**微算機系統**

**小組專案報告**

實驗三

組別： 18

班級、姓名與學號：

資工二 蕭耕宏 110590005

資工二 楊榮鈞 110590034

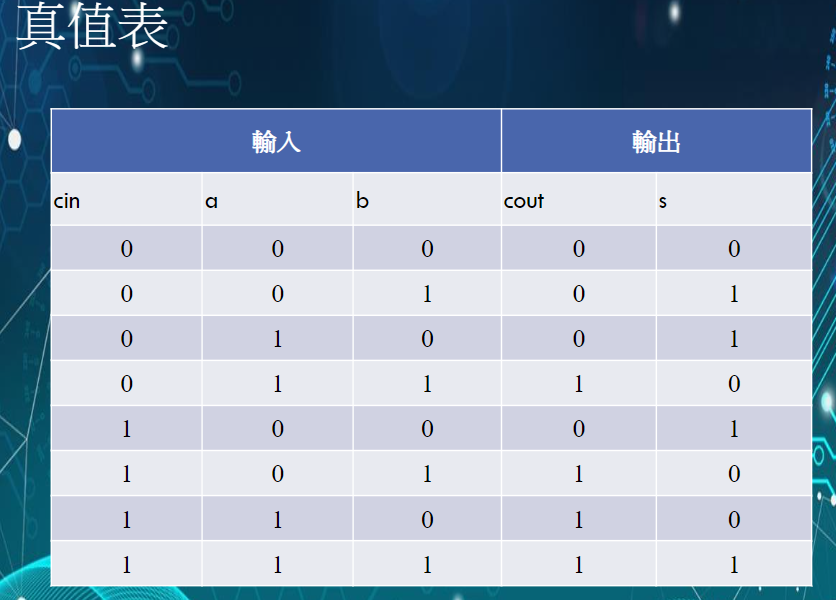
日期： 2022.10.16

1. 實驗內容：

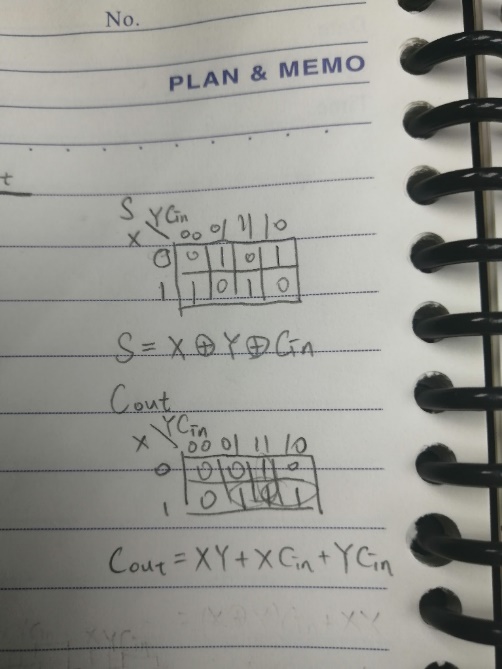
利用component、package及port map，使用1bit的全加器組合成4bits的加法器，再由4bits的加法器組成4bits的BCD加法器，最後再用4bits的BCD加法器組成8bits的BCD加法器並用七段顯示器顯示BCD加法器的輸出結果。

1. 實驗過程及結果：

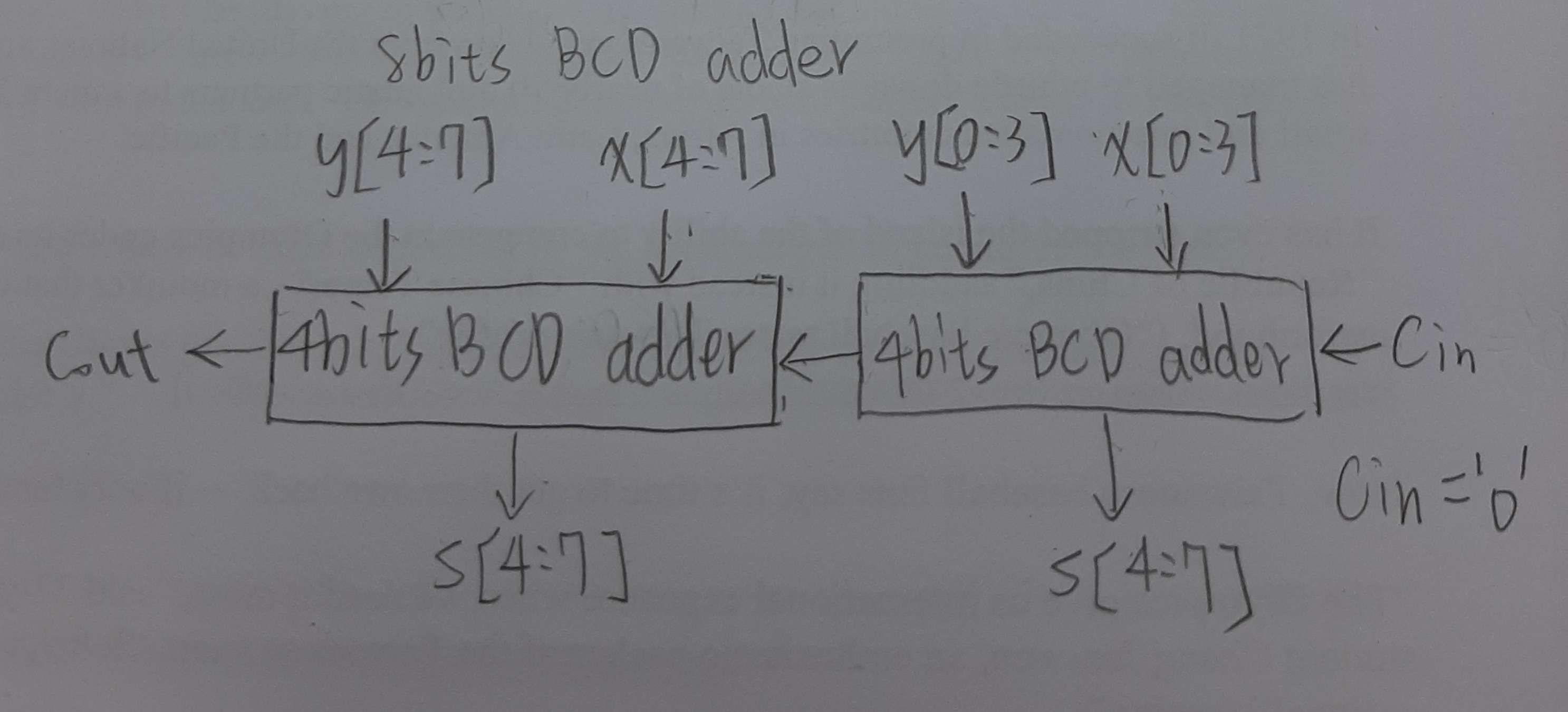
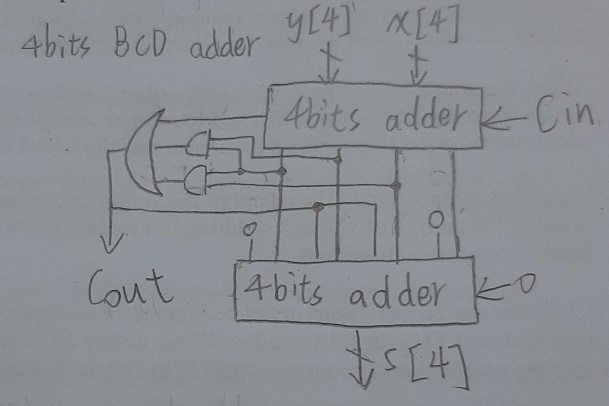
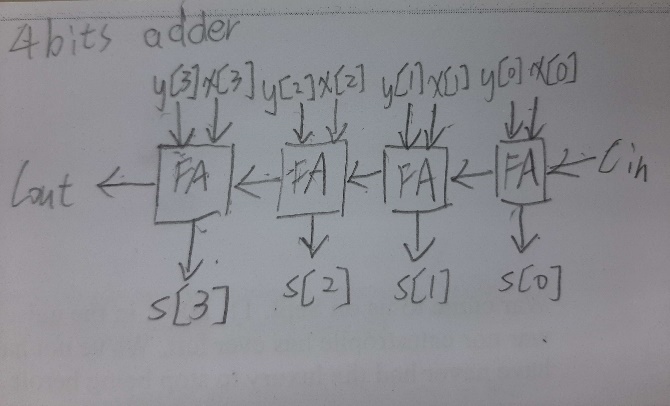
全加器的真值表



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

|  |  |
| --- | --- |
|  |  |
| 0001 0101(A, 右) + 0001 0101(B, 左)  = 0011 0000 | 0101 0000(A, 右) + 0100 1001(B, 左)  = 1001 1001 |
|  |  |
| 0001 0001(A, 右) + 0011 1001(B, 左)  = 0101 0000 | 0000 0111(A, 右) + 0000 0010(B, 左)  = 0000 1001 |

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; |
| add\_4bits.vhd |
| library ieee;  use ieee.std\_logic\_1164.all;  use ieee.std\_logic\_unsigned.all;  use work.lab3\_package.all;  entity add\_4bits is  port( X,Y : in std\_logic\_vector(3 downto 0);  Ci : in std\_logic;  Cout: out std\_logic;  S : out std\_logic\_vector(3 downto 0)  );  end add\_4bits;  architecture func of add\_4bits is  signal C : std\_logic\_vector(3 downto 0);  begin  C(0) <= Ci;  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),Cout);  end func; |
| BCDadd.vhd |
| library ieee;  use ieee.std\_logic\_1164.all;  use ieee.std\_logic\_unsigned.all;  use work.lab3\_package.all;  entity BCDadd is  port( A,B : in std\_logic\_vector(3 downto 0);  Ci : in std\_logic;  Co : inout std\_logic;  S : out std\_logic\_vector(3 downto 0)  );  end BCDadd;  architecture func of BCDadd is  signal Sum : std\_logic\_vector(3 downto 0);  signal Cout: std\_logic;  signal temp: std\_logic\_vector(3 downto 0);  begin  stage0: add\_4bits port map (X(3 downto 0)=>A(3 downto 0),Y(3 downto 0)=>B(3 downto 0),  Ci=>Ci,Cout=>Cout,S(3 downto 0)=>Sum(3 downto 0));    Co <= Cout or (Sum(3) and Sum(2)) or (Sum(3) and Sum(1));  temp(3) <= Co;  stage1: add\_4bits port map (X(3 downto 0)=>Sum(3 downto 0),  Y(0)=>'0',Y(1)=>temp(3),Y(2)=>Co,Y(3)=>'0',  Ci=>'0',Cout=>temp(2),S(3 downto 0)=>S(3 downto 0));  end func; |
| lab3\_package.vhd |
| library ieee;  use ieee.std\_logic\_1164.all;  package lab3\_package is  component fulladd  port( Cin,x,y : IN STD\_LOGIC;  s,Cout : OUT STD\_LOGIC  );  end component fulladd;    component add\_4bits  port( X,Y : in std\_logic\_vector(3 downto 0);  Ci : in std\_logic;  Cout: out std\_logic;  S : out std\_logic\_vector(3 downto 0)  );  end component add\_4bits;    component BCDadd  port( A,B : in std\_logic\_vector(3 downto 0);  Ci : in std\_logic;  Co : out std\_logic;  S : out std\_logic\_vector(3 downto 0)  );  end component BCDadd;    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 lab3\_package; |
| Lab3\_2.vhd |
| library ieee;  use ieee.std\_logic\_1164.all;  use ieee.std\_logic\_unsigned.all;  use work.lab3\_package.all;  entity Lab3\_2 is  port( A,B : in std\_logic\_vector(7 downto 0);  HEX2:out std\_logic\_vector(6 downto 0);  HEX1:out std\_logic\_vector(6 downto 0)  );  end Lab3\_2;  architecture behavioral of Lab3\_2 is  signal Ci : std\_logic;  signal Co : std\_logic;  signal temp: std\_logic;  signal S : std\_logic\_vector(7 downto 0);  begin  Ci <= '0';  stage0: BCDadd port map (A(3 downto 0)=>A(3 downto 0),B(3 downto 0)=>B(3 downto 0),  Ci=>Ci,Co=>Co,S(3 downto 0)=>S(3 downto 0));    stage1: BCDadd port map (A(3 downto 0)=>A(7 downto 4),B(3 downto 0)=>B(7 downto 4),  Ci=>Co,Co=>temp,S(3 downto 0)=>S(7 downto 4));    stage2: 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4bits的BCD加法器。

做法是使用1bit的全加器組成4bits的加法器，再使用4bits的加法器組成4bits的BCD加法器，其中在組成BCD加法器的時候要把BCD加法器的進位方式做出來，最後在接到七段顯示器輸出。

在做上述動作時，分別要把1bit的全加器、4bits的加法器、4bits的BCD加法器、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，把兩個4bits的BCD加法器組合成8bits的BCD加法器，在接上七段顯示器輸出。

實作過程中有遇到一個小問題，我們對quartus的package的做法不熟悉，所以在這邊卡了一段時間，原本用很多package去包裝每一個步驟，但發現太過冗贅，而且不太理解要如何用package去包裝另一個package。

解決方法:

我們後來結合網路上學到的以及一些Lab2的package的用法得出，我們可以直接用一個package包裝所有我們想要的包裝的東西，這樣可以在打程式的時候省時又省力。