數電HW1結報

110511232 徐培哲

電路簡介

設計概念

此次作業要做的是判斷麻將胡牌的電路，判斷的過程可以分為以下部分:

1. 將所有牌排序
2. 判斷是否為非法項
3. 判斷是否為triplet+pair
4. 判斷是否為sequence+pair

將所有牌由小排到大後，會使的我們判斷sequence和pair時需要考慮的狀況大幅減少，而2~4各項判斷的細部實現方式則會在下文詳細說明，。

1. 非法項判斷

```
// Check if input contains invalid terms
// sorted_hand為已排序的輸入

  for(int i=0;i<5;i++) begin
      //判斷目前檢視的牌是否是字牌
      is_honor[i] = (sorted_hand[i][5:4]==2'b00);

      //針對該牌的類型，以相對應的判斷方式處理
      //字牌->不可>6
      //非字牌->不可>8
      invalid_check[i] = (is_honor[i] ? sorted_hand[i][3:0]>6 : sorted_hand[i][3:0]>8);
  end

  // 若任何一項的數字過大，或是五張牌皆相同，則判斷為invalid
  // equal_02 = sorted_hand[0]==sorted_hand[2];
  // equal_24 = sorted_hand[2]==sorted_hand[4];
  invalid_term_exist = invalid_check[0] || invalid_check[1] || invalid_check[2] || invalid_check[3] || invalid_check[4] || (equal_02 &
```

非法項的情況分為2種:

(1) 五張牌皆相同

由於我們已經將輸入的五個數字全部排序過了，所以如果sorted_hand[0]、sorted_hand[4]相同，我們就能確定其他三張牌也是同一個數字。但由於sorted_hand[0]、sorted_hand[2]以及sorted_hand[2]、sorted_hand[4]之間的比較(equal_02、equal_24)在判斷sequence時也會用到，為了節省面積，在程式中選擇用這兩項重複利用，藉此判斷5張牌是否相等。

(2) 至少一張牌的數字過大

這裡還要再分成字牌和非字牌兩種情況分析，因此首先要先判斷對應這張牌的輸入中，代表牌型的最高位2位元是否是2’b00。如果是字牌，則判斷是否>6，否則判斷是否>8。這個步驟會透過for迴圈，將所有輸入檢查過一次。而由於是否是字牌的判斷在之後判斷 sequence+pair 時還會用到，在此將其存入一個logic變數中，方便重複使用。

最後將兩種可能情況的判斷結果取or，就可得出是否有非法項被輸入。

2. triplet + pair 判斷

```
// these indicate a pair
equal_01 = sorted_hand[0]==sorted_hand[1];
equal_34 = sorted_hand[3]==sorted_hand[4];

// these indicate a triplet
equal_02 = sorted_hand[0]==sorted_hand[2];
equal_24 = sorted_hand[2]==sorted_hand[4];

tri_012 = equal_02;
tri_234 = equal_24;

if((tri_012 && pair_34)||(tri_234 && pair_01))begin
    out_data = 2'b11;
end
```

由於排序，輸入中相同的項必定相鄰，所以此種情況只有兩種可能

```
 ___   _
|   | | |
A A A B B
4 3 2 1 0

or
 _   ___
| | |   |
B B A A A
4 3 2 1 0
```

但因為排序，我們在判斷 triplet 時只要判斷 triplet 頭尾兩個數字是否相等即可，因為如果頭尾相等，中間那項也必定相等。因此只需要判斷 equal_02、equal_24即可。

3. sequence + pair 判斷

```
// these indicate a pair
equal_01 = sorted_hand[0]==sorted_hand[1];
equal_34 = sorted_hand[3]==sorted_hand[4];

// triplet inside a sequence
equal_13 = sorted_hand[1]==sorted_hand[3];

// 01_12(!is_honor[2]), 12_23(!is_honor[2]), 23_34(!is_honor[2]) indicates a seq
diff_01 = sorted_hand[1]==sorted_hand[0]+1;
diff_12 = sorted_hand[2]==sorted_hand[1]+1;
diff_23 = sorted_hand[3]==sorted_hand[2]+1;
diff_34 = sorted_hand[4]==sorted_hand[3]+1;

pair_01 = equal_01;
pair_34 = equal_34;

seq_012 = diff_01 & diff_12;
seq_234 = diff_23 & diff_34;
seq_01_34 = diff_01 & diff_34 & equal_13;

if(is_honor[2])begin
    out_data = 2'b00;
end
else if((pair_01 && seq_234)||(seq_012 && pair_34)||(seq_01_34 &&tri_123))begin
    out_data = 2'b10;
end
```

此情況有3種可能:

```
//A、B、C 與相鄰項各差一，即 A = B-1, B = C-1，且其中任一項不得為字牌

A B C D D
4 3 2 1 0

or

D D A B C
4 3 2 1 0

or

A B B B C
4 3 2 1 0
```

首先我們要先確定 sequence 中，沒有任何一項是字牌，從上表可以發現，sequence 的三項中必定包含sorted_hand[2]，所以只要sorted_hand[0]是字牌，就不可能滿足 sequence+pair。而字牌的判斷(is_honor)在判斷非法項時就已經做過了，所以在此就直接把它重複運用，減少面積。

過濾掉 sequence 中有字牌的狀況後，我們就判斷牌的數字分布是否為三種情況中任何一種，而這些判斷有些在其他部分的程式中也有用到，在此也將它重複運用來減少面積，就可以得出是否有 sequence+pair。

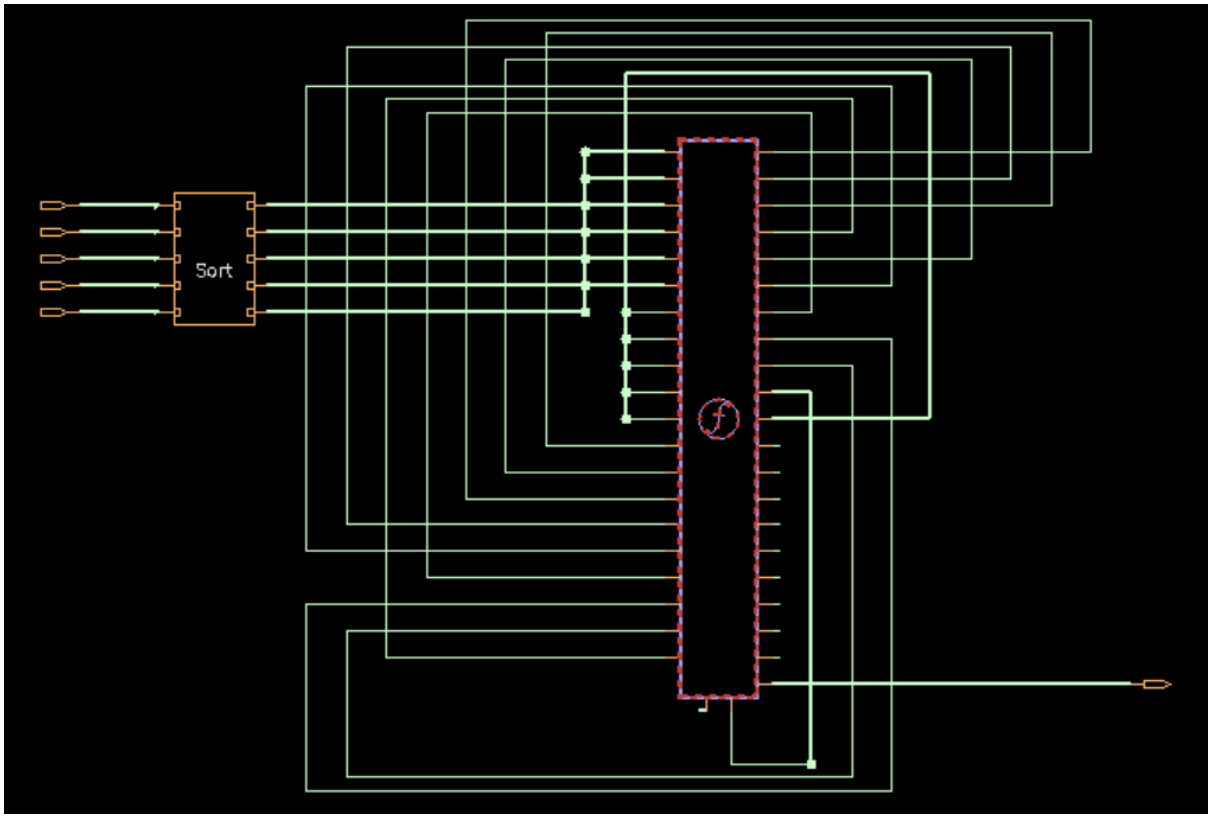
最後輸出判斷

程式最後綜合以上判斷結果，決定輸出值為何。

```
// 非法項判斷
if(invalid_term_exist)begin
    out_data = 2'b01;
end

// triplet + pair
else if((tri_012 && pair_34)||(tri_234 && pair_01))begin
    out_data = 2'b11;
end

// 非 triplet + pair，只剩 sequence + pair 這種胡牌條件
// 先過濾掉 sequence 中含有字牌的狀況
else if(is_honor[2])begin
    out_data = 2'b00;
end
// 再判斷數字分布是否符合 sequence + pair
else if((pair_01 && seq_234)||(seq_012 && pair_34)||(seq_01_34 &&tri_123))begin
    out_data = 2'b10;
end

// 沒有胡牌也沒有非法項
else begin
    out_data = 2'b00;
end
```

電路圖

1. verdi 電路圖



合成報告

面積: 5501.865664

slack (MET)　14.63

▼ syn.log

```
Design Compiler Graphical
                                DC Ultra (TM)
                                 DFTMAX (TM)
                             Power Compiler (TM)
                                DesignWare (R)
                                DC Expert (TM)
                              Design Vision (TM)
                              HDL Compiler (TM)
                              VHDL Compiler (TM)
                                 DFT Compiler
                              Design Compiler(R)

                 Version R-2020.09 for linux64 - Aug 26, 2020

                    Copyright (c) 1988 - 2020 Synopsys, Inc.
   This software and the associated documentation are proprietary to Synopsys,
 Inc. This software may only be used in accordance with the terms and conditions
 of a written license agreement with Synopsys, Inc. All other use, reproduction,
            or distribution of this software is strictly prohibited.
Initializing...
#======================================================
#
# Synopsys Synthesis Scripts (Design Vision dctcl mode)
#
#======================================================
#======================================================
#  Set Libraries
#======================================================
set search_path {       ./../01_RTL                                     ~iclabta01/umc018/Synthesis/                    /usr/synt
 ./../01_RTL      ~iclabta01/umc018/Synthesis/      /usr/synthesis/libraries/syn/      /usr/synthesis/dw/
set synthetic_library {dw_foundation.sldb}
dw_foundation.sldb
set link_library {* dw_foundation.sldb standard.sldb slow.db}
* dw_foundation.sldb standard.sldb slow.db
set target_library {slow.db}
slow.db
#======================================================
```

```
#
# Synopsys Synthesis Scripts (Design Vision dctcl mode)
#
#====================================================
#====================================================
#  Global Parameters
#====================================================
# set DESIGN "SMJ"
# set MAX_DELAY 12
#====================================================
#  Read RTL Code
#====================================================
read_sverilog { SMJ.sv}
Loading db file '/usr/synthesis/libraries/syn/dw_foundation.sldb'
Loading db file '/usr/synthesis/libraries/syn/standard.sldb'
Loading db file '/RAID2/COURSE/iclab/iclabta01/umc018/Synthesis/slow.db'
Loading db file '/RAID2/EDA/synopsys/synthesis/2020.09/libraries/syn/gtech.db'
Loading db file '/RAID2/EDA/synopsys/synthesis/2020.09/libraries/syn/standard.sldb'
  Loading link library 'slow'
  Loading link library 'gtech'
Loading sverilog file '/RAID2/COURSE/dcs/dcs166/HW01/01_RTL/SMJ.sv'
Detecting input file type automatically (-rtl or -netlist).
Reading with Presto HDL Compiler (equivalent to -rtl option).
Running PRESTO HDLC
Compiling source file /RAID2/COURSE/dcs/dcs166/HW01/01_RTL/SMJ.sv
Presto compilation completed successfully.
Current design is now '/RAID2/COURSE/dcs/dcs166/HW01/01_RTL/Sort.db:Sort'
Loaded 2 designs.
Current design is 'Sort'.
Sort SMJ
current_design "SMJ"
Current design is 'SMJ'.
{SMJ}
#====================================================
#  Global Setting
#====================================================
#set_wire_load_mode top
#====================================================
#  Set Design Constraints
#====================================================
#create_clock -name "CLK" -period $CLK_period CLK
#set_input_delay  [ expr $CLK_period*0.5 ] -clock CLK [all_inputs]
#set_output_delay [ expr $CLK_period*0.5 ] -clock CLK [all_outputs]
#set_input_delay 0 -clock CLK CLK
#set_load 0.05 [all_outputs]
set_max_delay 30 -from [all_inputs] -to [all_outputs]
1
#====================================================
#  Optimization
#====================================================
uniquify
1
set_fix_multiple_port_nets -all -buffer_constants
1
#set_fix_hold [all_clocks]
compile_ultra
Information: Performing power optimization. (PWR-850)
Alib files are up-to-date.
Information: Evaluating DesignWare library utilization. (UISN-27)

============================================================================
| DesignWare Building Block Library  |         Version         | Available |
============================================================================
| Basic DW Building Blocks           | Q-2019.12-DWBB_201912.0 |     *     |
| Licensed DW Building Blocks        | Q-2019.12-DWBB_201912.0 |     *     |
============================================================================

Information: Sequential output inversion is enabled.  SVF file must be used for formal verification. (OPT-1208)

Information: There are 10 potential problems in your design. Please run 'check_design' for more information. (LINT-99)

  Simplifying Design 'SMJ'

Loaded alib file './alib-52/slow.db.alib'
  Building model 'DW01_NAND2'
Information: Ungrouping hierarchy sort_ins before Pass 1 (OPT-776)
Information: Ungrouping 1 of 2 hierarchies before Pass 1 (OPT-775)
Information: State dependent leakage is now switched from on to off.

  Beginning Pass 1 Mapping
  ------------------------
  Processing 'SMJ'
Information: Added key list 'DesignWare' to design 'SMJ'. (DDB-72)
 Implement Synthetic for 'SMJ'.

  Updating timing information
Information: Updating design information... (UID-85)
```

```
Information: The library cell 'HOLDX1' in the library 'slow' is not characterized for internal power. (PWR-536)
Information: The target library(s) contains cell(s), other than black boxes, that are not characterized for internal power. (PWR-2

  Beginning Mapping Optimizations  (Ultra High effort)
  -------------------------------
Information: There is no timing violation in design SMJ. Delay-based auto_ungroup will not be performed. (OPT-780)

                                  TOTAL
   ELAPSED            WORST NEG   SETUP    DESIGN                              LEAKAGE
    TIME      AREA      SLACK     COST    RULE COST         ENDPOINT            POWER
  --------- --------- --------- --------- --------- ------------------------- ---------
    0:00:02    5694.8      0.00       0.0       0.0                           511615.7188
    0:00:02    5694.8      0.00       0.0       0.0                           511615.7188

  Beginning Constant Register Removal
  -----------------------------------
    0:00:03    5694.8      0.00       0.0       0.0                           511615.7188
    0:00:03    5694.8      0.00       0.0       0.0                           511615.7188

  Beginning Global Optimizations
  ------------------------------
  Numerical Synthesis (Phase 1)
  Numerical Synthesis (Phase 2)
  Global Optimization (Phase 1)
  Global Optimization (Phase 2)
  Global Optimization (Phase 3)
  Global Optimization (Phase 4)
  Global Optimization (Phase 5)
  Global Optimization (Phase 6)
  Global Optimization (Phase 7)
  Global Optimization (Phase 8)
  Global Optimization (Phase 9)
  Global Optimization (Phase 10)
  Global Optimization (Phase 11)
  Global Optimization (Phase 12)
  Global Optimization (Phase 13)
  Global Optimization (Phase 14)
  Global Optimization (Phase 15)
  Global Optimization (Phase 16)
  Global Optimization (Phase 17)
  Global Optimization (Phase 18)
  Global Optimization (Phase 19)
  Global Optimization (Phase 20)
  Global Optimization (Phase 21)
  Global Optimization (Phase 22)
  Global Optimization (Phase 23)
  Global Optimization (Phase 24)
  Global Optimization (Phase 25)
  Global Optimization (Phase 26)
  Global Optimization (Phase 27)
  Global Optimization (Phase 28)

  Beginning Isolate Ports
  -----------------------

  Beginning Delay Optimization
  ----------------------------
    0:00:03    5485.2      0.00       0.0       0.0                           422887.2500
    0:00:03    5485.2      0.00       0.0       0.0                           422887.2500
    0:00:03    5485.2      0.00       0.0       0.0                           422887.2500
    0:00:03    5485.2      0.00       0.0       0.0                           422887.2500
    0:00:03    5478.6      0.00       0.0       0.0                           421352.3125
    0:00:03    5478.6      0.00       0.0       0.0                           421352.3125

  Beginning WLM Backend Optimization
  --------------------------------------
    0:00:04    5461.9      0.00       0.0       0.0                           419523.0938
    0:00:04    5461.9      0.00       0.0       0.0                           419523.0938
    0:00:04    5461.9      0.00       0.0       0.0                           419523.0938
    0:00:04    5455.3      0.00       0.0       0.0                           363592.6250
    0:00:04    5455.3      0.00       0.0       0.0                           363592.6250
    0:00:04    5455.3      0.00       0.0       0.0                           363592.6250
    0:00:04    5455.3      0.00       0.0       0.0                           363592.6250
    0:00:04    5455.3      0.00       0.0       0.0                           363592.6250
    0:00:04    5455.3      0.00       0.0       0.0                           363592.6250
    0:00:04    5455.3      0.00       0.0       0.0                           363592.6250
    0:00:04    5455.3      0.00       0.0       0.0                           363592.6250
    0:00:04    5455.3      0.00       0.0       0.0                           363592.6250
    0:00:04    5455.3      0.00       0.0       0.0                           363592.6250
    0:00:04    5455.3      0.00       0.0       0.0                           363592.6250
    0:00:04    5455.3      0.00       0.0       0.0                           363592.6250
    0:00:04    5455.3      0.00       0.0       0.0                           363592.6250
    0:00:04    5455.3      0.00       0.0       0.0                           363592.6250
    0:00:04    5455.3      0.00       0.0       0.0                           363592.6250
    0:00:04    5455.3      0.00       0.0       0.0                           363592.6250
    0:00:04    5455.3      0.00       0.0       0.0                           363592.6250
    0:00:04    5455.3      0.00       0.0       0.0                           363592.6250
```

```
    0:00:04    5455.3      0.00       0.0       0.0                           363592.6250
    0:00:04    5455.3      0.00       0.0       0.0                           363592.6250
    0:00:04    5455.3      0.00       0.0       0.0                           363592.6250


  Beginning Leakage Power Optimization  (max_leakage_power 0)
  ------------------------------------

                                  TOTAL
   ELAPSED            WORST NEG   SETUP    DESIGN                              LEAKAGE
    TIME      AREA      SLACK     COST    RULE COST         ENDPOINT            POWER
  --------- --------- --------- --------- --------- ------------------------- ---------
    0:00:04    5455.3      0.00       0.0       0.0                           363592.6250
  Global Optimization (Phase 29)
  Global Optimization (Phase 30)
  Global Optimization (Phase 31)
  Global Optimization (Phase 32)
  Global Optimization (Phase 33)
  Global Optimization (Phase 34)
  Global Optimization (Phase 35)
  Global Optimization (Phase 36)
  Global Optimization (Phase 37)
  Global Optimization (Phase 38)
  Global Optimization (Phase 39)
    0:00:04    5548.4      0.00       0.0       0.0                           357694.6562
    0:00:04    5548.4      0.00       0.0       0.0                           357694.6562
    0:00:04    5548.4      0.00       0.0       0.0                           357694.6562
    0:00:04    5455.3      0.00       0.0       0.0                           363592.6250
    0:00:04    5455.3      0.00       0.0       0.0                           363592.6250
    0:00:04    5455.3      0.00       0.0       0.0                           363592.6250
    0:00:04    5455.3      0.00       0.0       0.0                           363592.6250
    0:00:04    5455.3      0.00       0.0       0.0                           363592.6250
    0:00:04    5455.3      0.00       0.0       0.0                           363592.6250
    0:00:04    5455.3      0.00       0.0       0.0                           363592.6250
    0:00:04    5455.3      0.00       0.0       0.0                           363592.6250
    0:00:04    5455.3      0.00       0.0       0.0                           363592.6250
    0:00:04    5455.3      0.00       0.0       0.0                           363592.6250
    0:00:04    5455.3      0.00       0.0       0.0                           363592.6250
    0:00:04    5455.3      0.00       0.0       0.0                           363592.6250
    0:00:04    5455.3      0.00       0.0       0.0                           363592.6250
    0:00:04    5455.3      0.00       0.0       0.0                           363592.6250
    0:00:04    5455.3      0.00       0.0       0.0                           363592.6250
    0:00:04    5455.3      0.00       0.0       0.0                           363592.6250
    0:00:04    5455.3      0.00       0.0       0.0                           363592.6250
    0:00:04    5455.3      0.00       0.0       0.0                           363592.6250
    0:00:04    5455.3      0.00       0.0       0.0                           363592.6250

                                  TOTAL
   ELAPSED            WORST NEG   SETUP    DESIGN                              LEAKAGE
    TIME      AREA      SLACK     COST    RULE COST         ENDPOINT            POWER
  --------- --------- --------- --------- --------- ------------------------- ---------
    0:00:04    5455.3      0.00       0.0       0.0                           363592.6250
    0:00:04    5455.3      0.00       0.0       0.0                           363592.6250
    0:00:04    5455.3      0.00       0.0       0.0                           363592.6250
    0:00:04    5455.3      0.00       0.0       0.0                           363592.6250
    0:00:04    5455.3      0.00       0.0       0.0                           363592.6250
    0:00:04    5501.9      0.00       0.0       0.0                           360643.6250
    0:00:04    5501.9      0.00       0.0       0.0                           360643.6250
    0:00:04    5501.9      0.00       0.0       0.0                           360643.6250
    0:00:04    5501.9      0.00       0.0       0.0                           360643.6250
    0:00:04    5501.9      0.00       0.0       0.0                           360643.6250
    0:00:04    5501.9      0.00       0.0       0.0                           360643.6250
Loading db file '/RAID2/COURSE/iclab/iclabta01/umc018/Synthesis/slow.db'


Note: Symbol # after min delay cost means estimated hold TNS across all active scenarios


  Optimization Complete
  ---------------------
Information: State dependent leakage is now switched from off to on.
Information: Propagating switching activity (low effort zero delay simulation). (PWR-6)
Warning: There is no defined clock in the design. (PWR-80)
1
#======================================================
#  Output Reports
#======================================================
report_timing >  Report/SMJ.timing
report_area >  Report/SMJ.area
report_resource >  Report/SMJ.resource
#======================================================
#  Change Naming Rule
#======================================================
set bus_inference_style "%s\[%d\]"
%s[%d]
set bus_naming_style "%s\[%d\]"
%s[%d]
```

```
set hdlout_internal_busses true
true
change_names -hierarchy -rule verilog
1
define_name_rules name_rule -allowed "a-z A-Z 0-9 _" -max_length 255 -type cell
1
define_name_rules name_rule -allowed "a-z A-Z 0-9 _[]" -max_length 255 -type net
1
define_name_rules name_rule -map {{"\\*cell\\*" "cell"}}
1
change_names -hierarchy -rules name_rule
1
#====================================================
#  Output Results
#====================================================
set verilogout_higher_designs_first true
true
write -format verilog -output Netlist/SMJ_SYN.v -hierarchy
Writing verilog file '/RAID2/COURSE/dcs/dcs166/HW01/02_SYN/Netlist/SMJ_SYN.v'.
1
write_sdf -version 2.1 -context verilog -load_delay cell Netlist/SMJ_SYN.sdf
Information: Writing timing information to file '/RAID2/COURSE/dcs/dcs166/HW01/02_SYN/Netlist/SMJ_SYN.sdf'. (WT-3)
1
#====================================================
#  Finish and Quit
#====================================================
report_timing

****************************************
Report : timing
        -path full
        -delay max
        -max_paths 1
Design : SMJ
Version: R-2020.09
Date   : Thu Mar 16 00:16:00 2023
****************************************

Operating Conditions: slow   Library: slow
Wire Load Model Mode: top

  Startpoint: hand_n4[0] (input port)
  Endpoint: out_data[1]
            (output port)
  Path Group: default
  Path Type: max

  Point                                    Incr       Path
  -----------------------------------------------------------
  input external delay                     0.00       0.00 f
  hand_n4[0] (in)                          0.00       0.00 f
  U367/Y (INVXL)                           0.10       0.10 r
  U368/Y (AOI222XL)                        0.24       0.34 f
  U370/Y (AOI222XL)                        0.54       0.88 r
  U372/Y (AOI222XL)                        0.29       1.17 f
  U373/Y (OAI2BB2XL)                       0.35       1.52 f
  U374/Y (AOI31XL)                         0.96       2.48 r
  U375/Y (MX2XL)                           0.63       3.11 f
  U387/Y (AOI222XL)                        0.60       3.71 r
  U388/Y (AOI222XL)                        0.31       4.03 f
  U389/Y (AOI222XL)                        0.54       4.57 r
  U390/Y (AOI21XL)                         0.13       4.70 f
  U391/Y (OAI31XL)                         0.28       4.97 r
  U392/Y (OAI2BB1XL)                       0.16       5.13 f
  U393/Y (AOI21XL)                         0.84       5.97 r
  U445/Y (MXI2XL)                          0.55       6.51 f
  U448/Y (AOI222XL)                        0.62       7.13 r
  U450/Y (AOI222XL)                        0.34       7.47 f
  U451/Y (AOI222XL)                        0.55       8.02 r
  U452/Y (AOI2BB1XL)                       0.14       8.16 f
  U454/Y (OAI2BB1XL)                       0.25       8.41 f
  U455/Y (OAI31XL)                         0.17       8.58 r
  U456/Y (OAI2BB1XL)                       0.54       9.12 f
  U477/Y (MXI2XL)                          0.44       9.56 r
  U478/Y (AOI222XL)                        0.34       9.90 f
  U481/Y (AOI222XL)                        0.47      10.37 r
  U482/Y (AOI21XL)                         0.19      10.55 f
  U484/Y (AOI222XL)                        0.52      11.08 r
  U485/Y (AOI222XL)                        0.46      11.53 f
  U494/Y (INVXL)                           0.44      11.97 r
  U319/Y (NOR2XL)                          0.23      12.20 f
  U567/Y (AOI222XL)                        0.98      13.19 r
  U587/Y (XOR2XL)                          0.67      13.85 r
  U595/Y (MX2XL)                           0.33      14.18 r
  U596/Y (OAI211XL)                        0.16      14.34 f
  U614/Y (AOI211XL)                        0.23      14.57 r
  U615/Y (AOI22XL)                         0.16      14.74 f
```

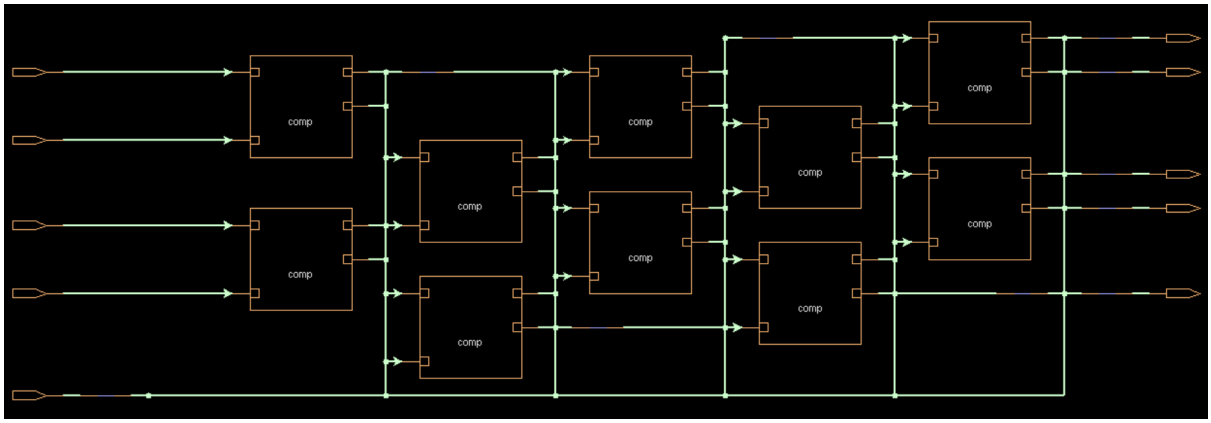
```
  U620/Y (AOI211XL)                        0.22      14.96 r
  U621/Y (AOI31XL)                         0.10      15.06 f
  U622/Y (AOI221XL)                        0.31      15.37 r
  out_data[1] (out)                        0.00      15.37 r
  data arrival time                                  15.37

  max_delay                               30.00      30.00
  output external delay                    0.00      30.00
  data required time                                 30.00
  -----------------------------------------------------------
  data required time                                 30.00
  data arrival time                                 -15.37
  -----------------------------------------------------------
  slack (MET)                                        14.63


1
report_area

****************************************
Report : area
Design : SMJ
Version: R-2020.09
Date   : Thu Mar 16 00:16:00 2023
****************************************

Library(s) Used:

    slow (File: /RAID2/COURSE/iclab/iclabta01/umc018/Synthesis/slow.db)

Number of ports:                          32
Number of nets:                          341
Number of cells:                         311
Number of combinational cells:           311
Number of sequential cells:                0
Number of macros/black boxes:              0
Number of buf/inv:                        73
Number of references:                     32

Combinational area:              5501.865664
Buf/Inv area:                     728.481627
Noncombinational area:              0.000000
Macro/Black Box area:               0.000000
Net Interconnect area:      undefined  (No wire load specified)

Total cell area:                 5501.865664
Total area:                 undefined
1
exit

Memory usage for this session 179 Mbytes.
Memory usage for this session including child processes 179 Mbytes.
CPU usage for this session 6 seconds ( 0.00 hours ).
Elapsed time for this session 10 seconds ( 0.00 hours ).

Thank you...
```

設計過程

最初版本(area=6153.840063)

一開始使用的是助教提供的sort架構，這個電路本身就有約4400的面積。這個階段只求驗證電路的邏輯，因此其他部分的判斷電路並沒有重複使用判斷結果，所以面積並不是很理想。

下圖為這個版本中使用的 sort 簡圖。



▼ code

```
// area 6153.840063
module comp(
    input [5:0] ai,bi,
    output logic [5:0] ao,bo
);

    always_comb begin : comp_module
        ao = (ai < bi) ? ai : bi;
        bo = (ai < bi) ? bi : ai;
    end

endmodule

module Sort (
    input [5:0] in_num0, in_num1, in_num2, in_num3, in_num4,
    output [5:0] out_num0, out_num1, out_num2, out_num3, out_num4
);
    //inter_wire[layer][row]
    logic [5:0] inter_wire [4:0][4:0];

    // 1st layer
    comp c1(in_num0, in_num1, inter_wire[0][0],inter_wire[0][1]);
    comp c2(in_num2, in_num3, inter_wire[0][2],inter_wire[0][3]);
    assign inter_wire[0][4] = in_num4;

    // 2nd layer
    assign inter_wire[1][0] = inter_wire[0][0];
    comp c3(inter_wire[0][1], inter_wire[0][2], inter_wire[1][1],inter_wire[1][2]);
    comp c4(inter_wire[0][3], inter_wire[0][4], inter_wire[1][3], inter_wire[1][4]);

    // 3nd layer
    assign inter_wire[2][4] = inter_wire[1][4];
    comp c5(inter_wire[1][0], inter_wire[1][1], inter_wire[2][0],inter_wire[2][1]);
    comp c6(inter_wire[1][2], inter_wire[1][3], inter_wire[2][2], inter_wire[2][3]);

    // 4nd layer
    assign inter_wire[3][0] = inter_wire[2][0];
    comp c7(inter_wire[2][1], inter_wire[2][2], inter_wire[3][1],inter_wire[3][2]);
    comp c8(inter_wire[2][3], inter_wire[2][4], inter_wire[3][3], inter_wire[3][4]);

    // 5th layer
    assign inter_wire[4][4] = inter_wire[3][4];
    comp c9(inter_wire[3][0], inter_wire[3][1], inter_wire[4][0],inter_wire[4][1]);
    comp c10(inter_wire[3][2], inter_wire[3][3], inter_wire[4][2], inter_wire[4][3]);

    // output
    assign out_num0 = inter_wire[4][0];
    assign out_num1 = inter_wire[4][1];
    assign out_num2 = inter_wire[4][2];
    assign out_num3 = inter_wire[4][3];
    assign out_num4 = inter_wire[4][4];

endmodule

module SMJ(
    // Input signals
    hand_n0,
    hand_n1,
    hand_n2,
    hand_n3,
    hand_n4,
    // Output signals
```

```
    out_data
);
//---------------------------------------------------------------------
//   INPUT AND OUTPUT DECLARATION
//---------------------------------------------------------------------
input [5:0] hand_n0;
input [5:0] hand_n1;
input [5:0] hand_n2;
input [5:0] hand_n3;
input [5:0] hand_n4;
output logic [1:0] out_data;

//---------------------------------------------------------------------
//   LOGIC DECLARATION
//---------------------------------------------------------------------
logic [5:0] sorted_hand[4:0];

logic invalid_check[4:0];
logic invalid_term_exist;
logic tri_plus_pair;
logic seq_plus_pair;

//---------------------------------------------------------------------
//   Your design
//---------------------------------------------------------------------

// Sort all input
Sort sort_ins(.in_num0 (hand_n0), .in_num1 (hand_n1), .in_num2 (hand_n2), .in_num3 (hand_n3),.in_num4 (hand_n4),
               .out_num0 (sorted_hand[0]), .out_num1 (sorted_hand[1]), .out_num2 (sorted_hand[2]), .out_num3 (sorted_hand[3]), .o

always_comb begin

    // Check if input contains invalid terms
    for(int i=0;i<5;i++) begin
        invalid_check[i] = ((sorted_hand[i][5:4]==2'b00) ? sorted_hand[i][3:0]>6 : sorted_hand[i][3:0]>8);
    end

    invalid_term_exist = invalid_check[0] || invalid_check[1] || invalid_check[2] || invalid_check[3] || invalid_check[4] || (sort

    // Check tri_plus_pair case
    tri_plus_pair = ((sorted_hand[0]==sorted_hand[1]) && (sorted_hand[2]==sorted_hand[4])) || ((sorted_hand[3]==sorted_hand[4]) &&

    //Check seq_plus_pair case
    seq_plus_pair = (sorted_hand[1]-sorted_hand[0]==1 && sorted_hand[2]-sorted_hand[1]==1 && sorted_hand[3]==sorted_hand[4] && sor
                || (sorted_hand[0]==sorted_hand[1] && sorted_hand[3]-sorted_hand[2]==1 && sorted_hand[4]-sorted_hand[3]==1 && sort
                || (sorted_hand[1]==sorted_hand[3] && sorted_hand[1]-sorted_hand[0]==1 && sorted_hand[4]-sorted_hand[3]==1 && sort

    if(invalid_term_exist)begin
        out_data = 2'b01;
    end
    else if(tri_plus_pair)begin
        out_data = 2'b11;
    end
    else if(seq_plus_pair)begin
        out_data = 2'b10;
    end
    else begin
        out_data = 2'b00;
    end
end

endmodule
```

重複使用字牌判斷結果、改變sequence判斷方式(area=6173.798460)

確定電路的模擬結果正確、合成正確後，我先試著重複使用字牌的判斷結果。另外，我有聽說有同學將 sequence 的判斷方式從減法改為用加法判斷後面積稍微降低，這個結果似乎有其道理，因為EDA工具中減法器的實現方式可能是在加法器的輸入端加上XOR閘，藉此控制是否將加/減數轉為2's complement ，因此我也嘗試看看。但結果面積沒有改變多少，顯示問題可能不是出在這裡。而是排序和其他的判斷電路。

▼ code

```
//area  6173.798460
module comp(
    input [5:0] ai,bi,
    output logic [5:0] ao,bo
);

    always_comb begin : comp_module
        ao = (ai < bi) ? ai : bi;
        bo = (ai < bi) ? bi : ai;
```

```
    end

endmodule

module Sort (
    input [5:0] in_num0, in_num1, in_num2, in_num3, in_num4,
    output [5:0] out_num0, out_num1, out_num2, out_num3, out_num4
);
    //inter_wire[layer][row]
    logic [5:0] inter_wire [4:0][4:0];

    // 1st layer
    comp c1(in_num0, in_num1, inter_wire[0][0],inter_wire[0][1]);
    comp c2(in_num2, in_num3, inter_wire[0][2],inter_wire[0][3]);
    assign inter_wire[0][4] = in_num4;

    // 2nd layer
    assign inter_wire[1][0] = inter_wire[0][0];
    comp c3(inter_wire[0][1], inter_wire[0][2], inter_wire[1][1],inter_wire[1][2]);
    comp c4(inter_wire[0][3], inter_wire[0][4], inter_wire[1][3], inter_wire[1][4]);

    // 3nd layer
    assign inter_wire[2][4] = inter_wire[1][4];
    comp c5(inter_wire[1][0], inter_wire[1][1], inter_wire[2][0],inter_wire[2][1]);
    comp c6(inter_wire[1][2], inter_wire[1][3], inter_wire[2][2], inter_wire[2][3]);

    // 4nd layer
    assign inter_wire[3][0] = inter_wire[2][0];
    comp c7(inter_wire[2][1], inter_wire[2][2], inter_wire[3][1],inter_wire[3][2]);
    comp c8(inter_wire[2][3], inter_wire[2][4], inter_wire[3][3], inter_wire[3][4]);

    // 5th layer
    assign inter_wire[4][4] = inter_wire[3][4];
    comp c9(inter_wire[3][0], inter_wire[3][1], inter_wire[4][0],inter_wire[4][1]);
    comp c10(inter_wire[3][2], inter_wire[3][3], inter_wire[4][2], inter_wire[4][3]);

    // output
    assign out_num0 = inter_wire[4][0];
    assign out_num1 = inter_wire[4][1];
    assign out_num2 = inter_wire[4][2];
    assign out_num3 = inter_wire[4][3];
    assign out_num4 = inter_wire[4][4];

endmodule

module SMJ(
    // Input signals
    hand_n0,
    hand_n1,
    hand_n2,
    hand_n3,
    hand_n4,
    // Output signals
    out_data
);
//---------------------------------------------------------------------
//   INPUT AND OUTPUT DECLARATION
//---------------------------------------------------------------------
input [5:0] hand_n0;
input [5:0] hand_n1;
input [5:0] hand_n2;
input [5:0] hand_n3;
input [5:0] hand_n4;
output logic [1:0] out_data;

//---------------------------------------------------------------------
//   LOGIC DECLARATION
//---------------------------------------------------------------------
logic [5:0] sorted_hand[4:0];

logic invalid_check[4:0];
logic invalid_term_exist;
logic tri_plus_pair;
logic seq_plus_pair;
logic is_honor [4:0];


//---------------------------------------------------------------------
//   Your design
//---------------------------------------------------------------------

// Sort all input
Sort sort_ins(.in_num0 (hand_n0), .in_num1 (hand_n1), .in_num2 (hand_n2), .in_num3 (hand_n3),.in_num4 (hand_n4),
                .out_num0 (sorted_hand[0]), .out_num1 (sorted_hand[1]), .out_num2 (sorted_hand[2]), .out_num3 (sorted_hand[3]), .o

always_comb begin
```

```
    // Check if input contains invalid terms
    for(int i=0;i<5;i++) begin
        is_honor[i] = (sorted_hand[i][5:4]==2'b00);
        invalid_check[i] = (is_honor[i] ? sorted_hand[i][3:0]>6 : sorted_hand[i][3:0]>8);
    end

    invalid_term_exist = invalid_check[0] || invalid_check[1] || invalid_check[2] || invalid_check[3] || invalid_check[4] || (sort

    // Check tri_plus_pair case
    tri_plus_pair = ((sorted_hand[0]==sorted_hand[1]) && (sorted_hand[2]==sorted_hand[4])) || ((sorted_hand[3]==sorted_hand[4]) &&

    //Check seq_plus_pair case
    seq_plus_pair = (sorted_hand[1]==sorted_hand[0]+1 && sorted_hand[2]==sorted_hand[1]+1 && sorted_hand[3]==sorted_hand[4] && !is
                || (sorted_hand[0]==sorted_hand[1] && sorted_hand[3]==sorted_hand[2]+1 && sorted_hand[4]==sorted_hand[3]+1 && !is_
                || (sorted_hand[1]==sorted_hand[3] && sorted_hand[1]==sorted_hand[0]+1 && sorted_hand[4]==sorted_hand[3]+1 && !is_

    if(invalid_term_exist)begin
        out_data = 2'b01;
    end
    else if(tri_plus_pair)begin
        out_data = 2'b11;
    end
    else if(seq_plus_pair)begin
        out_data = 2'b10;
    end
    else begin
        out_data = 2'b00;
    end
end

endmodule
```

重複使用sequence、triplet判斷結果(area=6070.680064)

我把 sequence、triplet 判斷電路中的==、>判斷結果全部都先存到logic值中，方便之後重複使用，減少比較器的數量，果然面積下降了約100。

▼ code

```
// area 6070.680064
module comp(
    input [5:0] ai,bi,
    output logic [5:0] ao,bo
);

    always_comb begin : comp_module
        ao = (ai < bi) ? ai : bi;
        bo = (ai < bi) ? bi : ai;
    end

endmodule

module Sort (
    input [5:0] in_num0, in_num1, in_num2, in_num3, in_num4,
    output [5:0] out_num0, out_num1, out_num2, out_num3, out_num4
);
    //inter_wire[layer][row]
    logic [5:0] inter_wire [4:0][4:0];

    // 1st layer
    comp c1(in_num0, in_num1, inter_wire[0][0],inter_wire[0][1]);
    comp c2(in_num2, in_num3, inter_wire[0][2],inter_wire[0][3]);
    assign inter_wire[0][4] = in_num4;

    // 2nd layer
    assign inter_wire[1][0] = inter_wire[0][0];
    comp c3(inter_wire[0][1], inter_wire[0][2], inter_wire[1][1],inter_wire[1][2]);
    comp c4(inter_wire[0][3], inter_wire[0][4], inter_wire[1][3], inter_wire[1][4]);

    // 3nd layer
    assign inter_wire[2][4] = inter_wire[1][4];
    comp c5(inter_wire[1][0], inter_wire[1][1], inter_wire[2][0],inter_wire[2][1]);
    comp c6(inter_wire[1][2], inter_wire[1][3], inter_wire[2][2], inter_wire[2][3]);

    // 4nd layer
    assign inter_wire[3][0] = inter_wire[2][0];
    comp c7(inter_wire[2][1], inter_wire[2][2], inter_wire[3][1],inter_wire[3][2]);
    comp c8(inter_wire[2][3], inter_wire[2][4], inter_wire[3][3], inter_wire[3][4]);

    // 5th layer
    assign inter_wire[4][4] = inter_wire[3][4];
    comp c9(inter_wire[3][0], inter_wire[3][1], inter_wire[4][0],inter_wire[4][1]);
```

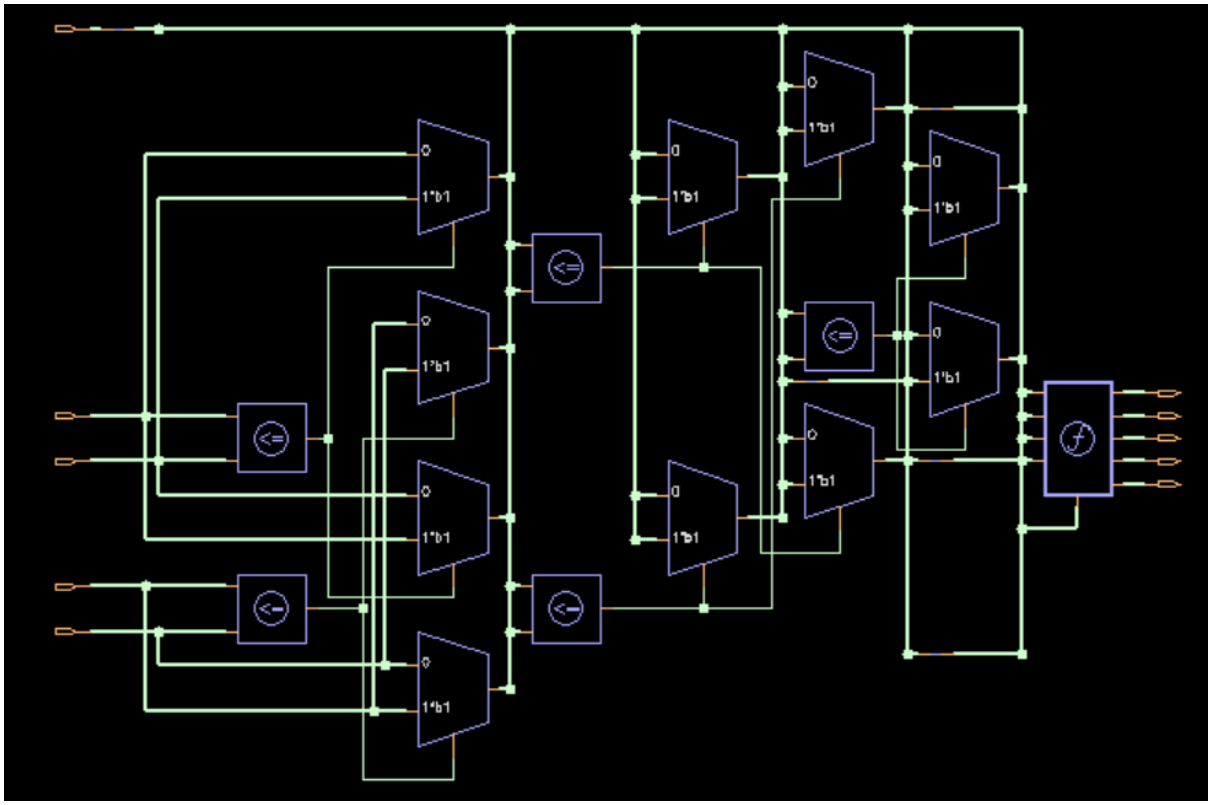
```
    comp c10(inter_wire[3][2], inter_wire[3][3], inter_wire[4][2], inter_wire[4][3]);

    // output
    assign out_num0 = inter_wire[4][0];
    assign out_num1 = inter_wire[4][1];
    assign out_num2 = inter_wire[4][2];
    assign out_num3 = inter_wire[4][3];
    assign out_num4 = inter_wire[4][4];

endmodule

module SMJ(
    // Input signals
    hand_n0,
    hand_n1,
    hand_n2,
    hand_n3,
    hand_n4,
    // Output signals
    out_data
);
//---------------------------------------------------------------------
//   INPUT AND OUTPUT DECLARATION
//---------------------------------------------------------------------
input [5:0] hand_n0;
input [5:0] hand_n1;
input [5:0] hand_n2;
input [5:0] hand_n3;
input [5:0] hand_n4;
output logic [1:0] out_data;

//---------------------------------------------------------------------
//   LOGIC DECLARATION
//---------------------------------------------------------------------
logic [5:0] sorted_hand[4:0];

logic invalid_check[4:0];
logic invalid_term_exist;
logic tri_plus_pair;
logic seq_plus_pair;
logic is_honor [4:0];

logic equal_01, equal_24, equal_34, equal_02, equal_13, diff_01, diff_12, diff_23, diff_34;


logic pair_01, pair_34;
logic seq_012, seq_234, seq_01_34;
logic tri_012, tri_123, tri_234;

//---------------------------------------------------------------------
//   Your design
//---------------------------------------------------------------------

// Sort all input

Sort sort_ins(.in_num0 (hand_n0), .in_num1 (hand_n1), .in_num2 (hand_n2), .in_num3 (hand_n3),.in_num4 (hand_n4),
                .out_num0 (sorted_hand[0]), .out_num1 (sorted_hand[1]), .out_num2 (sorted_hand[2]), .out_num3 (sorted_hand[3]), .o

always_comb begin
    // these indicate a pair
    equal_01 = sorted_hand[0]==sorted_hand[1];
    equal_34 = sorted_hand[3]==sorted_hand[4];

    // these indicate a triplet
    equal_02 = sorted_hand[0]==sorted_hand[2];
    equal_13 = sorted_hand[1]==sorted_hand[3];
    equal_24 = sorted_hand[2]==sorted_hand[4];

    // 01_12(!is_honor[2]), 12_23(!is_honor[2]), 23_34(!is_honor[2]) indicates a seq
    diff_01 = sorted_hand[1]==sorted_hand[0]+1;
    diff_12 = sorted_hand[2]==sorted_hand[1]+1;
    diff_23 = sorted_hand[3]==sorted_hand[2]+1;
    diff_34 = sorted_hand[4]==sorted_hand[3]+1;

    // Check if input contains invalid terms
    for(int i=0;i<5;i++) begin
        is_honor[i] = (sorted_hand[i][5:4]==2'b00/*&2'b11*/);
        invalid_check[i] = (is_honor[i] ? sorted_hand[i][3:0]>6 : sorted_hand[i][3:0]>8);
    end

    invalid_term_exist = invalid_check[0] || invalid_check[1] || invalid_check[2] || invalid_check[3] || invalid_check[4] || (equa


    pair_01 = equal_01;
    pair_34 = equal_34;

    seq_012 = diff_01 & diff_12;
```

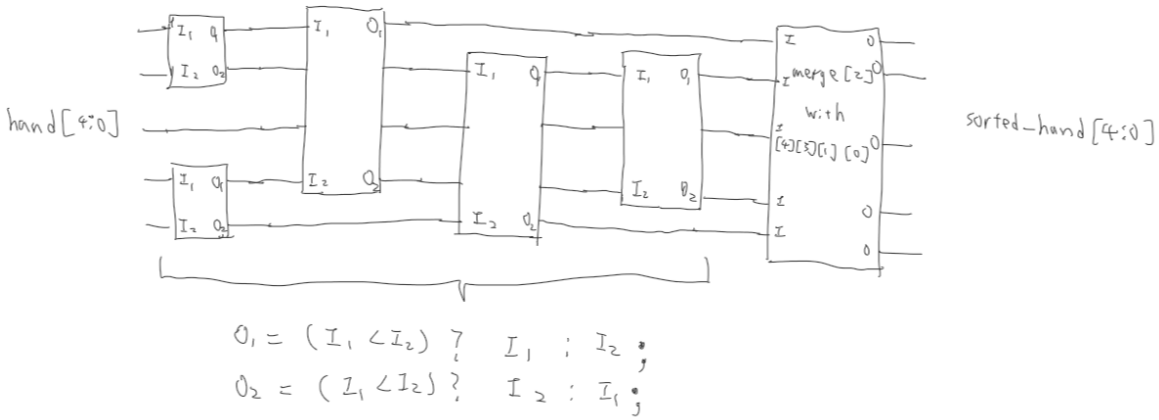
```
    seq_234 = diff_23 & diff_34;
    seq_01_34 = diff_01 & diff_34 & equal_13;

    tri_012 = equal_02;
    tri_123 = equal_13;
    tri_234 = equal_24;

    if(invalid_term_exist)begin
        out_data = 2'b01;
    end
    else if((tri_012 && pair_34)||(tri_234 && pair_01))begin
        out_data = 2'b11;
    end
    else if(is_honor[2])begin
        out_data = 2'b00;
    end
    else if((pair_01 && seq_234)||(seq_012 && pair_34)||(seq_01_34 &&tri_123))begin
        out_data = 2'b10;
    end
    else begin
        out_data = 2'b00;
    end


end

endmodule
```

改變sort方式(area=5501.865664)

我後來想起 sort 模組中的小 module — comp 因為 module 之間的優化是獨立的，並不會被EDA工具給化簡。而且還想到了另一個比原本的排序演算法少用一個比較器的排序方式。將其套用後，面積下降到5501。

下圖為此 sort 的簡圖，首先將0、1、3、4排序，再將 2 混合，形成最終的排序後輸出。





▼ code

```
// area 5501.865664
module Sort (
    input logic [5:0] in_num0, in_num1, in_num2, in_num3, in_num4,
    output logic  [5:0] out_num0, out_num1, out_num2, out_num3, out_num4
);

logic [5:0] inter_wire [2:0][4:0];
logic comp_01, comp_34, comp_03, comp_14,comp_13;

//1st layer
assign comp_01 = in_num0 <= in_num1;
assign comp_34 = in_num3 <= in_num4;

assign inter_wire[0][0] = (comp_01) ? in_num0 : in_num1;
assign inter_wire[0][1] = (comp_01) ? in_num1 : in_num0;
assign inter_wire[0][2] = in_num2;
assign inter_wire[0][3] = (comp_34) ? in_num3 : in_num4;
assign inter_wire[0][4] = (comp_34) ? in_num4 : in_num3;

//2nd layer
assign comp_03 = inter_wire[0][0] <= inter_wire[0][3];
assign comp_14 = inter_wire[0][1] <= inter_wire[0][4];

assign inter_wire[1][0] = (comp_03) ? inter_wire[0][0] : inter_wire[0][3];
assign inter_wire[1][1] = (comp_14) ? inter_wire[0][1] : inter_wire[0][4];
assign inter_wire[1][2] = inter_wire[0][2];
assign inter_wire[1][3] = (comp_03) ? inter_wire[0][3] : inter_wire[0][0];
assign inter_wire[1][4] = (comp_14) ? inter_wire[0][4] : inter_wire[0][1];

//3rd layer
assign comp_13 = inter_wire[1][1] <= inter_wire[1][3];

assign inter_wire[2][0] = inter_wire[1][0];
assign inter_wire[2][1] = (comp_13) ? inter_wire[1][1] : inter_wire[1][3];
assign inter_wire[2][2] = inter_wire[1][2];
assign inter_wire[2][3] = (comp_13) ? inter_wire[1][3] : inter_wire[1][1];
assign inter_wire[2][4] = inter_wire[1][4];

always_comb begin
    // output selection
    if(inter_wire[2][2] > inter_wire[2][4])begin
        out_num0 = inter_wire[2][0];
        out_num1 = inter_wire[2][1];
        out_num2 = inter_wire[2][3];
        out_num3 = inter_wire[2][4];
        out_num4 = inter_wire[2][2];
    end
    else if(inter_wire[2][2] > inter_wire[2][3])begin
        out_num0 = inter_wire[2][0];
        out_num1 = inter_wire[2][1];
        out_num2 = inter_wire[2][3];
        out_num3 = inter_wire[2][2];
        out_num4 = inter_wire[2][4];
    end
    else if(inter_wire[2][2] > inter_wire[2][1])begin
        out_num0 = inter_wire[2][0];
        out_num1 = inter_wire[2][1];
        out_num2 = inter_wire[2][2];
        out_num3 = inter_wire[2][3];
        out_num4 = inter_wire[2][4];
```

```
    end
    else if(inter_wire[2][2] > inter_wire[2][0])begin
        out_num0 = inter_wire[2][0];
        out_num1 = inter_wire[2][2];
        out_num2 = inter_wire[2][1];
        out_num3 = inter_wire[2][3];
        out_num4 = inter_wire[2][4];
    end
    else begin
        out_num0 = inter_wire[2][2];
        out_num1 = inter_wire[2][0];
        out_num2 = inter_wire[2][1];
        out_num3 = inter_wire[2][3];
        out_num4 = inter_wire[2][4];
    end
end

endmodule

module SMJ(
    // Input signals
    hand_n0,
    hand_n1,
    hand_n2,
    hand_n3,
    hand_n4,
    // Output signals
    out_data
);
//---------------------------------------------------------------------
//   INPUT AND OUTPUT DECLARATION
//---------------------------------------------------------------------
input [5:0] hand_n0;
input [5:0] hand_n1;
input [5:0] hand_n2;
input [5:0] hand_n3;
input [5:0] hand_n4;
output logic [1:0] out_data;

//---------------------------------------------------------------------
//   LOGIC DECLARATION
//---------------------------------------------------------------------
logic [5:0] sorted_hand[4:0];

logic invalid_check[4:0];
logic invalid_term_exist;
logic tri_plus_pair;
logic seq_plus_pair;
logic is_honor [4:0];

logic equal_01, equal_24, equal_34, equal_02, equal_13, diff_01, diff_12, diff_23, diff_34;


logic pair_01, pair_34;
logic seq_012, seq_234, seq_01_34;
logic tri_012, tri_123, tri_234;

//---------------------------------------------------------------------
//   Your design
//---------------------------------------------------------------------

// Sort all input

Sort sort_ins(.in_num0 (hand_n0), .in_num1 (hand_n1), .in_num2 (hand_n2), .in_num3 (hand_n3),.in_num4 (hand_n4),
                .out_num0 (sorted_hand[0]), .out_num1 (sorted_hand[1]), .out_num2 (sorted_hand[2]), .out_num3 (sorted_hand[3]), .o

always_comb begin
    // these indicate a pair
    equal_01 = sorted_hand[0]==sorted_hand[1];
    equal_34 = sorted_hand[3]==sorted_hand[4];

    // these indicate a triplet
    equal_02 = sorted_hand[0]==sorted_hand[2];
    equal_13 = sorted_hand[1]==sorted_hand[3];
    equal_24 = sorted_hand[2]==sorted_hand[4];

    // 01_12(!is_honor[2]), 12_23(!is_honor[2]), 23_34(!is_honor[2]) indicates a seq
    diff_01 = sorted_hand[1]==sorted_hand[0]+1;
    diff_12 = sorted_hand[2]==sorted_hand[1]+1;
    diff_23 = sorted_hand[3]==sorted_hand[2]+1;
    diff_34 = sorted_hand[4]==sorted_hand[3]+1;

    // Check if input contains invalid terms
    for(int i=0;i<5;i++) begin
        is_honor[i] = (sorted_hand[i][5:4]==2'b00);
        invalid_check[i] = (is_honor[i] ? sorted_hand[i][3:0]>6 : sorted_hand[i][3:0]>8);
    end
```

```
    invalid_term_exist = invalid_check[0] || invalid_check[1] || invalid_check[2] || invalid_check[3] || invalid_check[4] || (equa

    pair_01 = equal_01;
    pair_34 = equal_34;

    seq_012 = diff_01 & diff_12;
    seq_234 = diff_23 & diff_34;
    seq_01_34 = diff_01 & diff_34 & equal_13;

    tri_012 = equal_02;
    tri_123 = equal_13;
    tri_234 = equal_24;

    if(invalid_term_exist)begin
        out_data = 2'b01;
    end
    else if((tri_012 && pair_34)||(tri_234 && pair_01))begin
        out_data = 2'b11;
    end
    else if(is_honor[2])begin
        out_data = 2'b00;
    end
    else if((pair_01 && seq_234)||(seq_012 && pair_34)||(seq_01_34 &&tri_123))begin
        out_data = 2'b10;
    end
    else begin
        out_data = 2'b00;
    end

end

endmodule
```