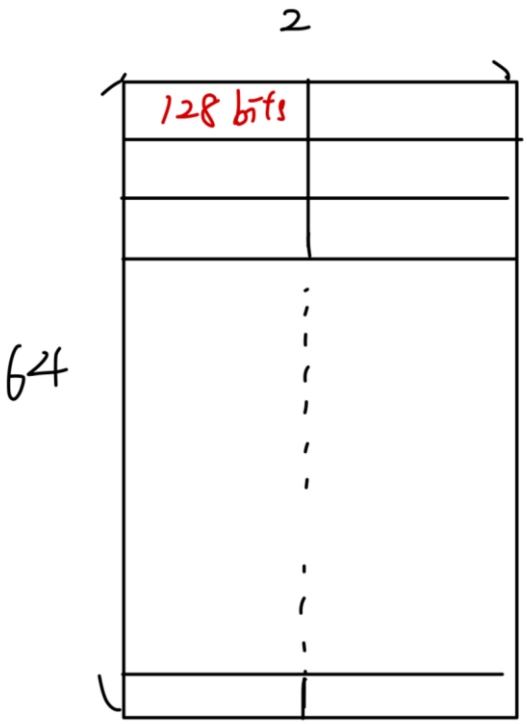
MIdeterm Project Report

學號: 312510144
姓名:張理為

- 電路架構
  - Location map:

    是一個 2 x 64 的 sram，但每個 word 都是 128 個 bits，示意圖如下:

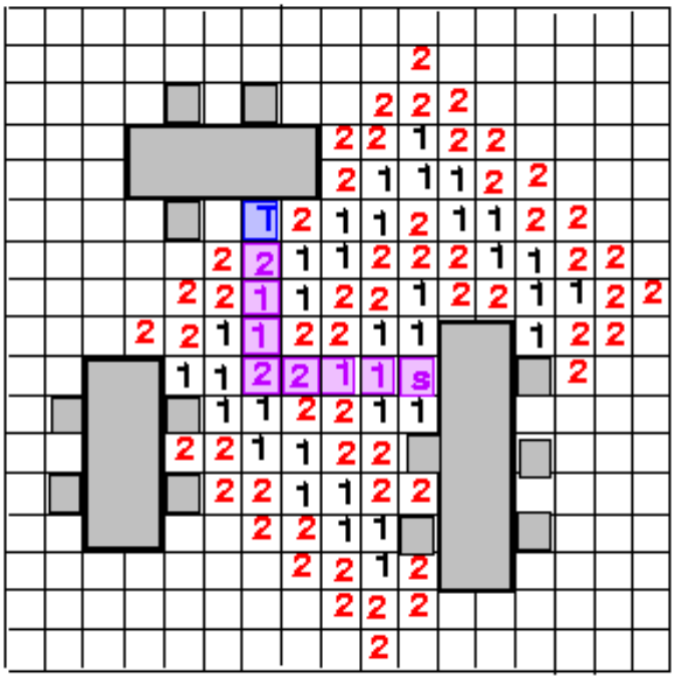


    因為每個數字為 4 個 bits，所以實際上這個 sram 是一個 64 x 64 的正方形，這樣在拿特定資料時，如我要取 map[0][40]的話，可以使用{y[5:0],x[5]}去取資料位置，就會拿到第 0 個 row、第 1 個 column 的 128bits，再使用 x[4:0]去找 128bits 裡的 4 個 bits，也就是 map[0][40]。
  - Weight map

    與 location map 相同的 sram，存取與取值方式皆與 location map 相同。

  - Temp map

    是一個 64 x 64 的 2 bits register，相較於講義提供的 lee's algorithm 的實作方法(1,2,3,1,2,3)，這次我使用的是(1,1,2,2,1,1,2,)的方法，如下圖:



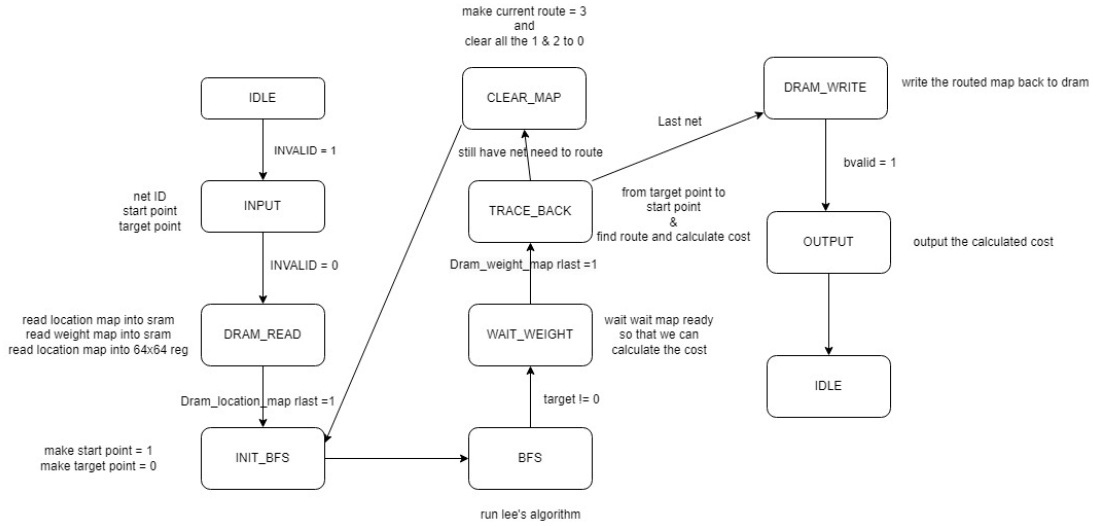
Sequence: 1, 1, 2, 2, 1, 1, 2, 2, ...

要注意，使用這種實作方法的話，必須記錄走到 target 時的步數，這樣才能知道從 target retrace 到 source 時，第一步要找 1 還是 2。

不使用 squence 123 是因為這樣 temp map 必須開到 3 bits，但由於 temp map 是一個超大的 register，在面積限制下開 2 bits 面積才能進入範圍內，而且 2 bits 剛好也能表示圖中的 4 種期況，也就是 0:空、1:標記 1、2:標記 2、3: 障礙物。

也因為我這次 project 使用 register 存 map，在漣漪擴散的過程中非常輕鬆，但是必須注意圖的上下左右邊界與四個角落的擴散條件，以左下角舉例，他只需判斷他的上方與右方即可，若判斷左方或下方會超出 map 的範圍，造成 error。

- FSM



- 優化方法

  除了以上使用 1122 的優化方法外，在這次 project 中我也使用其他的優化，使得面積能夠縮的更小:

  - 使用 input counter 將 dram 的資料存進 register 中會比使用 shift 的方法面積小八萬。
  - 將共用的訊號線拉出，如 assign current_state_IDLE = current_state == IDLE，若有其他 always block 用到 current_state == IDLE 時，就使用 current_state_IDLE 去做判斷。
  - 在 register 擴散時，需判斷上下左右是否為 1 或 2，可以將 1 mapping 成 2'b00、2 mapping 到 2'b01，此時就可以只判斷 MSB 就可以知道該格的鄰居是否需要擴散，而不需要寫 temp_map[x][y] == 1 || temp_map[x][y] == 2，可以省下不少面積。