**微算機系統**

**小組專案報告**

實驗四

組別： 18

班級、姓名與學號：

資工二 蕭耕宏 110590005

資工二 楊榮鈞 110590034

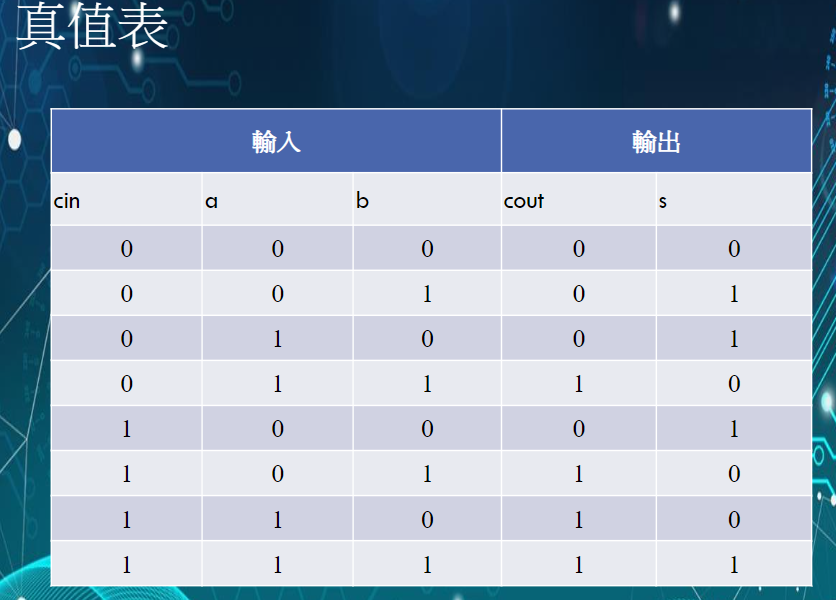
日期： 2022.10.25

1. 實驗內容：

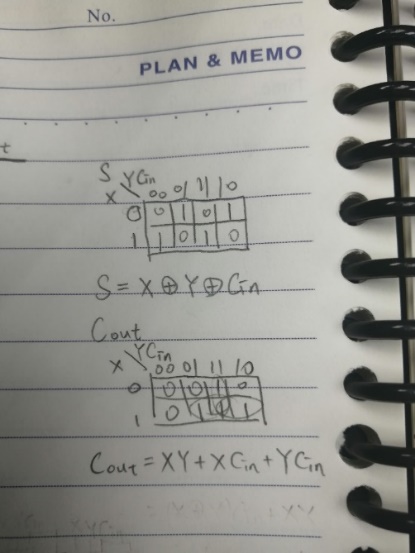
利用component、package、port map、for generate、if generate完成7bits的ALU(功能包括and、or、add、subtract、set-less-than、nor)，並用七段顯示器顯示ALU的輸出結果，overflow用LED顯示。

1. 實驗過程及結果：

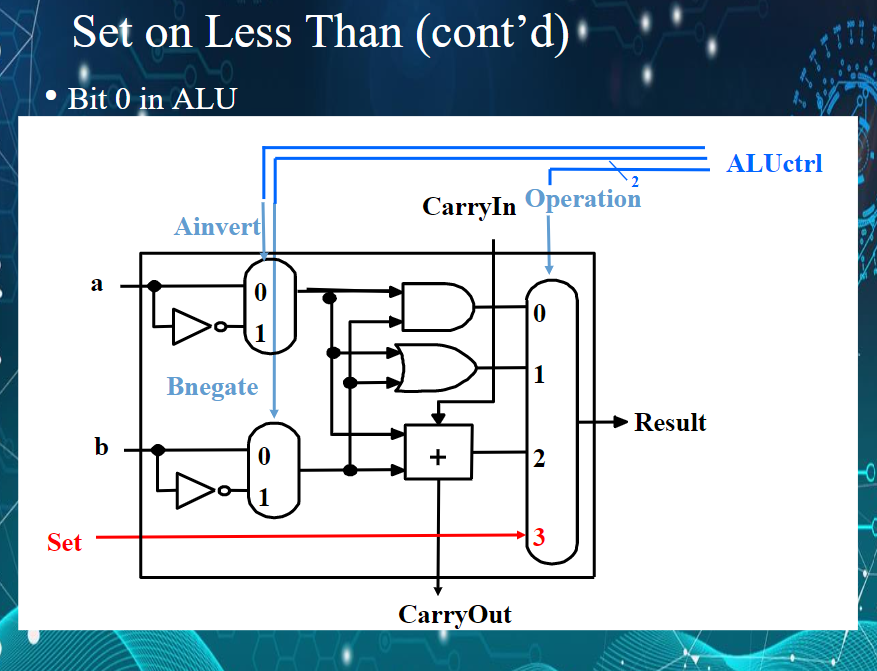
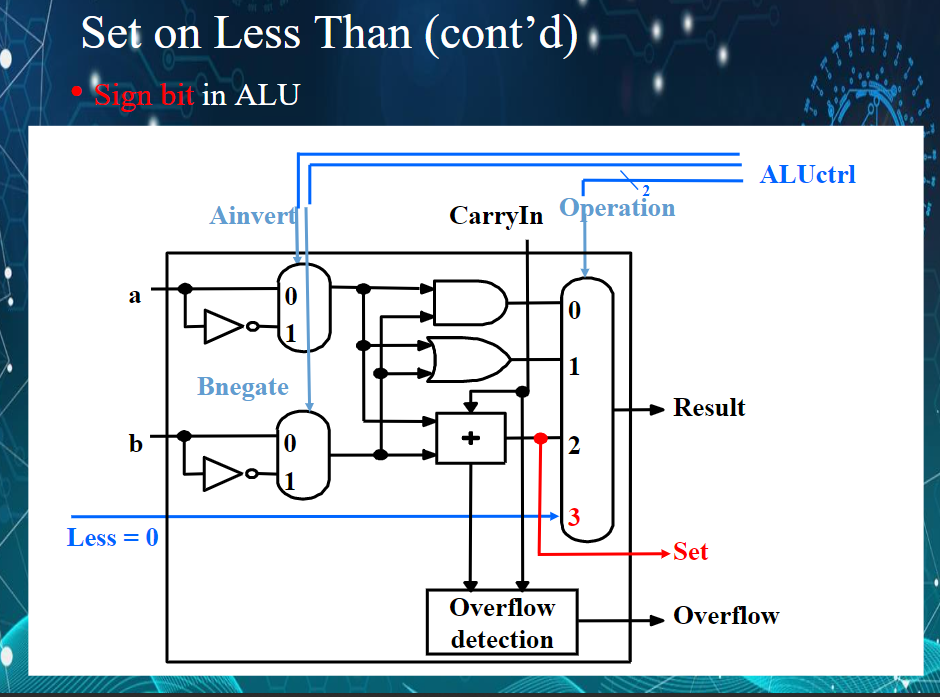
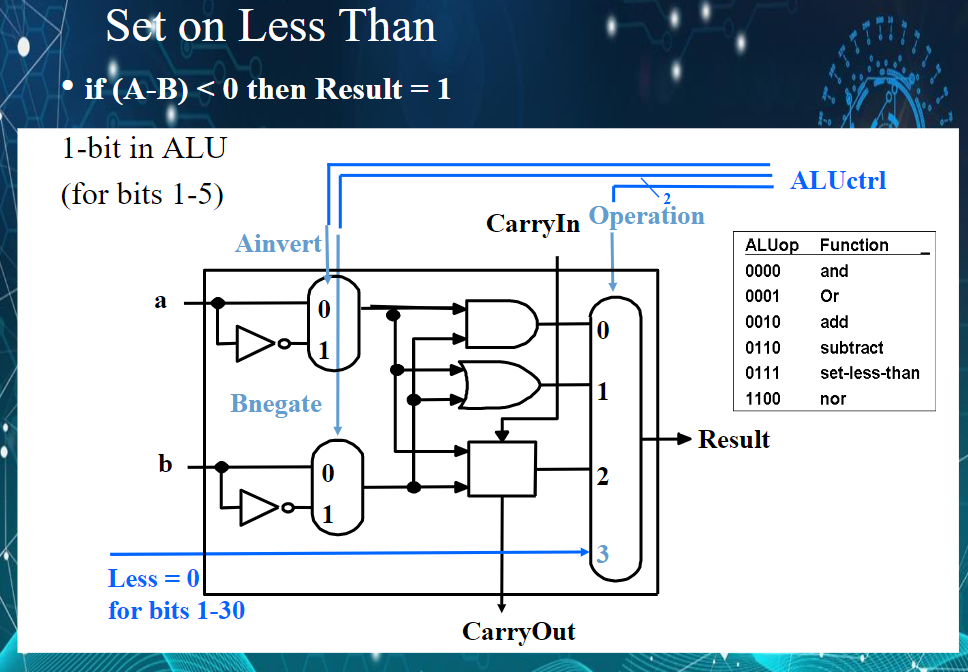
全加器的真值表

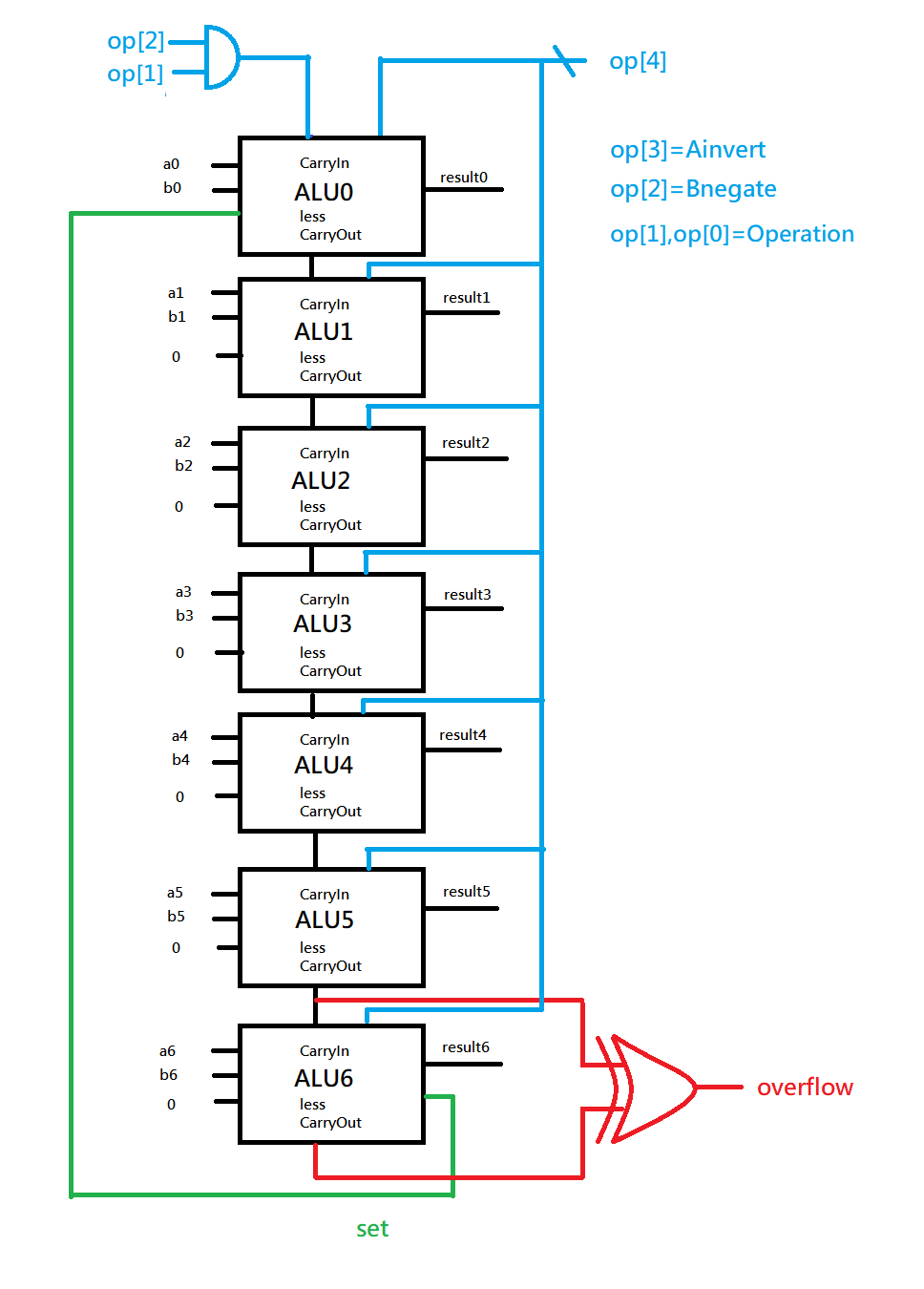


根據上方真值表使用卡諾圖來化簡並設計出電路



卡諾圖化簡後設計出的電路





實驗的結果

進階題

|  |  |
| --- | --- |
|  |  |
| 111 1111(A, 右) and 111 1111(B, 左)  = 111 1111 | 011 1111(A, 右) and 111 1110(B, 左)  = 011 1110 |
|  |  |
| 101 0101(A, 右) or 010 1010(B, 左)  = 111 1111 | 000 0011 (A, 右) or 011 0000(B, 左)  = 011 0011 |
|  |  |
| 000 1000(A, 右) + 000 0100(B, 左)  = 000 1100 (no overflow) | 100 0000(A, 右) + 100 0000(B, 左)  = 000 0000 (overflow) |
|  |  |
| 000 1000(A, 右) - 000 0010(B, 左)  = 000 0110 (no overflow) | 100 0000(A, 右) - 000 0001(B, 左)  = 011 1111 (overflow) |
|  |  |
| 000 1000(A, 右) - 000 0100(B, 左)  = 000 0100 > 0 (no overflow)  Result = 000 0000 | 100 0000(A, 右) - 000 0001(B, 左)  = 011 1111 > 0 (overflow)  Result = 000 0000 |
|  |  |
| 001 0000(A, 右) - 010 0000(B, 左)  = 111 0000 < 0 (no overflow)  Result = 000 0001 | 010 0000(A, 右) - 100 0000(B, 左)  = 110 0000 < 0 (overflow)  Result = 000 0001 |
|  |  |
| 000 0001(A, 右) nor 000 0010(B, 左)  = 111 1100 | 100 0000(A, 右) nor 100 0000(B, 左)  = 011 1111 |

1. 程式碼

|  |
| --- |
| 進階題 |
| 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; |
| mux\_2to1.vhd |
| library ieee;  use ieee.std\_logic\_1164.all;  entity mux\_2to1 is  port( w0,w1 : in std\_logic;  s : in std\_logic;  f : out std\_logic  );  end mux\_2to1;  architecture behavioral of mux\_2to1 is  signal temp : std\_logic;  begin  with s select  f <= w0 when '0',  w1 when others;  end behavioral; |
| mux\_4to1.vhd |
| library ieee;  use ieee.std\_logic\_1164.all;  entity mux\_4to1 is  port( w0,w1,w2,w3 : in std\_logic;  s : in std\_logic\_vector(1 downto 0);  f : out std\_logic  );  end mux\_4to1;  architecture behavioral of mux\_4to1 is  begin  with s select  f <= w0 when "00",  w1 when "01",  w2 when "10",  w3 when others;  end behavioral; |
| onebitALU.vhd |
| library ieee;  use ieee.std\_logic\_1164.all;  use ieee.std\_logic\_unsigned.all;  use work.lab4\_package.all;  entity onebitALU is  port( A,B,less,carryin : in std\_logic;  opcode : in std\_logic\_vector(3 downto 0);--0000  result,set,carryout : out std\_logic  );  end onebitALU;  architecture LogicFunc of onebitALU is  signal Ain : std\_logic;  signal Bin : std\_logic;  signal w0\_and: std\_logic;  signal w1\_or : std\_logic;  signal w2\_add: std\_logic; -- set  begin    stage0:mux\_2to1 port map(A,not A,opcode(3),Ain);  stage1:mux\_2to1 port map(B,not B,opcode(2),Bin);    w0\_and <= Ain and Bin;  w1\_or <= Ain or Bin;    stage2:fulladd port map(Cin=>carryin,x=>Ain,y=>Bin,s=>w2\_add,Cout=>carryout);    set <= w2\_add;    stage3:mux\_4to1 port map(w0=>w0\_and,w1=>w1\_or,w2=>w2\_add,w3=>less,s(1 downto 0)=>opcode(1 downto 0),f=>result);    end LogicFunc; |
| lab4\_package.vhd |
| library ieee;  use ieee.std\_logic\_1164.all;  package lab4\_package is  component fulladd  port( Cin,x,y : IN STD\_LOGIC;  s,Cout : OUT STD\_LOGIC  );  end component fulladd;    component mux\_4to1  port( w0,w1,w2,w3 : in std\_logic;  s : in std\_logic\_vector(1 downto 0);  f : out std\_logic  );  end component mux\_4to1;    component mux\_2to1  port( w0,w1 : in std\_logic;  s : in std\_logic;  f : out std\_logic  );  end component mux\_2to1;    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;    component onebitALU  port( A,B,less,carryin : in std\_logic;  opcode : in std\_logic\_vector(3 downto 0);  result,set,carryout : out std\_logic  );  end component onebitALU;  end lab4\_package; |
| Lab4\_1.vhd |
| library ieee;  use ieee.std\_logic\_1164.all;  use ieee.std\_logic\_unsigned.all;  use work.lab4\_package.all;  entity Lab4\_1 is  port( A,B : in std\_logic\_vector(6 downto 0);  opcode : in std\_logic\_vector(3 downto 0);  overflow : out std\_logic;  HEX2:out std\_logic\_vector(6 downto 0);  HEX1:out std\_logic\_vector(6 downto 0)  );  end Lab4\_1;  architecture behavioral of Lab4\_1 is  signal result : std\_logic\_vector(7 downto 0);  signal set : std\_logic\_vector(6 downto 0);  signal carryout : std\_logic\_vector(6 downto 0);  signal temp : std\_logic\_vector(9 downto 0);  begin  temp(9) <= opcode(2) and opcode(1);  G1:for i in 0 to 6 generate  G2:if i=0 generate  stage0: onebitALU port map (A=>A(0),B=>B(0),less=>'0',carryin=>temp(9),  opcode(3 downto 0)=>opcode(3 downto 0),  result=>temp(0),set=>set(0),carryout=>carryout(0));  end generate;  G3:if (i<6) and (i>0)generate  stage1: onebitALU port map (A=>A(i),B=>B(i),less=>'0',carryin=>carryout(i-1),  opcode(3 downto 0)=>opcode(3 downto 0),  result=>result(i),set=>set(i),carryout=>carryout(i));  end generate;  G4:if i=6 generate  stage2: onebitALU port map (A=>A(i),B=>B(i),less=>'0',carryin=>carryout(i-1),  opcode(3 downto 0)=>opcode(3 downto 0),  result=>result(i),set=>set(i),carryout=>carryout(i));    stage3: onebitALU port map (A=>A(0),B=>B(0),less=>set(6),carryin=>temp(9),  opcode(3 downto 0)=>opcode(3 downto 0),  result=>result(0),set=>temp(1),carryout=>temp(2));  end generate;  end generate;    result(7) <= '0';  overflow <= carryout(6) xor carryout(5);    stage4: hex port map (SW2(3 downto 0)=>result(7 downto 4),SW1(3 downto 0)=>result(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1bit的ALU。

做法是使用兩個2對1的多工器決定A、B是否為not A、not B，然後再用2對1多工器得出的A、B去實作and、or、add (1bit的全加器)、subtract(1bit的全加器)、set-less-than (if(A-B)<0 then result = 1 else result = 0(用subtract去看))、nor(其做法為(not A) and (not B) = A nor B)，最後再用4對1多工器去選擇result要輸出什麼。

其中opcode有4bits，opcode[3]為Ainvert(用MUX去決定A還是not A)，opcode[2]為Bnegate(用MUX去決定B還是not B)，opcode[1]、opcode[0]為operation(用MUX去決定要輸出and、or、add、less哪一個)。

最後再用七段顯示器顯示出ALU的result，overflow則是以carrryin xor carryout決定是否亮LED燈。

在做上述動作時，分別要把1bit的全加器、1bit的ALU(overflow由主程式判斷，未放在ALU的package裡面)、七段顯示器給包裝成package。

而進階題就是直接利用基礎題的程式碼，做出7bits的ALU，再用七段顯示器輸出ALU的result，並且LSB的set-less-than的輸出是MSB(sign bit)的set的值，其餘的bit的set-less-than的輸出都是0。其中overflow一樣在主程式判斷，未放在1bit ALU的package中。

實作過程中有遇到一些小問題，就是我們對quartus的if generate的做法不熟悉，所以在寫MUX的時候用if generate沒寫出來，然後用老師的MUX的code的select with when的寫法的時候沒跑動。

解決方法:

1. if generate雖然是if的寫法，但是他不能把要輸入的東西當作判斷式，所以我們改用select with when的寫法，完成MUX的code。

2. 老師的MUX的code的select with when的寫法的時候沒跑動的原因是我們是先寫完老師的code再去試if generate能否寫出來的時候，我們在處理註解的時候不小心多加了一個符號在code裡面沒有發現所以沒跑動。在我們刪掉那個符號之後，MUX的code就能跑了。