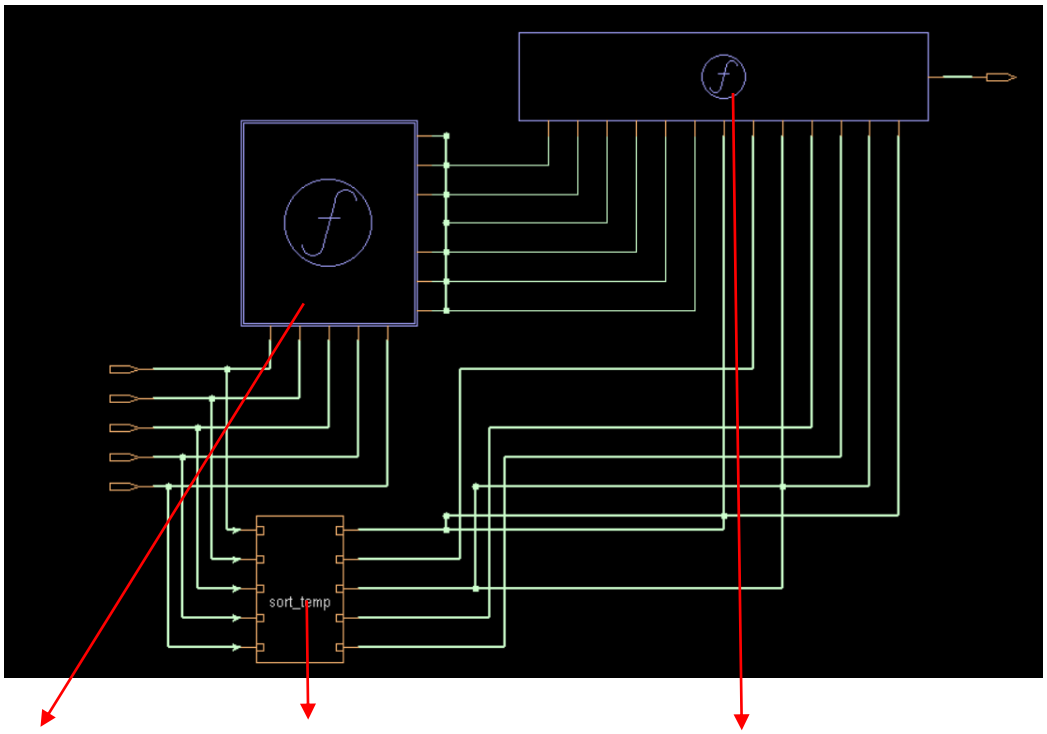
Report_dcs193 HW01 蔡東宏 110511277

1. 架構圖

將此作業分成三部分，第一部分為判斷是否有牌 impossible，第二部分為將牌行進行排序(運用 Lab02 學到的 merge sort)，最後在判斷已排序好的牌為題目的哪一種類型。



第一部分　　第二部分　　第三部分

2. 分段說明:

a. 第一部分:

這部分是用來判斷是否有牌 impossible，我新增了一個 tempx 的陣列(長度為 6)並將 tempx 的值都預設為 0，tempx[0~4]分別記錄五張牌是否 impossible，若 impossible，將對應的 tempx 改成 1。

```
if(hand_n0[5:4]==2'b00)begin
        tempx[0] = (hand_n0[3:0]>4'b0110)? 1:0;
end
else begin
        tempx[0] = (hand_n0[3:0]>4'b1000)? 1:0;
 end
```

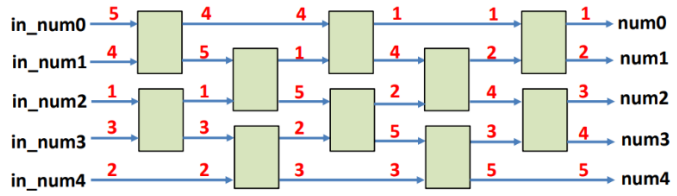
除此之外，還需檢查五張牌是否都長一樣，若都一樣，則將 tempx[5]改成 1。

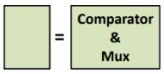
```
32      if((hand_n0==hand_n1)&&(hand_n1==hand_n2)&&(hand_n2==hand_n3)&&(hand_n4==hand_n3))
33              begin
34                      tempx[5]=1;
35              end
```

b. 第二部分:

這部分是將輸入的牌進行排序，我用的排序方法是 Lab02 的 merge sort，這部分我將 sort 寫在另外一個子 module，在主 module 中呼叫這個排序的子 module。

架構圖:





Code:

```
119  module sort_temp(in1,in2,in3,in4,in5,
120                   out1,out2,out3,out4,out5);
121      input [5:0]in1;
122      input [5:0]in2;
123      input [5:0]in3;
124      input [5:0]in4;
125      input [5:0]in5;
126      output logic[5:0]out1;
127      output logic[5:0]out2;
128      output logic[5:0]out3;
129      output logic[5:0]out4;
130      output logic[5:0]out5;
131
132      logic [5:0]temp[20];
133      assign {temp[0],temp[1]}=(in1<in2)?{in1,in2}:{in2,in1};
134      assign {temp[2],temp[3]}=(in3<in4)?{in3,in4}:{in4,in3};
135      assign {temp[4],temp[5]}=(temp[1]<temp[2])?{temp[1],temp[2]}:{temp[2],temp[1]};
136      assign {temp[6],temp[7]}=(temp[3]<in5)?{temp[3],in5}:{in5,temp[3]};
137      assign {temp[8],temp[9]}=(temp[0]<temp[4])?{temp[0],temp[4]}:{temp[4],temp[0]};
138      assign {temp[10],temp[11]}=(temp[5]<temp[6])?{temp[5],temp[6]}:{temp[6],temp[5]};
139      assign {temp[12],temp[13]}=(temp[9]<temp[10])?{temp[9],temp[10]}:{temp[10],temp[9]};
140      assign {temp[14],temp[15]}=(temp[11]<temp[7])?{temp[11],temp[7]}:{temp[7],temp[11]};
141      assign {temp[16],temp[17]}=(temp[8]<temp[12])?{temp[8],temp[12]}:{temp[12],temp[8]};
142      assign {temp[18],temp[19]}=(temp[13]<temp[14])?{temp[13],temp[14]}:{temp[14],temp[13]};
143      assign {out1,out2,out3,out4,out5}={temp[16],temp[17],temp[18],temp[19],temp[15]};
144  endmodule
```

c. 第三部分:

這部分為最後的牌型判斷，第一種為牌中有 impossible 的情況，由第一部分我們得知，當 tempx[0~5]其中一個或多個為 1 時，代表牌中有 impossible 的情況，故輸出為 2’b01。

```
if ((tempx[0]||tempx[1]||tempx[2]||tempx[3]||tempx[4]||tempx[5])==1)
begin
  out_data = 2'b01;
end
```

第二種為牌型為 a triplet and a pair，由於已經排好順序了，所以一對的只有可能為前兩張或後兩張，並且其他三張為 triplet。在此情況下輸出 2’b11。

```
if((o1==o2)&&(o2==o3) && (o4==o5))
  out_data = 2'b11 ;
else if((o1==o2) && (o3==o4)&&(o4==o5))
  out_data = 2'b11 ;
```

第三種牌型為 a sequence and a pair，由於已經排好序了，所以如果有 sequence，他一定是連續的三張牌，剩下的兩張即為一對，除此之外，因為字牌不會有 sequence，所以在判斷的時候，還須加上連續的三張(但只需要判斷一張即可)不為字牌的條件，在這種情況下輸出 2'b10。

```
else if((o1==o2)&&((o3+2)==(o4+1)&&(o4+1)==(o5))&&o3[5:4]!=2'b00)
  out_data = 2'b10;
else if((o2==o3)&&((o1+2)==(o4+1)&&(o4+1)==(o5))&&o1[5:4]!=2'b00)
  out_data = 2'b10;
else if((o3==o4)&&((o1+2)==(o2+1)&&(o2+1)==(o5))&&o1[5:4]!=2'b00)
  out_data = 2'b10;
else if((o4==o5)&&((o1+2)==(o2+1)&&(o2+1)==(o3))&&o1[5:4]!=2'b00)
  out_data = 2'b10;
```

第四種為最後一種情況，在上面的條件都不成立的情況下，即為此條件，輸出 2'b00。

3. 心得報告:

我在這次的 HW 遇到最大的困難是在壓面積的時候，第一次做完的時候面積比很多同學大，所以我試了滿多方法去優化電路，雖然滿多時後覺得好像有比較優化，但最後的面積反而是變大的，希望助教可以教一些優化電路的方法。除此之外，一開始的時候，我在跑合成中遇到 Latch，後來在 Lab02 的時候問了助教才知道是因為我宣告的變數中有一些值沒有初始化，而我的 code 又讓那些值在某個特定的條件下才給它值，所以當那個條件沒有成立的時候，我的變數就會不知道是多少，所以解決的方法就是將他們全部初始化。