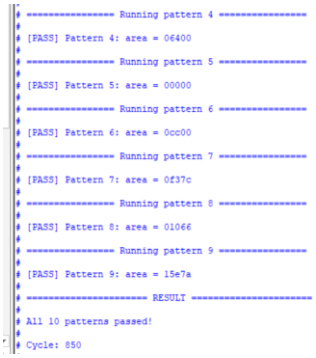
2025 Digital IC Design Homework 5

NAME	傅信豪
Student ID	NE6121084

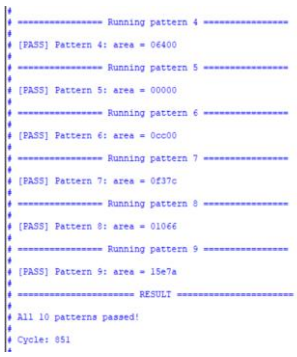
Simulation Result

Functional simulation	Pass	Pre-Layout simulation	Pass/Fail

(your functional sim result).



(your pre-sim result)



Synthesis Result

Total logic elements	3822
Total memory bits	0
Total registers	395
Embedded multiplier 9-bit elements	24
Clock period (ns)	20
Total Cycle used	851

(your flow summary of MCH)

Flow Summary

<<Filter>>

Flow Status	Successful - Fri Jun 6 18:09:12 2025
Quartus Prime Version	22.1std.0 Build 915 10/25/2022 SC Lite Edition
Revision Name	MCH
Top-level Entity Name	MCH
Family	Cyclone IV E
Device	EP4CE55F23A7
Timing Models	Final
Total logic elements	3,822 / 55,856 (7 %)
Total registers	395
Total pins	36 / 325 (11 %)
Total virtual pins	0
Total memory bits	0 / 2,396,160 (0 %)
Embedded Multiplier 9-bit elements	24 / 308 (8 %)
Total PLLs	0 / 4 (0 %)

Description of your design

狀態（State）定義

- IDLE：系統初始／等待狀態，程式進入後會立即跳到 READ
- READ：讀取階段
- SWAP：交換階段
- SORT：排序階段
- SCAN：Graham Scan 階段
- AREA：面積累加階段
- DONE：完成輸出階段

READ

- 持續讀入共 20 個 (X, Y) 座標到 x_coord[0..19] 與 y_coord[0..19]
- 同時比較當前輸入的點是否比先前記錄的 pivot（最低 y、最左 x）更「下或更左」，並更新 idx_min
- 讀滿 20 筆後，即觸發 SWAP 狀態

SWAP

- 將之前找到的 pivot（idx_min）與索引 0 的座標互換──也就是把「最下且最左」的點移到 x_coord[0], y_coord[0]
- 後續排序時，x_coord[0] 一定是那個 anchor 點

SORT

- **目的**：將剩餘 19 個點依據對 anchor 的極角（polar angle）做排序，使掃描時從最小極角開始。
- **實作方法**：採用 **odd-even bubble sort**（奇偶氣泡排序）
  - 共執行固定 20 次迴圈（sort_cycles == 20 時結束）
  - 每個時鐘週期內，分兩個 Phase：
    - **偶 phase（sort_idx == 0）**：同時比較 (1 vs.2)、(3 vs.4)、…、(17 vs.18) 共 9 對，若需要就交換座標。
    - **奇 phase（sort_idx == 1）**：同時比較 (2 vs.3)、(4 vs.5)、…、(18 vs.19) 共 9 對，若需要就交換座標。
  - 每次比較 **以 two-point cross product** (cross_res(dx[i], dy[i], dx[i+1], dy[i+1])) 決定相對順序：
    - 若外積 <0（順時針或共線且 dx 大小不符）則要交換
    - 否則保持原順序
- **硬體優化重點**：

- *時間複雜度*：即使是 bubble sort，固定跑 20 個 cycle 就能保證 19 個點排序完成 → 總 cycle = O(n)
- *組合邏輯*：在每個 cycle 內需要同時比較 9 對點對，因此至少需要 O(n/2) 個 cross-product 算元（也就是 O(n/2) 個 9×9 乘法器＋減法器＋比較邏輯）
- *交替 Phase*：用一個 sort_idx 訊號切換「偶 phase」「奇 phase」，保持 pipeline 流暢
- *為何選 20 cycle*：理論上 odd-even bubble sort 在 n=19 點時，只要跑足 n 個 iteration（這裡寫成 20），就能保證排序正確

SCAN（Graham Scan）

- **目的**：以已排序的點為基礎，逐一決定哪些點應該在 convex hull 中（push）、哪些要彈出（pop）
- **主要暫存**：
  - stack_ptr：指向下一個可 push 的位置（等於堆疊大小）
  - cur_index：當前要處理的 sorted 座標索引（初始設為 2，因為 x_coord[0] 與 x_coord[1] 已是 hull 的兩個起始點）
- **外積判斷**：
  - 取堆疊頂部的兩點 (x1, y1)、(x2, y2) 以及當前點 (cx, cy)
  - 計算 crs = (x2 - x1)*(cy - y2) - (y2 - y1)*(cx - x2)
  - 若 crs > 0 → 左轉，代表 stack 上兩點與當前點所形成的角度是逆時針 → **push** 當前點到 stack，stack_ptr+1; 並 cur_index+1
  - 否則 (crs ≤ 0) → 代表右轉或共線 → **pop** stack_ptr-1（只要 stack_ptr > 1），再繼續與新頂部重算
  - 若 stack_ptr == 1 時（堆疊只剩一點），直接 **push**；stack_ptr+1, 並 cur_index+1
- **重點維護**：
  - stack_ptr 紀錄目前 stack 的大小／下個推入位置
  - cur_index 逐步從 2 遞增到 19，直到所有點都處理完成
  - 由於每次 pop 後不改 cur_index，需要在下一個時鐘再重新算一次新的頂部組合
  - 最終在 cur_index == 20 時，全部掃描結束，stack 中即為 convex hull 的頂點序列

AREA（面積計算）

- 使用 Shoelace 公式，對 hull 上的所有邊做外積累加以取得二倍面積值
- 先將 stack 裡的頂點依序取出兩兩計算，最後還要加上從最後一個頂點回到第一個頂點的外積
- 每次只需單一外積計算單元，並搭配一個累加暫存器逐次累加，以節省硬體資源
- 最終把累積結果（即二倍面積）輸出作為模組輸出，以供驗證程式進一步計算或比較

DONE（結束訊號）

- 當面積計算完成後，將 Done 訊號拉高，表示本批二十筆座標的處理已結束
- 同時清空所有讀入的座標、計數器、堆疊指標與累加器，等待下一輪輸入再次進入 READ 階段