1. 組別：第 22 組

2. 組員：404415073 電機二 蔡孟勳　404415055 電機二 劉恩瑞

3. 題目名稱：實驗 10 自動販賣機

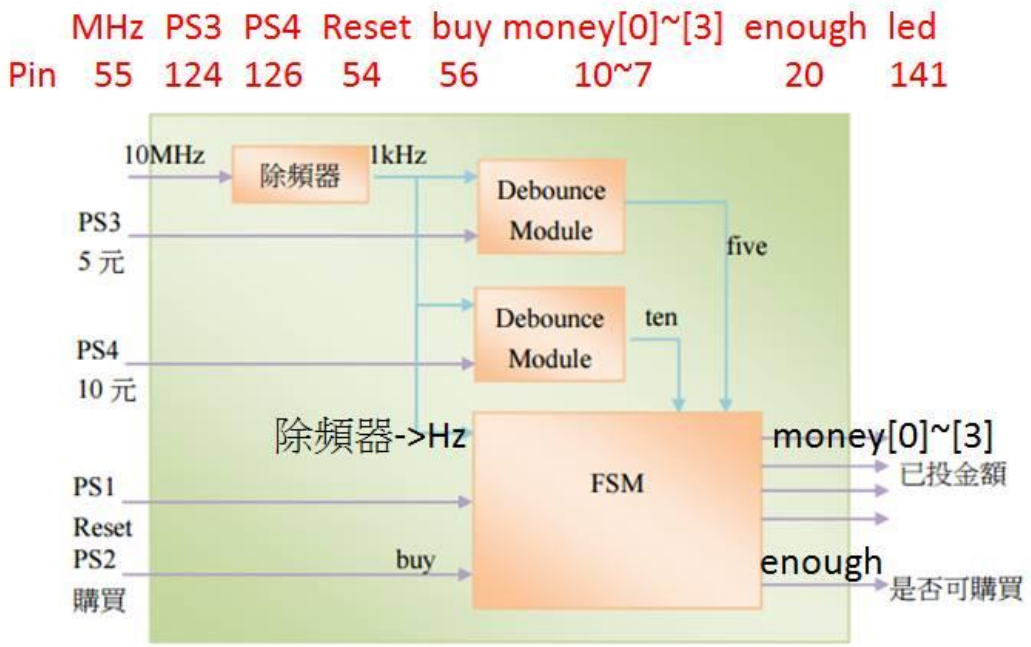
4. 功能說明：

這次的實驗為撰寫一個自動販賣機

共有四個輸入訊號分別為 5 元或 10 元、購買和退幣，按下 5 元鍵，則代表投了 5 元；按下 10 元鍵，則代表投了 10 元；按下購買鍵，則代表花了 15 元買東西；按下退幣鍵，則代表重新開始。輸出訊號則分為兩種，一種是以 4 顆 LED 代表目前有多少錢，每顆 LED 分別代表 5 元，另一種是判斷是否可以購買的燈，若超過 15 元，則燈會亮起。

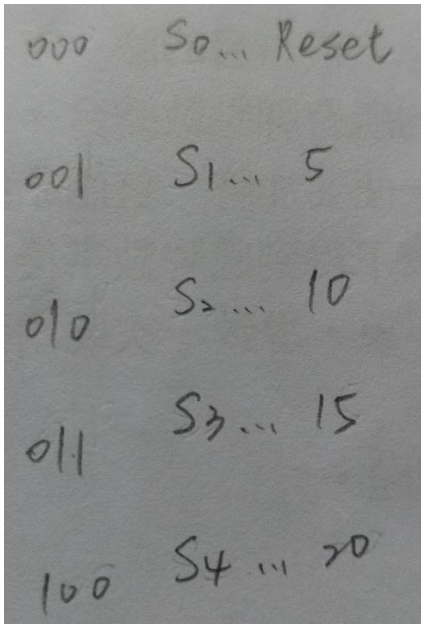
整個電路的功能為：模擬實際的自動販賣機

5. 硬體架構圖：



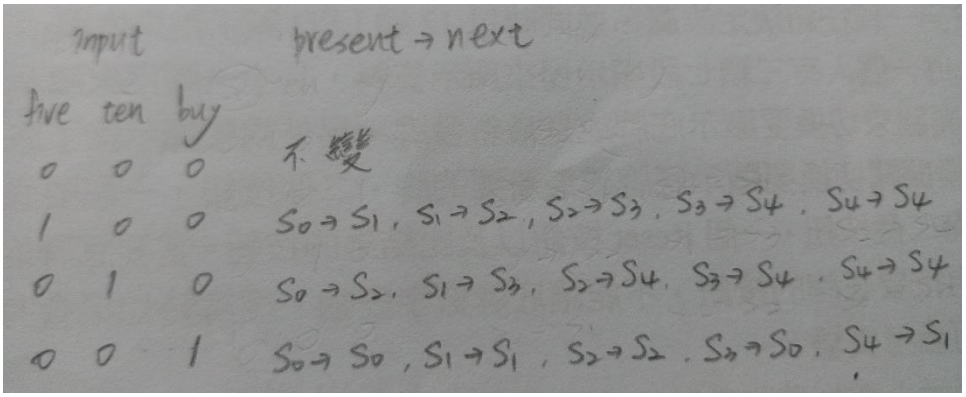
電路設計的想法:

本次實驗我們選擇以 Moore 的方式完成，我們先以除頻器將 10MHz 的脈波降至 1kHz 和 1Hz，1kHz 是為了讓 5 元和 10 元的按鍵得以消除彈跳，1Hz 則是為了驅動本次的 FSM。其實剩下的 FSM 只要將 state table 畫好，並隨著輸入變化即可，投 5 元就亮一顆燈，投 10 元就亮兩顆燈，超過 20 元時，以 20 元 (四顆燈) 表示，超過 15 元時，可購買的燈會亮起；按下購買鍵就暗三顆燈，若原本就少於三顆燈，狀態則維持原樣、按下退幣鍵就讓一切重頭開始，這樣就可以完成本次的實驗了。



(左圖為 state destription)

(下圖為 state table)



6. 程式碼&註解：

```
module exp10(money,enough,led,MHz,PS3,PS4,Reset,buy); //自動販賣機模組宣告
 input MHz,PS3,PS4,Reset,buy; //輸入為MHz,PS3,PS4,Reset,buy
 output reg [3:0]money; //保留輸出money[0]~[3]的值到下一次指定新值
 output enough,led; //輸出為enough,led
 wire kHz,Hz,five,ten; //自動販賣機的邏輯電路圖中會用到4條線連接不同模組
 reg [2:0]current,next; //保留current[0]~[2],next[0]~[2]的值到下一次指定新值
 parameter s0=3'd0,s1=3'd1,s2=3'd2,s3=3'd3,s4=3'd4; //定義固定的值s0為0,s1為1,s2為2,s3為3,s4為4

 div10000 d1(kHz,MHz); //使用除頻器1的模組，輸出為kHz，輸入為MHz
 div1000 d2(Hz,kHz); //使用除頻器2的模組，輸出為Hz，輸入為kHz
 debounce d3(five,PS3,kHz); //使用消除彈跳模組，輸出為five，輸入為PS3,kHz
 debounce d4(ten,PS4,kHz); //使用消除彈跳模組，輸出為ten，輸入為PS4,kHz

 always@(current,five,ten,buy) //當current,five,ten,buy的值有改變時，底下的Behavioral Model的敘述會被執行
  begin //開始
   if(five==1) //如果five的值等於1，執行以下敘述
    case(current) //current的值為true時，執行以下敘述
     s0:next=s1; //current為s0時，next=s1
     s1:next=s2; //current為s1時，next=s2
     s2:next=s3; //current為s2時，next=s3
     s3:next=s4; //current為s3時，next=s4
     s4:next=s4; //current為s4時，next=s4
     default: next=s0; //當current為其他值時，next=s0
    endcase //結束case


   else if(ten==1) //其他如果ten的值等於1，執行以下敘述
    case(current) //current的值為true時，執行以下敘述
     s0:next=s2; //current為s0時，next=s2
     s1:next=s3; //current為s1時，next=s3
     s2:next=s4; //current為s2時，next=s4
     s3:next=s4; //current為s3時，next=s4
     s4:next=s4; //current為s4時，next=s4
     default: next=s0; //當current為其他值時，next=s0
    endcase //結束case

   else if(buy==0) //其他如果buy的值等於0，執行以下敘述
    case(current)  //current的值為true時，執行以下敘述
     s0:next=s0; //current為s0時，next=s0
     s1:next=s1; //current為s1時，next=s1
     s2:next=s2; //current為s2時，next=s2
     s3:next=s0; //current為s3時，next=s0
     s4:next=s1; //current為s4時，next=s1
     default: next=s0; //當current為其他值時，next=s0
    endcase //結束case

   else //其他情況
    next=current; //next的值變為current的值
  end //結束
```

```
  always@(posedge Hz or negedge Reset) //當Hz正緣觸發或Reset負緣觸發時，底下的Behavioral Model的敘述會被執行
   if(~Reset) //如果Reset反相為true，執行以下敘述
    current<=s0; //current的值變為s0
   else //其他情況
    current<=next; //current的值變為next的值

  assign enough=(current==s3 || current==s4); //宣告enough的值為current==s3或current==s4的邏輯判斷

  always@(current) //當current的值有變動時，底下的Behavioral Model的敘述會被執行
   case(current) //current的值為true時，執行以下敘述
    s0:money=4'b0000; //current為s0時，money的值為二進位0000
    s1:money=4'b0001; //current為s1時，money的值為二進位0001
    s2:money=4'b0011; //current為s2時，money的值為二進位0011
    s3:money=4'b0111; //current為s3時，money的值為二進位0111
    s4:money=4'b1111; //current為s4時，money的值為二進位1111
    default: money=4'b0000; //current為其他值時，money的值為二進位0000
   endcase //結束case
  assign led=1'b1;  //宣告led的值為1
endmodule   //結束模組
```

```
module div1000(out,in); //除頻器除以1000的模組宣告
  input in; //輸入為in
  output reg out; //保留輸出out的值到下一次指定新值
  reg [8:0]counter; //保留輸出counter[0]~[8]的值到下一次指定新值
  always@(posedge in) //當in正緣觸發時，底下的Behavioral Model的敘述會被執行
  if(counter==9'd499) //如果counter的值等於499時，執行以下敘述
   begin //開始
    counter<=9'd0; //counter的值變為0
    out<=~out; //out反相
   end //結束
  else //其他情況
   counter<=counter+9'd1; //conuter的值+1
endmodule  //結束模組

module div10000(out,in); //除頻器除以10000的模組宣告
  input in; //輸入為in
  output reg out; //保留輸出out的值到下一次指定新值
  reg [12:0]counter; //保留輸出counter[0]~[12]的值到下一次指定新值
  always@(posedge in) //當in正緣觸發時，底下的Behavioral Model的敘述會被執行
  if(counter==13'd4999) //如果counter的值等於4999時，執行以下敘述
   begin //開始
    counter<=13'd0; //counter的值變為0
    out<=~out; //out反相
   end //結束
  else //其他情況
   counter<=counter+13'd1; //conuter的值+1
endmodule  //結束模組
```

```
module debounce(out,in,kHz); //消除彈跳的模組宣告
  input kHz,in; //輸入為kHz,in
  output out; //輸出為out
  reg [10:0]d; //保留d[0]~d[10]的值到下一次指定新值
  always@(posedge kHz) //當kHz正緣觸發時，底下的Behavioral Model的敘述會執行
   begin //開始
    d[10]<=d[9]; //d[10]的值變為d[9]的值
    d[9]<=d[8]; //d[9]的值變為d[8]的值
    d[8]<=d[7]; //d[8]的值變為d[7]的值
    d[7]<=d[6]; //d[7]的值變為d[6]的值
    d[6]<=d[5]; //d[6]的值變為d[5]的值
    d[5]<=d[4]; //d[5]的值變為d[4]的值
    d[4]<=d[3]; //d[4]的值變為d[3]的值
    d[3]<=d[2]; //d[3]的值變為d[2]的值
    d[2]<=d[1]; //d[2]的值變為d[1]的值
    d[1]<=d[0]; //d[1]的值變為d[0]的值
    d[0]<=in; //d[0]的值變為in的值
   end //結束
  and(out,d[10],d[9],d[8],d[7],d[6],d[5],d[4],d[3],d[2],d[1],d[0]); //邏輯閘AND，輸入為d[0]~d[10]；輸出為out
endmodule   //消除彈跳的模組結束
```

7. 心得：

404415073 蔡孟勳

這次的實驗我們試了幾個不同的打法，發現 always 中不能有 and，也發現了一些 always 的問題，然後 PS2 也是負緣觸發，所以 buy 的判斷應該是等於 0 時，啟動購買功能，最困難的點大概就是狀態的轉換，幸好最後還是順利完成了。離期末只剩下專題和期末考，希望都能順利完成。