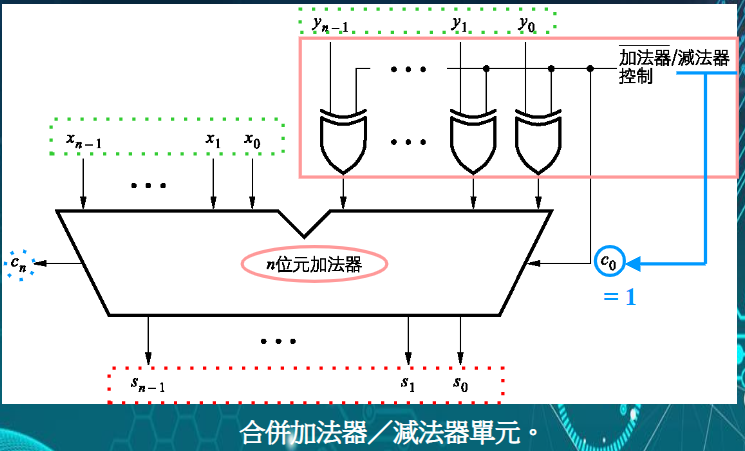
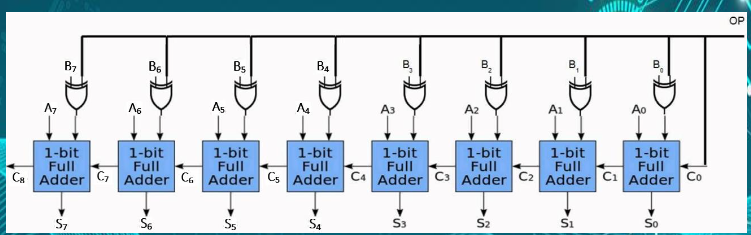
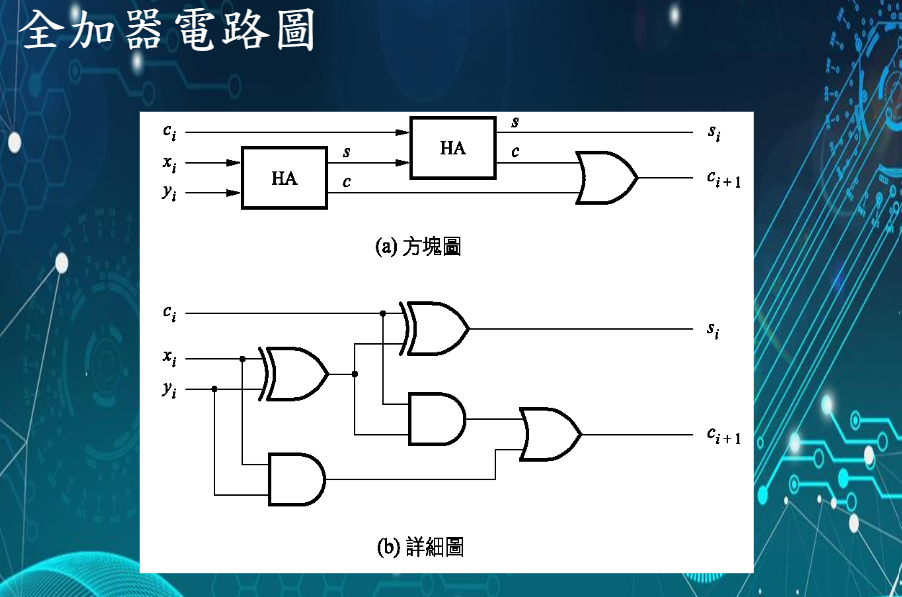
全加器的真值表



根據上方真值表使用卡諾圖來化簡並設計出電路



卡諾圖化簡後設計出的電路



實驗的結果

基礎題

|  |  |
| --- | --- |
|  |  |
| 1000 0000(X, 右) + 1000 0000(Y, 左)  = 0000 0000 (overflow) | 1100 0001(X, 右) + 1110 1100(Y, 左)  = 1010 1101 (no overflow) |
|  |  |
| 1000 0001(X, 右) + 0000 1000(Y, 左)  = 1000 1001 (no overflow) | 0100 0001(X, 右) + 0110 1000(Y, 左)  = 1010 1001 (overflow) |
|  |  |
| 0100 0001(X, 右) + 0010 1000(Y, 左)  = 0110 1001 (no overflow) | 0111 1111(X, 右) + 1000 0000(Y, 左)  = 1111 1111 (no overflow) |

進階題

|  |  |
| --- | --- |
|  |  |
| 0110 1000(X, 右) - 1001 0100(Y, 左)  = 1101 0100 (overflow) | 0010 1000(X, 右) - 1100 0000(Y, 左)  = 0110 1000 (no overflow) |
|  |  |
| 0010 1000(X, 右) - 0100 0000(Y, 左)  = 1110 1000 (no overflow) | 1010 1000(X, 右) - 0101 0100(Y, 左)  = 0101 0100 (overflow) |
|  |  |
| 1010 1000(X, 右) - 0001 0100(Y, 左)  = 1001 0100 (no overflow) | 1010 1000(X, 右) - 1001 0100(Y, 左)  = 0001 0100 (no overflow) |

1. 程式碼

|  |
| --- |
| 基本題 |
| fulladd.vhd |
| library ieee;  use ieee.std\_logic\_1164.all;  entity fulladd is  port( Cin,x,y : in std\_logic;  s,Cout : out std\_logic  );  end fulladd;  architecture func of fulladd is  begin  s <= x xor y xor Cin;  Cout <= (x and y) or (Cin and x) or (Cin and y);  end func; |
| hex.vhd |
| library ieee;  use ieee.std\_logic\_1164.all;  entity hex is  port(  SW2:in std\_logic\_vector(3 downto 0);  HEX2:out std\_logic\_vector(6 downto 0);  SW1:in std\_logic\_vector(3 downto 0);  HEX1:out std\_logic\_vector(6 downto 0)  );  end hex;  architecture behavioral of hex is  --boolean  begin  --a  HEX2(0) <= (SW2(2) and not(SW2(1)) and not(SW2(0))) or (SW2(3) and SW2(2) and not(SW2(1))) or (not(SW2(3)) and not(SW2(2)) and not(SW2(1)) and SW2(0)) or (SW2(3) and not(SW2(2)) and SW2(1) and SW2(0));  --b  HEX2(1) <= (SW2(3) and SW2(1) and SW2(0)) or (SW2(2) and SW2(1) and not(SW2(0))) or (SW2(3) and SW2(2) and not(SW2(0))) or (not(SW2(3)) and SW2(2) and not(SW2(1)) and SW2(0));  --c  HEX2(2) <= (SW2(3) and SW2(2) and not(SW2(0))) or SW2(3) and SW2(2) and SW2(1)) or (not(SW2(3)) and not(SW2(2)) and SW2(1) and not(SW2(0)));  --d  HEX2(3) <= (not(SW2(2)) and not(SW2(1)) and SW2(0)) or SW2(2) and SW2(1) and SW2(0)) or (not(SW2(3)) and SW2(2) and not(SW2(1)) and not(SW2(0))) or (SW2(3) and not(SW2(2)) and SW2(1) and not(SW2(0)));  --e  HEX2(4) <= (not(SW2(3)) and SW2(0)) or (not(SW2(3)) and SW2(2) and not(SW2(1))) or (not(SW2(2)) and not(SW2(1)) and SW2(0));  --f  HEX2(5) <= (SW2(3) and SW2(2) and not(SW2(1))) or (not(SW2(3)) and not(SW2(2)) and SW2(0)) or (not(SW2(3)) and not(SW2(2)) and SW2(1)) or (not(SW2(3)) and SW2(1) and SW2(0));  --g  HEX2(6) <= (not(SW2(3)) and not(SW2(2)) and not(SW2(1))) or (not(SW2(3)) and SW2(2) and SW2(1) and SW2(0));  ----1  --a  HEX1(0) <= (SW1(2) and not(SW1(1)) and not(SW1(0))) or (SW1(3) and SW1(2) and not(SW1(1))) or (not(SW1(3)) and not(SW1(2)) and not(SW1(1)) and SW1(0)) or (SW1(3) and not(SW1(2)) and SW1(1) and SW1(0));  --b  HEX1(1) <= (SW1(3) and SW1(1) and SW1(0)) or (SW1(2) and SW1(1) and not(SW1(0))) or (SW1(3) and SW1(2) and not(SW1(0))) or (not(SW1(3)) and SW1(2) and not(SW1(1)) and SW1(0));  --c  HEX1(2) <= (SW1(3) and SW1(2) and not(SW1(0))) or (SW1(3) and SW1(2) and SW1(1)) or (not(SW1(3)) and not(SW1(2)) and SW1(1) and not(SW1(0)));  --d  HEX1(3) <= (not(SW1(2)) and not(SW1(1)) and SW1(0)) or (SW1(2) and SW1(1) and SW1(0)) or (not(SW1(3)) and SW1(2) and not(SW1(1)) and not(SW1(0))) or (SW1(3) and not(SW1(2)) and SW1(1) and not(SW1(0)));  --e  HEX1(4) <= (not(SW1(3)) and SW1(0)) or (not(SW1(3)) and SW1(2) and not(SW1(1))) or (not(SW1(2)) and not(SW1(1)) and SW1(0));  --f  HEX1(5) <= (SW1(3) and SW1(2) and not(SW1(1))) or (not(SW1(3)) and not(SW1(2)) and SW1(0)) or (not(SW1(3)) and not(SW1(2)) and SW1(1)) or (not(SW1(3)) and SW1(1) and SW1(0));  --g  HEX1(6) <= (not(SW1(3)) and not(SW1(2)) and not(SW1(1))) or (not(SW1(3)) and SW1(2) and SW1(1) and SW1(0));  end behavioral; |
| Lab2\_package.vhd |
| library ieee;  use ieee.std\_logic\_1164.all;  package lab2\_package is  component fulladd  port( Cin,x,y : IN STD\_LOGIC;  s,Cout : OUT STD\_LOGIC  );  end component fulladd;    component hex  port( SW2:in std\_logic\_vector(3 downto 0);  HEX2:out std\_logic\_vector(6 downto 0);  SW1:in std\_logic\_vector(3 downto 0);  HEX1:out std\_logic\_vector(6 downto 0)  );  end component hex;  end lab2\_package; |
| Lab2\_1.vhd |
| library ieee;  use ieee.std\_logic\_1164.all;  use work.lab2\_package.all;  entity Lab2\_1 is  port( X,Y : in std\_logic\_vector(7 downto 0);  hex2,hex1: out std\_logic\_vector(6 downto 0);  overflow: out std\_logic  );  end Lab2\_1;  architecture behavioral of Lab2\_1 is  signal C : std\_logic\_vector(7 downto 0);  signal Cout: std\_logic;  signal S : std\_logic\_vector(7 downto 0);  begin  C(0) <= '0';  stage0: fulladd port map (C(0),X(0),Y(0),S(0),C(1));  stage1: fulladd port map (C(1),X(1),Y(1),S(1),C(2));  stage2: fulladd port map (C(2),X(2),Y(2),S(2),C(3));  stage3: fulladd port map (C(3),X(3),Y(3),S(3),C(4));  stage4: fulladd port map (C(4),X(4),Y(4),S(4),C(5));  stage5: fulladd port map (C(5),X(5),Y(5),S(5),C(6));  stage6: fulladd port map (C(6),X(6),Y(6),S(6),C(7));  stage7: fulladd port map (C(7),X(7),Y(7),S(7),Cout);  overflow <= C(7) xor Cout;  stage8: hex port map(SW2(3 downto 0)=>S(7 downto 4), SW1(3 downto 0)=>S(3 downto 0),  HEX2(6 downto 0)=>hex2(6 downto 0),HEX1(6 downto 0)=>hex1(6 downto 0));    end behavioral; |

|  |
| --- |
| 加分題 |
| Lab2\_2.vhd |
| library ieee;  use ieee.std\_logic\_1164.all;  use work.lab2\_package.all;  entity Lab2\_2 is  port( X,Y : in std\_logic\_vector(7 downto 0);  hex2,hex1: out std\_logic\_vector(6 downto 0);  overflow: out std\_logic  );  end Lab2\_2;  architecture behavioral of Lab2\_2 is  signal C : std\_logic\_vector(7 downto 0);  signal Cout: std\_logic;  signal S : std\_logic\_vector(7 downto 0);  begin  C(0) <= '1';  stage0: fulladd port map (C(0),X(0),Y(0) xor C(0),S(0),C(1));  stage1: fulladd port map (C(1),X(1),Y(1) xor C(0),S(1),C(2));  stage2: fulladd port map (C(2),X(2),Y(2) xor C(0),S(2),C(3));  stage3: fulladd port map (C(3),X(3),Y(3) xor C(0),S(3),C(4));  stage4: fulladd port map (C(4),X(4),Y(4) xor C(0),S(4),C(5));  stage5: fulladd port map (C(5),X(5),Y(5) xor C(0),S(5),C(6));  stage6: fulladd port map (C(6),X(6),Y(6) xor C(0),S(6),C(7));  stage7: fulladd port map (C(7),X(7),Y(7) xor C(0),S(7),Cout);  overflow <= C(7) xor Cout;  stage8: hex port map(SW2(3 downto 0)=>S(7 downto 4), SW1(3 downto 0)=>S(3 downto 0),  HEX2(6 downto 0)=>hex2(6 downto 0),HEX1(6 downto 0)=>hex1(6 downto 0));    end behavioral; |

1. 本次實驗過程說明與解決方法:

實驗過程:

基本題是實作一個8bits加法器，使用兩個七段顯示器輸出，基本上是利用老師在PPT上的加法器布林代數式和上一次Lab的code，把上一次Lab的code轉換成一個component，和一個加法器的component一起裝在package裡，都做好後，再將Carry In MSB和Carry Out MSB做XOR做overflow偵測即完成本題。

進階題與基本題大同小異，相較第一題，只需將一邊取二補數相加，意即將C(0)=1，再對一個數的每個bit(減數)和1(我們用的是C(0))做XOR，即可實作減法器。

實作過程中有遇到幾個小問題，我們無意間把S和C做成有input和output，導致一開始燒錄的時候會有些腳位是預設在一些我們不希望亮的腳位。

解決方法:

因為腳位莫名奇妙亮起來，我們開始檢查FPGA是否壞了或是老師PPT上的腳位是否正確，結果FPGA沒壞，PPT上的腳位也是正確的。

最後，我們發現code中的錯誤，我們把S和C用成signal後，問題就解決了。