姓名：陳美瑜

實驗名稱: 擲骰子
功能說明: 兩個人分別擲三顆骰子，按底下所列遊戲規則論贏。
規則一：如果兩人的數字組合都相同，則平手。
規則二：三數目相同>二數目相同>均不相同。
規則三：如果雙方都是三數目相同，則數字大者贏。
規則四：如果雙方都是二數目相同，則相同的數字大者贏，如果相同的數字一樣大，則不相同的數字大者贏。
規則五：如果雙方都是三數字均不相同，則以總合加起來的數字大者贏，如果加起來數字一樣大，則平手。

首先按 Reset 重新開始一局，每個人的三個數字分別用三顆七段顯示器顯示。一開始都顯示 0。每個人都有三個按鈕分別對應擲三顆骰子，按一個按鈕則相對應的七段顯示器顯示擲出來的數字。當兩人共六顆骰子都擲完後，自動判定輸贏，並以 12 顆 LED 顯示跑馬燈顯示誰贏或是平手。當某個骰子擲出數字後，如果再按按鈕不能讓數字變動。 提示：判斷輸贏的方式，可將擲出的 3 個點數，轉成 1 個分數，然後比較兩人的分數即可
硬體架構:
架構圖:

實驗記錄(程式註解)：
主程式(主要指呼叫各個模組)

```
//期末主程式
module fn(l,k1,k2,k3,m1,m2,m3,rst,o1,seq,sel);
input k1,k2,k3,m1,m2,m3,rst,l;
output [6:0]o1;
output reg[2:0]seq;
output [2:0]sel;
wire w1,w2,w3,w4,w5,w6,w7,w8;
wire [2:0]w9,w10,w11,w12,w13,w14,w15;
wire [4:0]w16,w17;
wire [2:0]w18;
div50000 c0(l,w1);
debounce a1(w1,k1,w2);
debounce a2(w1,k2,w3);
debounce a3(w1,k3,w4);
debounce b1(w1,m1,w5);
debounce b2(w1,m2,w6);
debounce b3(w1,m3,w7);
debounce c1(w1,rst,w8);
numgen d1(w2,w1,w9,w8);
numgen d2(w3,w1,w10,w8);
numgen d3(w4,w1,w11,w8);
numgen e1(w5,w1,w12,w8);
numgen e2(w6,w1,w13,w8);
numgen e3(w7,w1,w14,w8);
counter c2(w1,sel);
sixonemux c3(w9,w10,w11,w12,w13,w14,sel,w15);
sev c4(w15,o1);
endmodule
```

```
module sev(in,out);  //七段顯示器
input [2:0]in;  //宣告輸入（搬子數值）
output reg [6:0]out;  //宣告輸出（七段顯示器上七段燈號）
always@(in)  //偵測輸入（搬子數值）
case(in)
3'd1:out=7'b0000110;  //搬子數值為1顯示1
3'd2:out=7'b1011011;  //搬子數值為2顯示2
3'd3:out=7'b1001111;  //搬子數值為3顯示3
3'd4:out=7'b1100110;  //搬子數值為4顯示4
3'd5:out=7'b1101101;  //搬子數值為5顯示5
3'd6:out=7'b1111101;  //搬子數值為6顯示6
default:out=7'b0111111;  //搬子數值初始值1顯示0
endcase
endmodule  //結束module

module plus(in1,in2,in3,in4,in5,in6,o1,o2);  //加法器
input [2:0]in1,in2,in3,in4,in5,in6;  //宣告輸入（六個搬子數值）
output [4:0]o1,o2;  //宣告輸出（兩組搬子個別總和）
assign o1=in1+in2+in3;  //設定輸出（第一組搬子總和）
assign o2=in4+in5+in6;  //設定輸出（第二組搬子總和）
endmodule  //結束module

module sixonemux(in1,in2,in3,in4,in5,in6,s,out);  //六對一多工器
input [2:0]in1,in2,in3,in4,in5,in6;  //宣告輸入（六顆搬子數值）
input [2:0]s;  //宣告輸入（選擇線）
output reg [2:0]out;  //宣告輸出（七段顯示器顯示其一結果）
always@(s)  //偵測選擇線
case(s)
3'b000:out=in1;  //第一個七段顯示器顯示第一顆搬子數值
3'b001:out=in2;  //第二個七段顯示器顯示第二顆搬子數值
3'b010:out=in3;  //第三個七段顯示器顯示第三顆搬子數值
3'b011:out=in4;  //第四個七段顯示器顯示第四顆搬子數值
3'b100:out=in5;  //第五個七段顯示器顯示第五顆搬子數值
3'b101:out=in6;  //第六個七段顯示器顯示第六顆搬子數值
endcase
endmodule  //結束module

module counter(in,out);  //計數器
input in;  //宣告輸入（頻率）
output reg [2:0]out;  宣告暫存輸出（表示5種選擇的3bits輸出）
    always@(posedge in)  //正向輸入觸發
    begin
        if(out==3'b101)  //當輸出為5
                out<=3'b000;  //即歸0
        else
                out<=out+1;  //反之，則不斷加1
        end
endmodule  //結束module
```

```
module debounce(kHz,in,out);  //消除彈跳
input kHz,in;  //宣告輸入（頻率、需消除彈跳之訊號）
output out;  //宣告輸出
reg [6:0]d;  //宣告7bits暫存
always@(posedge kHz)  //以正向頻率觸發
begin
 d<={d[5:0],in};  //將後6bits與輸入訊號組合為新的訊號
end
and(out,d[6],d[5],d[4],d[3],d[2],d[1],d[0]);  //將新的訊號每一bit進行and為輸出結果
endmodule  //結束module
```

```
module div50000(in,out);  //50000除頻器
input in;   //宣告輸入頻率
output reg out;   //宣告輸出頻率
reg [15:0]counter;   //  宣告暫存（計數至24999）
always@(posedge in)  //正向輸入觸發
if (counter==24999)  //若計數至24999
begin
        counter<=0;  //計數歸0
        out<=~out;  //輸出為原五萬分之一
end
else
        counter<=counter+1;  //反之，則不斷計數
endmodule  //結束module
```

```
module numgen(in,clk,out,rst);  //亂數產生器
input in,clk,rst;  //宣告輸入（按鈕輸入、clk、重置）
output reg [2:0]out;  宣告輸出（搬子數值）
reg [2:0]counter;  //宣告暫存（計數器）
reg trig;  //宣告暫存（觸發訊號）
always@(posedge clk)  //偵測clk正向觸發
if (counter==6)  //若計數器為6
        counter<=1;  //計數器為1
else  //反之
        counter<=counter+1;  //計數器加一
always@(negedge in or negedge rst)  //偵測按鈕負向觸發（搬子、重置）
if(~rst)  //若重置輸入不為0
begin
        out<=0;  //輸出數值為0
        trig<=0;  //觸發訊號為0
end
else   //反之
begin
        if(trig==0)  //若觸發訊號為0
        begin
                out<=counter;  //輸出為亂數值
                trig<=1;  //觸發訊號為1
        end
        else //反之
        begin
                out<=out;  //輸出不變
                trig<=trig;  //觸發訊號不變
        end
end
endmodule  //結束module
```

```
//設定一比較模組讓輸贏結果在所有骰子擲完後才進行判斷
compl c5(w9,w10,w11,w12,w13,w14,w18);
always@(w9,w10,w11,w12,w13,w14)
begin
    if(w9==0||w10==0||w11==0||w12==0||w13==0||w14==0)
    begin
    seq<=3'b000;
    end
    else
    begin
    seq<=w18;
    end
end
endmodule
```

```
//初步讀取擲骰子結果的模組，以便確認是要以哪個條件判斷輸贏
module read(in1,in2,in3,out,p,e);
input [2:0]in1,in2,in3;
output reg[2:0]out,p,e;
always@(in1,in2,in3)
begin
if(in1==in2)
begin
    if(in2==in3)//三個骰子結果值一樣
    begin
        out<=3'b100;
        p<=3'b000;
        e<=3'b000;
    end
    else//兩個骰子結果值一樣，另一個不同
        out<=3'b010;
        p<=in2;//存入兩個一樣的骰子的值
        e<=in3;//存入不一樣的那個骰子的值
end
```

```
else if(in2==in3)
begin
     if(in1==in3)//三個骰子結果值一樣
     begin
          out<=3'b100;
          p<=3'b000;
          e<=3'b000;
     end
     else//兩個骰子結果值一樣，另一個不同
          out<=3'b010;
          p<=in2;//存入兩個一樣的骰子的值
          e<=in1;//存入不一樣的那個骰子的值
end
```

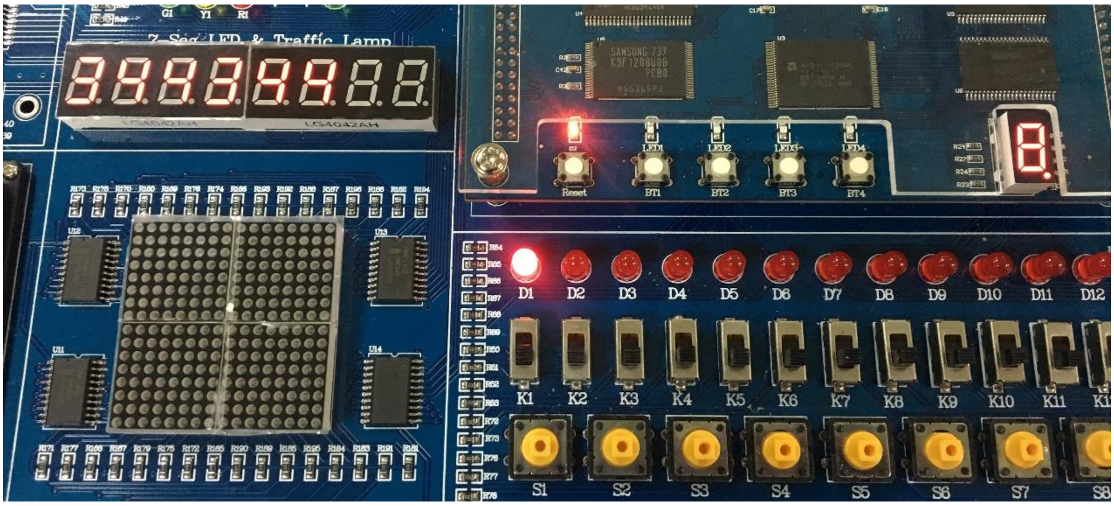
```
else if(in1==in3)
begin
    if(in2==in3)//三個骰子結果值一樣
    begin
        out<=3'b100;
        p<=3'b000;
        e<=3'b000;
    end
    else//兩個骰子結果值一樣，另一個不同
        out<=3'b010;
        p<=in3;//存入兩個一樣的骰子的值
        e<=in2;//存入不一樣的那個骰子的值
end
else//三個骰子結果值都不一樣
begin
    out<=3'b001;
    p<=3'b000;
    e<=3'b000;
end
end
endmodule
```

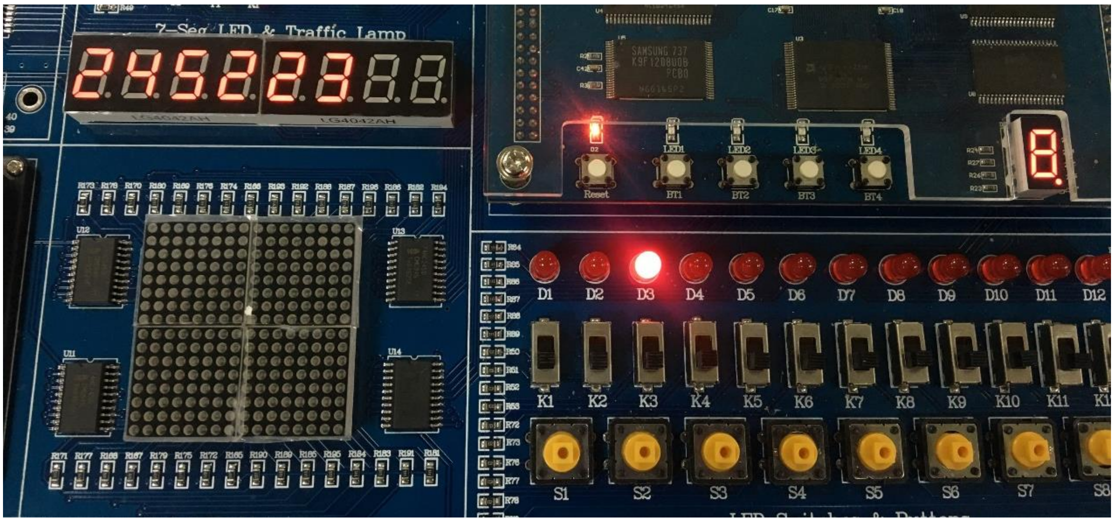
```
//當擲出結果皆為兩個相同，另一個不同時，用於比較的模組
module comp2(p1,p2,e1,e2,out);
output reg[2:0]out;
input[2:0]p1,p2,e1,e2;
always@(p1,p2,e1,e2)
begin
if(p1>p2)//比較兩相同的值的大小
begin
    out<=3'b010;
end
else if(p1<p2)//比較兩相同的值的大小
begin
    out<=3'b001;
end
else if(p1==p2)//兩相同的值的一樣，比較不同的那顆骰子大小
begin
    if(e1>e2)
    begin
        out<=3'b010;
    end
    else if(e1<e2)
    begin
        out<=3'b001;
    end
    else if(e1==e2)
    begin
        out<=3'b100;
    end
end
end
endmodule
```

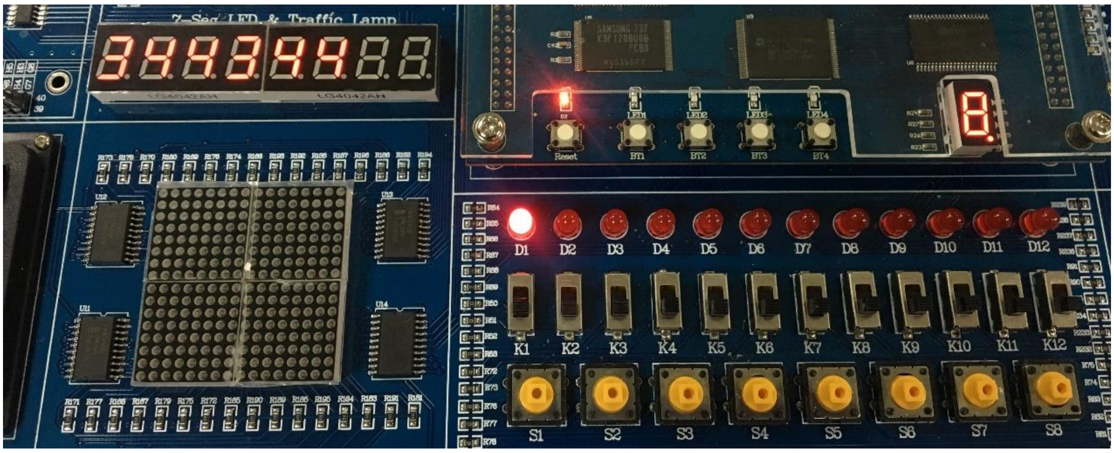
```
//真正做所有輸贏判定的模組
module comp(in1,in2,in3,in4,in5,in6,out);
input [2:0]in1,in2,in3,in4,in5,in6;
output reg[2:0]out;
reg [4:0]o1,o2;
wire [4:0]a1,a2;
wire[2:0]w1,w2,p1,p2,out1,out2,e1,e2;
read R0(in1,in2,in3,w1,p1,e1);//讀取第1位擲骰子結果
read R1(in4,in5,in6,w2,p2,e2);//讀取第2位擲骰子結果
comp2 A0(p1,p2,e1,e2,out2);
plus A1(in1,in2,in3,in4,in5,in6,a1,a2);
comp5 A2(a1,a2,out1);
always@(w1,w2)//先確認是三顆不同、三顆相同還是兩同一異
begin
    if(w1==3'b100)
    begin
        if(w2==3'b100)
        begin
        out<=out1;
        end
        else
        begin
        out=3'b010;
        end
   end
   else if(w2==3'b100)
   begin
        if(w1==3'b100)
        begin
        out<=out1;
        end
        else
        begin
        out=3'b001;
        end
   end
   else if(w1==3'b010)
   begin
     if(w2==3'b010)
  begin
  out<=out2;
  end
  else
  begin
  out<=3'b010;
  end
end
else if(w2==3'b010)
begin
  if(w1==3'b010)
  begin
  out<=out2;
  end
  else
  begin
  out<=3'b001;
  end
end
```

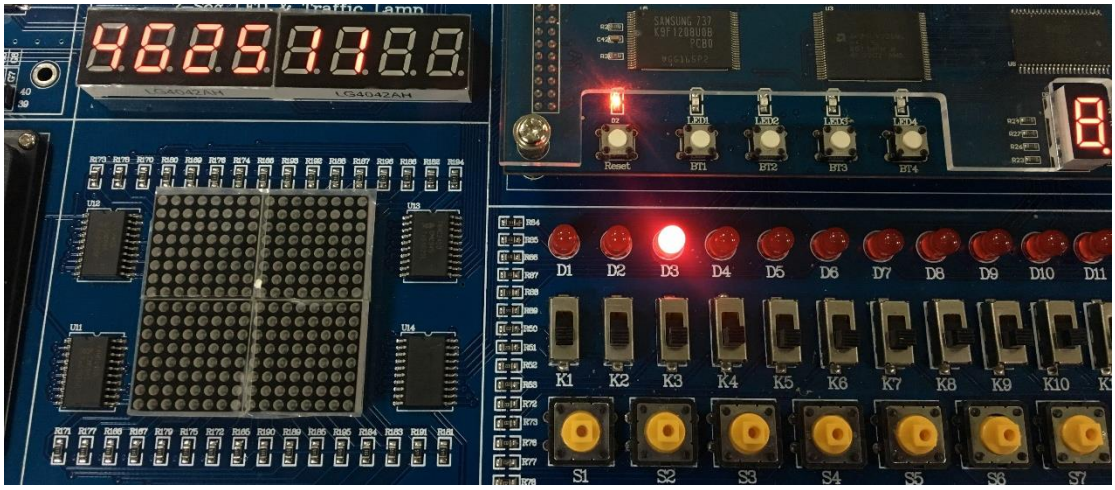
```
    else
    begin
    out<=out1;
    end
end

endmodule
```

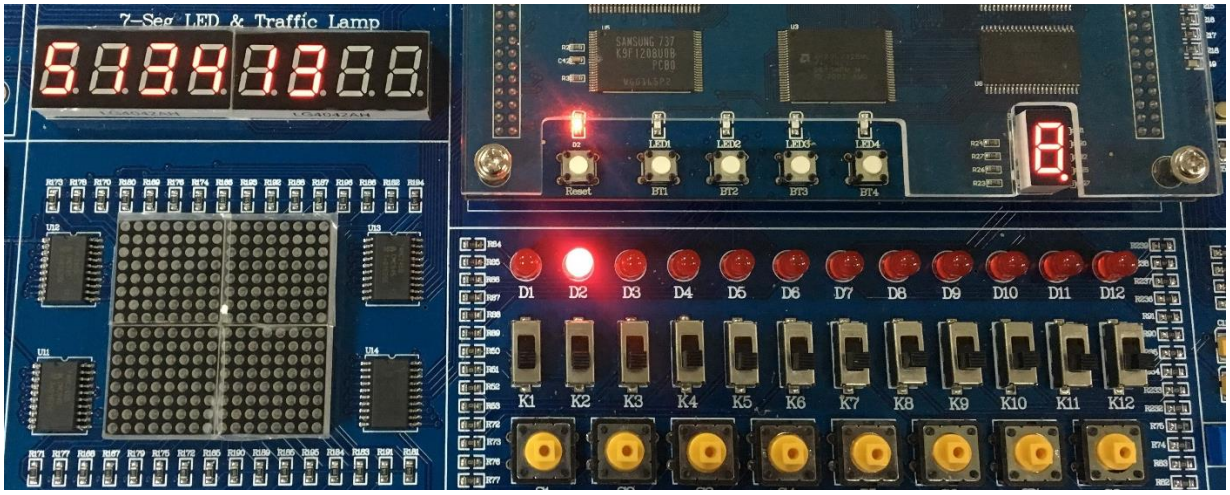
實作成果:

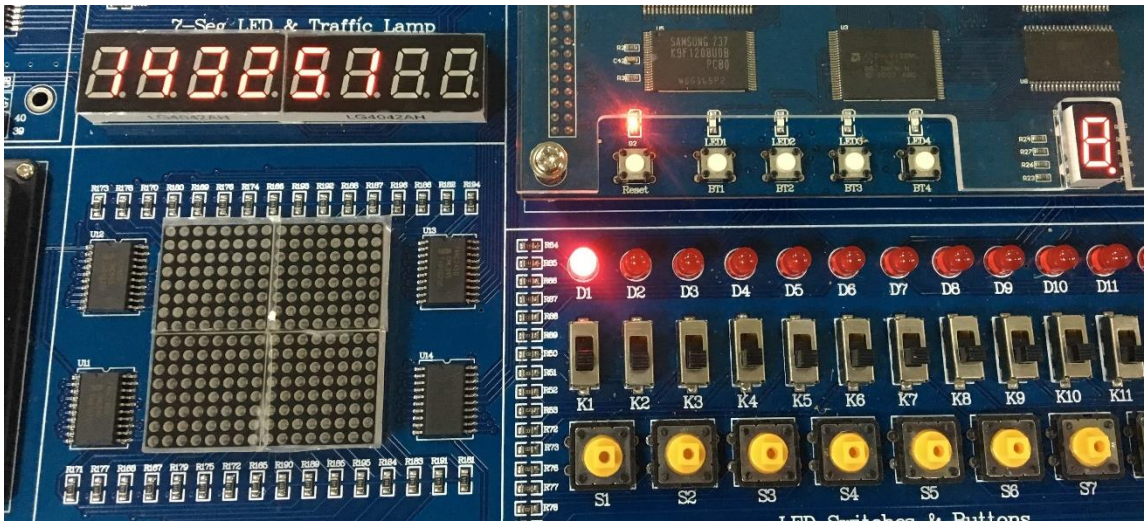


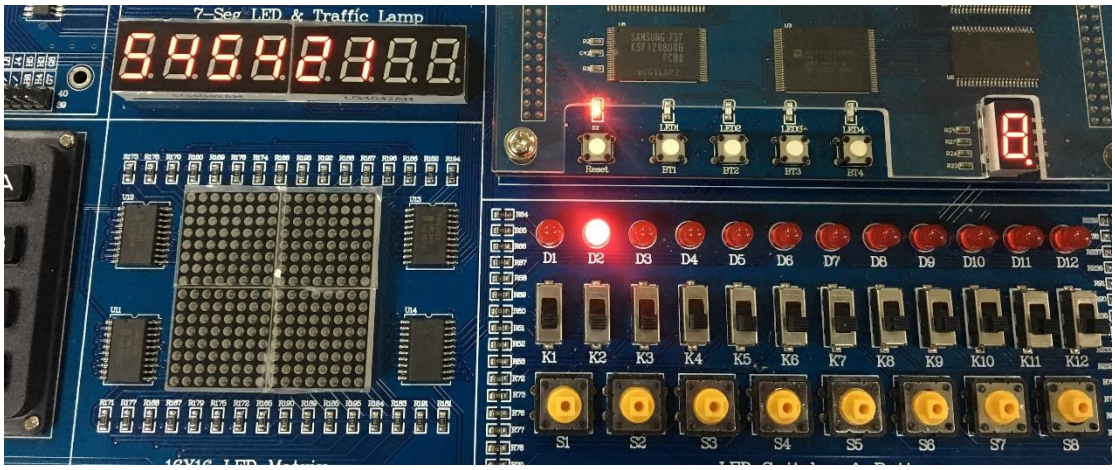


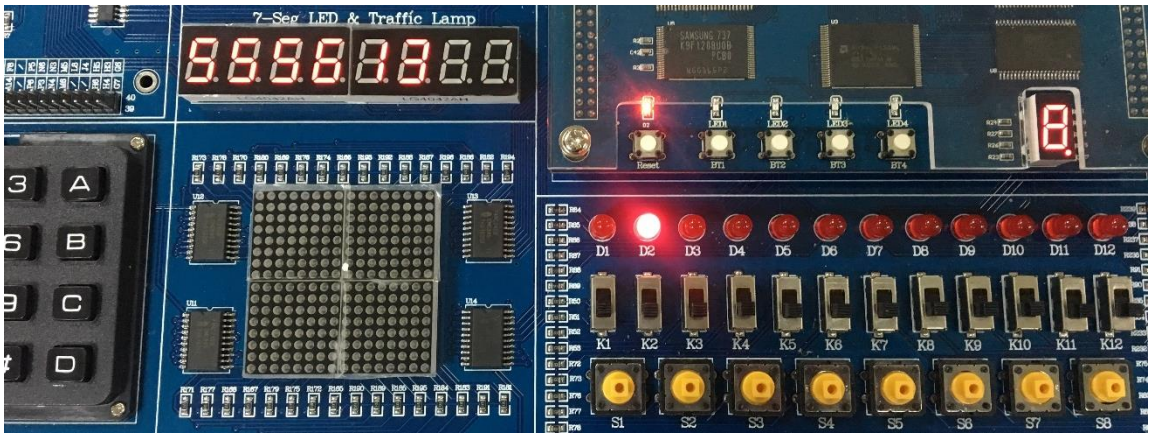



SAMSUNG 737
K9F1208U0B
PCB0
Reset
LED1
LED2
LED3
LED4
BT1
BT2
BT3
BT4
D1 D2 D3 D4 D5 D6 D7 D8 D9 D10 D11
K1 K2 K3 K4 K5 K6 K7 K8 K9 K10
S1 S2 S3 S4 S5 S6 S7


7-Seg LED & Traffic Lamp
SAMSUNG 737
K9F1208U0B
PCB0
Reset
LED1
LED2
LED3
LED4
BT1
BT2
BT3
BT4
D1 D2 D3 D4 D5 D6 D7 D8 D9 D10 D11 D12
K1 K2 K3 K4 K5 K6 K7 K8 K9 K10 K11 K12


7-Seg LED & Traffic Lamp
SAMSUNG 737
K9F1208U0B
PCB0
Reset
LED1
LED2
LED3
LED4
BT1
BT2
BT3
BT4
D1 D2 D3 D4 D5 D6 D7 D8 D9 D10 D11
K1 K2 K3 K4 K5 K6 K7 K8 K9 K10 K11
S1 S2 S3 S4 S5 S6 S7





心得:

學了一學期終於到了最後的期末專題了，真的比以往的電路要求又更複雜了一些，能看的出來老師希望我們活用過去一學期所學 verilog 的使用方式來完每完成這次的實作。

在這分專題上我主要負責輸贏判斷的部分，我先完成一個讀取模組去確認骰子投值解果的狀況再分門別類，針對各個規則寫上比較模組以便進行。

雖然整體看起來很複雜，一開始看到規則有那麼多條件時不知道該如何下手，但冷靜下來思考，一層層的將問題分開去解決，其實就很好理解。

最後想感謝這學期以來一直辛苦跟我奮戰的隊友，也謝謝每堂課都不厭其煩回答我們問題的助教!雖然有時候我們這組卡觀的問題真的很蠢，但助教都願意好好的跟我們解釋，讓我們能夠理解並把每次的電路完成，萬分感謝!