2025 Digital IC Design Homework 2

NAME	傅信豪
Student ID	NE6121084

Functional Simulation Result

Pattern1	Pattern2	Pattern3	Pattern4	Pattern5
Pass	Pass	Pass	Pass	Pass

Pattern 1

(your simulation result)

```
VSIM 56> run -all
# All data have been generated successfully!
#
#                    ////////////////////////
#                    /                      /       |__||
#                    /  Congratulations !!  /      / 0.0  |
#                    /                      /    /_____   |
#                    /  Simulation PASS !!  /   /^ ^ ^ \  |
#                    /                      /  |^ ^ ^ ^ |w|
#                    ////////////////////////   \m___m__|_|
#
#
# ** Note: $finish    : D:/CSIE_COURSE/ICDESIGN/2025HW/HW2/StudentID/file/testfixture.sv(162)
#    Time: 1355 ns  Iteration: 0  Instance: /testfixture
```

Pattern 2

(your simulation result)

```
VSIM 54> run -all
# All data have been generated successfully!
#
#                    ////////////////////////
#                    /                      /       |__||
#                    /  Congratulations !!  /      / 0.0  |
#                    /                      /    /_____   |
#                    /  Simulation PASS !!  /   /^ ^ ^ \  |
#                    /                      /  |^ ^ ^ ^ |w|
#                    ////////////////////////   \m___m__|_|
#
#
# ** Note: $finish    : D:/CSIE_COURSE/ICDESIGN/2025HW/HW2/StudentID/file/testfixture.sv(162)
#    Time: 1405 ns  Iteration: 0  Instance: /testfixture
```

Pattern 3

(your simulaton result)

```
VSIM 58> run -all
# All data have been generated successfully!
#
#                    ////////////////////////
#                    /                      /       |__||
#                    /  Congratulations !!  /      / 0.0  |
#                    /                      /    /_____   |
#                    /  Simulation PASS !!  /   /^ ^ ^ \  |
#                    /                      /  |^ ^ ^ ^ |w|
#                    ////////////////////////   \m___m__|_|
#
#
# ** Note: $finish    : D:/CSIE_COURSE/ICDESIGN/2025HW/HW2/StudentID/file/testfixture.sv(162)
#    Time: 1445 ns  Iteration: 0  Instance: /testfixture
```

Pattern 4

(your simulation result)

```
VSIM 60> run -all
# All data have been generated successfully!
#
#                    //////////////////////////
#                    /                       /       |__||
#                    /  Congratulations !!   /      / 0.0  |
#                    /                       /    /_____   |
#                    /  Simulation PASS !!   /   /^ ^ ^ \  |
#                    /                       /  |^ ^ ^ ^ |w|
#                    //////////////////////////   \m___m__|_|
#
#
# ** Note: $finish    : D:/CSIE_COURSE/ICDESIGN/2025HW/HW2/StudentID/file/testfixture.sv(162)
#    Time: 1485 ns  Iteration: 0  Instance: /testfixture
```

Pattern 5

(your simulation result)

```
VSIM 62> run -all
# All data have been generated successfully!
#
#                    //////////////////////////
#                    /                       /       |__||
#                    /  Congratulations !!   /      / 0.0  |
#                    /                       /    /_____   |
#                    /  Simulation PASS !!   /   /^ ^ ^ \  |
#                    /                       /  |^ ^ ^ ^ |w|
#                    //////////////////////////   \m___m__|_|
#
#
# ** Note: $finish    : D:/CSIE_COURSE/ICDESIGN/2025HW/HW2/StudentID/file/testfixture.sv(162)
#    Time: 1565 ns  Iteration: 0  Instance: /testfixture
```

LCD_CTRL Finite-State Machine Design:

有限狀態機 (FSM) 設計說明

1. **組合電路邏輯部分（決定下一個狀態）**
   - **always @(*) 區塊：**
     根據目前的狀態與輸入（例如 cmd_valid 與 cmd）的情況，利用 case 語句決定下一個狀態。各狀態轉換邏輯如下：
     - **IDLE → READY：**
       重置結束後，從 IDLE 狀態直接跳轉到 READY 狀態。
     - **READY → FETCH：**
       在 READY 狀態時，提前啟動 IROM 讀取訊號 (IROM_rd) 以便 IROM 進入讀取準備狀態，接著轉入 FETCH 狀態。
     - **FETCH 狀態：**
       根據內部計數器 rom_addr_cnt 的值持續讀取資料；當計數器達到特定值（此處判定為 65）時，轉換到 OFFSET_ORGN 狀態。
     - **OFFSET_ORGN 狀態：**
       此狀態主要等待外部輸入的 cmd 指令，具體行為如下：

        - 當接收到移動類指令（例如 Shift Up/Down/Left/Right）時，只會更新操作座標 (op_x, op_y)，狀態仍保持在 OFFSET_ORGN。
        - 當收到處理指令（如最大值、最小值、平均值計算）時，轉換至 PROCESS 狀態以執行運算。
        - 當 cmd 為 0（代表 SAVE 命令）時，狀態轉換為 SAVE_TORAM，開始將快取資料寫回 IRAM。
    - **PROCESS → UPDATE：**
      在 PROCESS 狀態中根據指令執行對應運算（呼叫相應的計算函式），運算完成後進入 UPDATE 狀態。
    - **UPDATE → OFFSET_ORGN：**
      更新操作完成後，返回 OFFSET_ORGN 狀態，等待下一個指令。
    - **SAVE_TORAM 狀態：**
      利用計數器 ram_sw_counter 逐一將快取中的資料寫入 IRAM，直至所有資料都已寫入，再轉換至 DONE 狀態。
    - **DONE 狀態：**
      進入 DONE 狀態後，停止 IRAM 的寫入動作並保持該狀態，同時將 done 訊號拉高。
2. **序向電路邏輯部分（狀態與資料的更新）**
    - **狀態更新：**
      使用正緣觸發的 always@(posedge clk or posedge rst) 區塊，根據組合邏輯決定的 next_state，在每個 clock edge 時更新 current_state。
    - **資料與計數器更新：**
      另一個正緣 always 區塊中，根據目前的狀態來更新內部的各項計數器（如 rom_addr_cnt、cache_write_addr、ram_sw_counter）、操作座標 (op_x, op_y) 以及其他暫存變數（例如 kernel_result）。各狀態下的具體動作如下：
        - **READY：**
          提升 IROM_rd 訊號。
        - **FETCH：**
          將 IROM_A 設定為當前讀取的地址，並將從 IROM 取得的資料存入內部快取 CACHE。
        - **OFFSET_ORGN：**
          停止 IROM 的讀取，並根據指令更新操作座標（用於移動操作）。

- **PROCESS：**
  根據 cmd 指令呼叫對應的計算函式（如 compute_max、compute_min、compute_average），計算出 kernel_result。
- **UPDATE：**
  利用運算結果，更新快取中指定區域（通常為 4x4 區域）的資料。
- **SAVE_TORAM：**
  透過內部計數器依序將快取中的資料寫入 IRAM，並同時控制 IRAM_ceb 與 IRAM_web 訊號。
- **DONE：**
  停止 IRAM 寫入，並恢復 IRAM 為讀取模式。

- **負緣觸發部分：**
  在負緣觸發的 always 區塊中：
  - 在 FETCH 狀態，將 IROM 讀取到的資料寫入快取 CACHE，同時根據狀態累加 rom_addr_cnt。
  - 在 SAVE_TORAM 狀態，利用負緣觸發更新 ram_sw_counter，以確保每個 clock cycle 寫入下一筆資料到 IRAM。