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 11 二十一點

4. 功能說明：

這次的實驗為撰寫一個常見的撲克牌遊戲_二十一點

輸入訊號共有六個，分別為：

Player1 的加牌鍵...第一次為發底牌(不會顯示)，之後依序加牌

Player2 的加牌鍵...第一次為發底牌(不會顯示)，之後依序加牌

Player1 的顯示底牌鍵...可切換目前為底牌或目前發的牌

Player2 的顯示底牌鍵...可切換目前為底牌或目前發的牌

重新開始...可隨時重新開始遊戲

決定結果...雙方都不再加牌後，可按此鍵決定比賽結果。

輸出訊號共有兩個，分別為：

顯示比賽結果

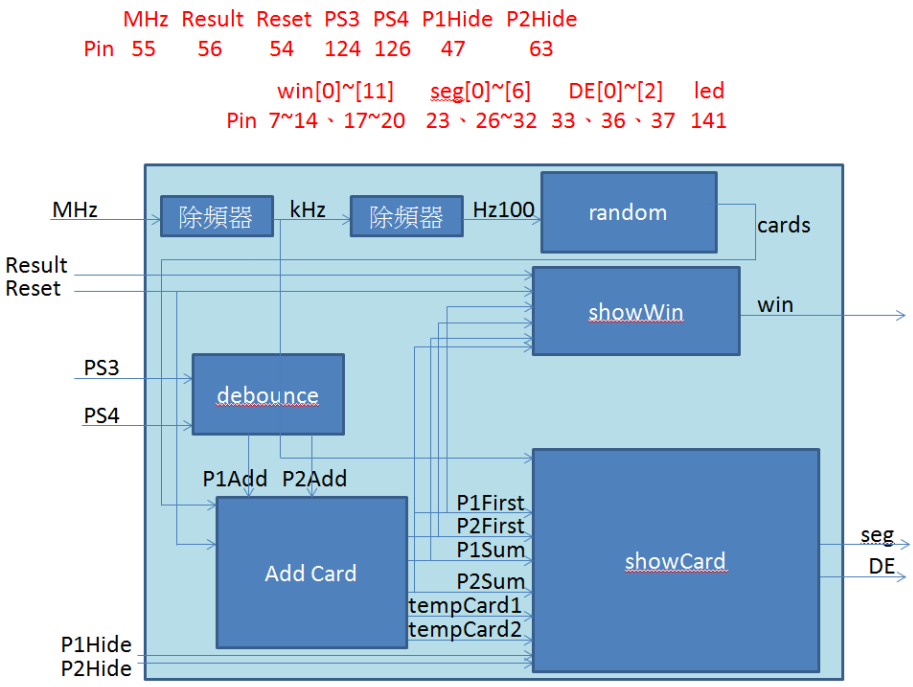
→由 12 顆 LED 以不同形式決定比賽結果。

顯示目前玩家牌的總合及發下的牌或底牌

→由六個七段顯示器顯示，分別為底牌或發下的牌、總和的十位數、總和的個位數，各有 Player1 和 Player2

整個電路的功能為：模擬撲克牌遊戲_二十一點

5. 硬體架構圖：



電路設計的想法:

本次實驗為模擬常見的撲克牌遊戲_二十一點，由於燒錄板只能提供 1MHz 的方波，因此我們設計兩個除頻器，分別降至 1kHz 和 100Hz，1kHz 用來消除彈跳、100Hz 用來隨機產生撲克牌。

再來是處理玩家加牌的部分，我們將 Player1 和 Player2 分開處理，因此不一定要等到 Player1 家玩牌後 Player2 才可以加牌，也不一定要交替進行，方法為：將第一次亂數產生的牌暫存起來 (即為底牌)，之後加的牌依序疊加起來 (即為牌的總和)，當超過 21 點時，加牌功能就會停止無法進行，按下 Reset 鍵則全部歸零 (即重新開始)，而 Player1 和 Player2 的加牌鍵為不同按鍵 (觸發條件不同)，因此可以達到不同步的效果。

最後是顯示的部分，我們只需將加牌部分得到的結果暫存起來，並將之顯示出來即可，總和除 10 取十位數、總和%10 取個位數，並設計一個按鍵，按下後可顯示底牌，為按下則顯示目前亂數產生的牌，最後當按下 Result 後，則可依當時的情況判斷目前誰的總和比較大並顯示相對應的 LED，這樣就可以完成本次實驗了。

6. 程式碼&註解：

```
module exp11(seg, DE, win, led, MHz, Reset, Result, PS3, PS4, P1Hide, P2Hide); //21點模組宣告
 input MHz, Reset, Result, PS3, PS4, P1Hide, P2Hide; //輸入為MHz,Reset,Result,PS3,PS4,P1Hide,P2Hide
 output led; //輸出為led
 output [6:0]seg; //輸出為seg[0]~[6]
 output [2:0]DE; //輸出為DE[0]~[2]
 output [11:0]win; //輸出為win[0]~[11]
 wire kHz, Hz100, P1Add, P2Add; //用kHz,Hz100,P1Add,P2Add，4條線來連接不同模組
 wire [3:0]cards; //用cards[0]~[3]來連接不同模組
 reg [4:0]tempCard1, tempCard2, P1First, P2First, P1Sum, P2Sum;
 //保留tempCard1[0]~[4],tempCard2[0]~[4],P1First[0]~[4],P2First[0]~[4],P1Sum[0]~[4],P2Sum[0]~[4]的值到下一次指定新值
 reg P1count, P2count; //保留P1count,P2count的值到下一次指定新值

 div10000 d1(kHz, MHz); //使用除頻器10000的模組，輸入為MHz，輸出為kHz
 div10    d2(Hz100, kHz); //使用除頻器10的模組，輸入為kHz，輸出為Hz100
 debounce d3(P1Add, kHz, PS3); //使用消除彈跳模組，輸入為kHz,PS3，輸出為P1Add
 debounce d4(P2Add, kHz, PS4); //使用消除彈跳模組，輸入為kHz,PS4，輸出為P2Add

 random(cards, Hz100); //使用隨機的模組，輸入為Hz100，輸出為cards
```

```
 always@(posedge P1Add or negedge Reset) //當P1Add正緣觸發或Reset負緣觸發時，底下的Behavioral Model的敘述會被執行
  begin //開始
   if(~Reset) //如果Reset反相為true，執行以下敘述
    begin //開始
     P1Sum<=0; //P1Sum的值變為0
     P1First<=0; //P1First的值變為0
     P1count<=0; //P1count的值變為0
     tempCard1<=0; //tempCard1的值變為0
    end //結束
   else //其他情況
    begin //開始
     if(P1Sum <= 5'd21) //如果P1Sum的值為21時，執行以下敘述
      begin //開始
       if(P1count == 0) //如果P1count的值為0，執行以下敘述
        begin //開始
         P1First<=cards; //P1First的值變為cards的值
         tempCard1<=5'd0;; //tempCard1的值變為0
         P1count<=1; //P1count的值變為1
        end //結束
       else //其他情況
        begin //開始
         tempCard1<=cards; //tempCard1的值變為cards的值
         P1Sum<=P1Sum+cards; //P1Sum的值變為P1Sum+cards的值
        end //結束
      end //結束
    end //結束
  end //結束
```

```
always@(posedge P2Add or negedge Reset) //當P2Add正緣觸發或Reset負緣觸發時，底下的Behavioral Model的敘述會被執行
 begin //開始
  if(~Reset) //如果Reset反相為true，執行以下敘述
   begin //開始
    P2Sum<=0; //P2Sum的值變為0
    P2First<=0; //P2First的值變為0
    P2count<=0; //P2count的值變為0
    tempCard2<=0; //tempCard2的值變為0
   end //結束
  else //其他情況
   begin //開始
    if(P2Sum <= 5'd21) //如果P2Sum的值為21時，執行以下敘述
     begin //開始
      if(P2count == 0) //如果P2count的值為時，執行以下敘述
       begin //開始
        P2First<=cards; //P2First的值變為cards的值
        tempCard2<=5'd0; //tempCard2的值變為0
        P2count<=1; //P2count的值變為0
       end //結束
      else //其他情況
       begin //開始
        tempCard2<=cards; //tempCard2的值變為cards的值
        P2Sum<=P2Sum+cards; //P2Sum的值變為P2Sum+cards的值
       end //結束
     end //結束
   end //結束
 end //結束
```

```
  //使用showCard模組，輸入為P1Sum,P2Sum,P1First,P2First,P1Hide,P2Hide,kHz,tempCard1,tempCard2，輸出為DE,seg
  showCard s1(DE, seg, P1Sum, P2Sum, P1First, P2First, P1Hide, P2Hide, kHz, tempCard1, tempCard2);

  //使用showWin模組，輸入為Result,Reset,P1Sum,P2Sum,P1First,P2First，輸出為win
  showWin  s2(win, Result, Reset, P1Sum, P2Sum, P1First, P2First);

  assign led=1'b1; //宣告led的值為1

endmodule  //結束模組

module random(counter, Hz100); //隨機的模組宣告
  input Hz100; //輸入為Hz100
  output reg [3:0]counter; //保留輸出counter[0]~[3]的值到下一次指定新值
  always@(posedge Hz100) //當Hz100正緣觸發時，底下的Behavioral Model的敘述會被執行
    if(counter == 4'd10) //如果counter的值為10時，執行以下敘述
     counter<=4'd1; //counter的值變為1
    else //其他情況
     counter<=counter+4'd1; //counter的值+1
endmodule  //結束模組

module showWin(win, Result, Reset, P1Sum, P2Sum, P1First, P2First); //決定輸贏的模組宣告
  input Result,Reset; //輸入為Result,Reset
  input [4:0]P1Sum, P2Sum, P1First, P2First; //輸入為P1Sum[0]~[4],P2Sum[0]~[4],P1First[0]~[4],P2First[0]~[4]
  output reg [11:0]win; //保留輸出win[0]~[11]的值到下一次指定新值
  wire [4:0]P1R,P2R; //用P1R[0]~[4],P2R[0]~[4]連接不同的模組
  assign P1R = P1Sum+P1First; //宣告P1R的值等於P1Sum+P1First的值
  assign P2R = P2Sum+P2First; //宣告P2R的值等於P2Sum+P2First的值

  always@(negedge Result or negedge Reset) //當Result正緣觸發或Reset負緣觸發時，底下的Behavioral Model的敘述會被執行
   begin //開始
    if(~Reset) //如果Reset反相為true，執行以下敘述
     win = 12'b000_000_000_000; //win的值變為000_000_000_000
   if(~Result) //如果Result反相為true，執行以下敘述
    begin //開始
     if(P1R > 5'd21 && P2R > 5'd21) //如果P1R的值大於21且P2R的值大於21，執行以下敘述
      win = 12'b011_111_111_110; //win的值變為011_111_111_110
     else if(P1R > 5'd21) //其他如果P1R的值大於21
      win = 12'b000_000_000_001; //win的值變為000_000_000_001
     else if(P2R > 5'd21) //其他如果P2R的值大於21
      win = 12'b100_000_000_000; //win的值變為100_000_000_000
     else if(P1R > P2R) //其他如果P1R的值大於P2R的值
       win = 12'b100_000_000_000; //win的值變為100_000_000_000
     else if(P1R < P2R) //其他如果P1R的值小於P2R的值
      win = 12'b000_000_000_001; //win的值變為000_000_000_001
     else //其他情況
      win = 12'b011_111_111_110; //win的值變為011_111_111_110
    end //結束
  end //結束
endmodule  //結束模組

module showCard(DE, seg, P1Sum, P2Sum, P1First, P2First, P1Hide, P2Hide, kHz, tempCard1, tempCard2);
  input kHz, P1Hide, P2Hide; //輸入為kHz,P1Hide,P2Hide
  input [4:0]P1Sum, P2Sum, tempCard1, tempCard2, P1First,P2First;
  //輸入為P1Sum[0]~[4],P2Sum[0]~[4],tempCard1[0]~[4],tempCard2[0]~[4],P1First[0]~[4],P2First[0]~[4]
  output reg [2:0]DE; //保留輸出DE[0]~[2]的值到下一次指定新值
  output reg [6:0]seg; //保留輸出seg[0]~[6]的值到下一次指定新值
  reg [4:0]temp1, temp2, temp3, temp4, number;
  //保留temp1[0]~[4],temp2[0]~[4],temp3[0]~[4],temp4[0]~[4], number[0]~[4]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的敘述會被執行
   begin //開始
    if(DE == 3'b110) //如果DE的值為6時，執行以下敘述
     DE<=3'b000; //DE的值變為0
    else //其他情況
     DE<=DE+3'b001; //DE的值+1
  end //結束

  always@(*) //當裡面的值變動時，底下的Behavioral Model的敘述會被執行
   case(DE) //當DE的值為true時，執行以下敘述
    3'b000: //DE的值為0
     begin //開始
      if(P1Hide == 0) //如果P1Hide的值為0，執行以下敘述
       number<=tempCard1; //number的值變為tempCard1的值
      else //其他情況
       number<=P1First; //number的值變為P1First的值
     end //結束

    3'b001: //DE的值為1
     begin //開始
      number<=P1Sum/5'd10; //number的值變為P1Sum/10的值
     end //結束
    3'b010: //DE的值為2
     begin //開始
      number<=P1Sum%5'd10; //number的值變為P1Sum%10的值
     end //結束
    3'b011: //DE的值為3
     begin //開始
      if(P2Hide == 0) //如果P2Hide的值為0，執行以下敘述
       number<=tempCard2; //number的值變為tempCard2的值
      else //其他情況
       number<=P2First; //number的值變為P2First的值
     end //結束
    3'b100: //DE的值為4
     begin //開始
      number<=P2Sum/5'd10; //number的值變為P2Sum/10的值
     end //結束
    3'b101: //DE的值為5
     begin //開始
      number<=P2Sum%5'd10; //number的值變為P2Sum%10的值
     end //結束
   endcase //結束case
```

```
 always@(*) //當裡面的值變動時，底下的Behavioral Model的敘述會被執行
  case(number) //當number的值為true時，執行以下敘述
    5'd0:seg=7'b1111_110; //number的值為0時，七段顯示器顯示0
    5'd1:seg=7'b0110_000; //number的值為1時，七段顯示器顯示1
    5'd2:seg=7'b1101_101; //number的值為2時，七段顯示器顯示2
    5'd3:seg=7'b1111_001; //number的值為3時，七段顯示器顯示3
    5'd4:seg=7'b0110_011; //number的值為4時，七段顯示器顯示4
    5'd5:seg=7'b1011_011; //number的值為5時，七段顯示器顯示5
    5'd6:seg=7'b1011_111; //number的值為6時，七段顯示器顯示6
    5'd7:seg=7'b1110_000; //number的值為7時，七段顯示器顯示7
    5'd8:seg=7'b1111_111; //number的值為8時，七段顯示器顯示8
    5'd9:seg=7'b1111_011; //number的值為9時，七段顯示器顯示9
   default:seg=7'b0001_111; //number的值為其他的值時，顯示倒過來的f
  endcase //結束case
endmodule   //結束模組
```

```
module debounce(out, kHz, in); //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

  //邏輯閘AND，輸入為d[0]~d[10]；輸出為out
  and(out,d[10],d[9],d[8],d[7],d[6],d[5],d[4],d[3],d[2],d[1],d[0]);

 endmodule   //消除彈跳的模組結束
```

```
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 div10(out, in); //除頻器除以10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 [2:0]counter; //保留輸出counter[0]~[2]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==3'd4) //如果counter的值等於4時，執行以下敘述
    begin //開始
     counter<=3'd0; //counter的值變為0
     out<=~out; //out反相
    end //結束
   else //其他情況
    counter<=counter+3'd1; //conuter的值+1
 endmodule   //結束模組
```

7. 心得：

404415073 蔡孟勳

這次的專題，我們抽到了比較簡單的 21 點，在加牌或顯示的邏輯上其實都還好，不難想出來，但因為這次沒有附硬體加構圖，因此若沒有在打程式之前先規劃好，則最後很可能會亂掉或不知道該從何 debug。

經過這次的專題後，我了解到自己其實對 verilog 的語法還不是那麼的熟悉，還是會出錯在一些基本的語法上，也了解到往後在設計更複雜的電路時，應該要先規劃好硬體架構圖 (最好是先畫在紙上)，這樣不僅能讓自己更加清楚自己的程式內容，還能讓未看過程式碼的人能更快了解我們設計的程式。