Project1: Local Median Filter Engine

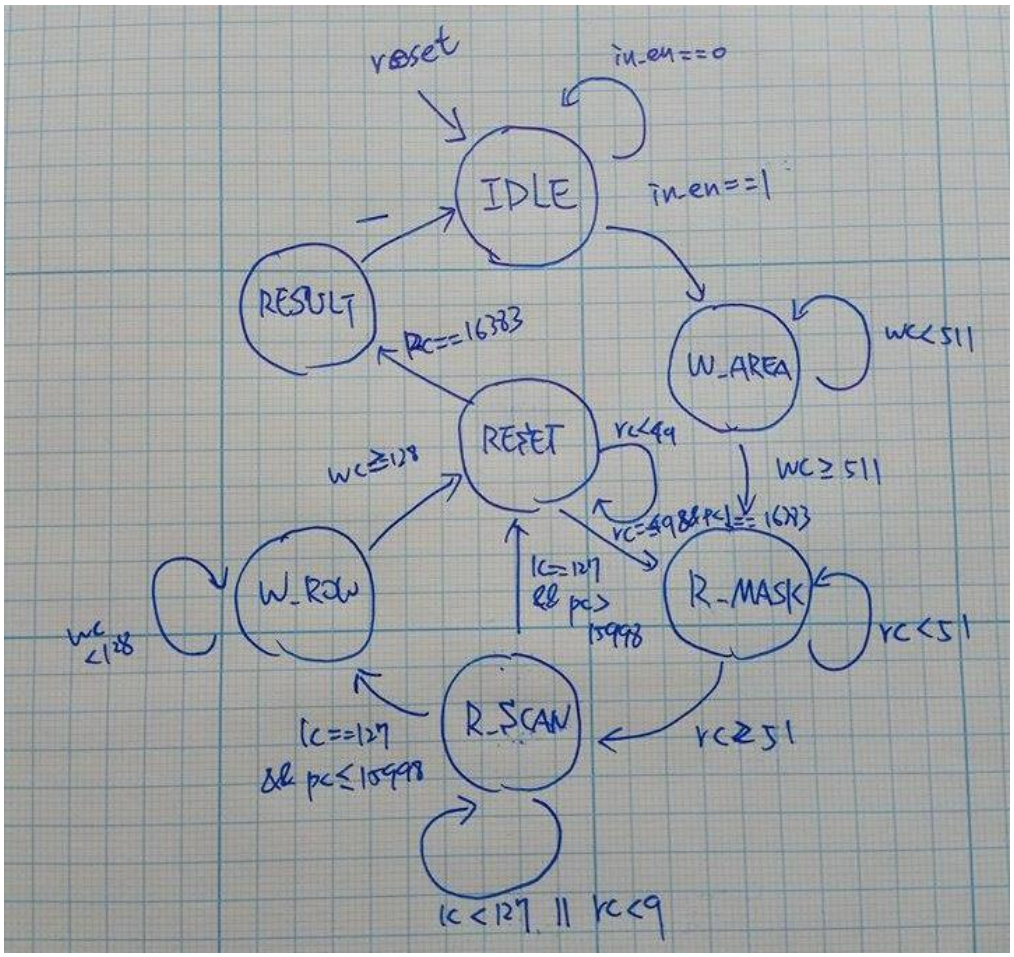
23:05, November 1, 2016

Design Concepts

設計主要概念是用 LMFE 做為頂層包住我所使用的 SRAM、7x7 中位數計算器(lmfe_med49)以及 controller(filter_ctrl)，各 module 詳細說明如下。

Filter Controller-FSM

下圖為 controller 使用的 FSM 概念，各狀態詳述如下：



IDLE: 接收到 in_en 的話便會開始進行 median filter。

W_AREA: 讀取前四 row 的 pixel data 並寫入 SRAM。

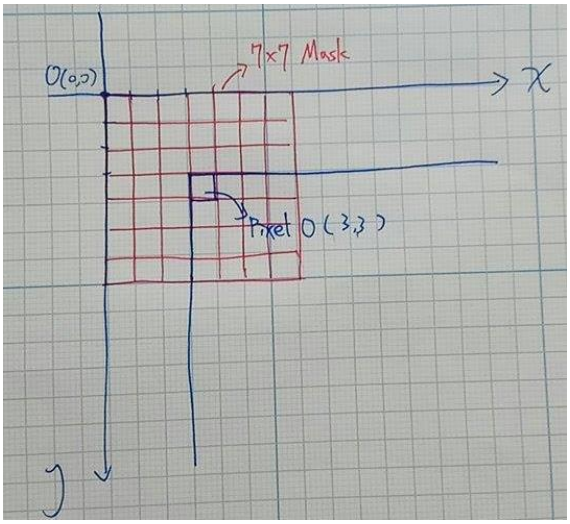
R_MASK: 以 7x7 的 mask 所遮罩的部分依序從 SRAM 讀取並寫送入中位數計算器內，直到送完 49 筆 pixel data 為止(即 Round counter == 51)。

R_SCAN: 從 R_MASK 的點右邊持續輸出 median pixel data(px = px+1)，直到整條 row 掃完為止(即 line counter==127)，並且判斷是否需要寫入下一排的 column，若需要則進入 W_ROW state，不需要則直接進入 RESET state。

W_ROW: 讀取下一 row 的 pixel data 並寫入 SRAM。

RESET: 先判斷當前是否已計算完 16384 個點，若計算完進入 RESULT state，若還沒則重置中位數計算器，並且在重置完後更新 pixel 座標到下一列的第一點(即 px = 3; py = py + 1)然後重回 R_MASK state。

Filter Controller-Trick



在計算 pixel 位置的部分，我把影像的 pixel0 定位在(3,3)，如此一來只要判斷我 mask 涵蓋到的當下 pixel 的 pixel X/Y-coordinate 是否都在 3~130 內即可判別該 pixel 是否屬於影像內的 pixel，是的話則給 Q(即屬於該 pixel 的 pixel data)，超出的部分則代表為 mask 涵蓋超出影像部分，則給 0 值

計算記憶體位址的部分，我使用一個 10bits counter 叫做 ma(即 memory address)，會不斷在每次寫入值後+1，若該 pixel 已經超出 1023(即 10bits 無法表示了)便會 overflow，但因為只有 10bit 所以便會重頭開始計算，此時便會覆蓋住原本已經用不到的 data。

舉例來說 1023 = 10’b11_1111_1111 而 1024 = 11’b100_0000_0000，若此時 ma 從 1023 加到 1024，因為 ma 只有 10bits 的緣故，這時候的 ma 變因為 overflow 變成 10’b00_0000_0000，即記憶體位址 0，因此我只需要 Mask 會遮罩到的 7 row 便可以保證不會有資料還沒用完便被覆蓋住的問題，這也是我只選用 1024bytes SRAM 的緣故。

此外在 decode 出要寫入的記憶體位址時 pixel 的 X/Y-coordinate 都要先減去 3，然後因為 spec 是以 row major 方式給 index，因此我是使用(Y-coordinate-3)*128+X-coordinate 去進行 decode。

Median Number Caculator

會輸入兩個 input，分別為 INS(要 insert 的 number)和 DEL(要 delete 的 number)，去比較兩者大小，如果一樣則代表沒必要變動到排序(即插入的數等於刪掉的數)，之後經過判斷得出剩餘的數列是要往後擠還是往前推，然後再去送入每個值。

Median Number Caculator-Trick

使用一個 10bits 的記憶體 mv 去記錄每個 pixel 的 Q 值(超出影像的部分為 0 值)，然後 mask 右移的時候先去寫入新增的數到 mv_next 內，然後再把剩餘 mv 的數送到全部往左移一位送到 mv_next 內，即可完成 mv 內記錄每個 pixel 的更新。

Result Display

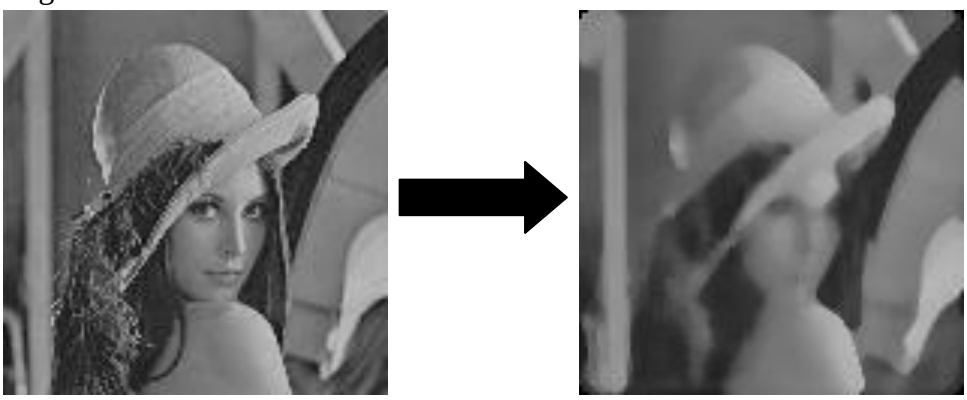
我使用 python 的 pillow 函式庫進行影像的輸出，程式碼如下：

```
from PIL import Image
import sys

if len(sys.argv) < 4 :
    print 'Format'
    # print 'python image.py <input file> <output file>'
    print 'python image.py <width size> <height size> <input file> ...'
    sys.exit()

width = int(sys.argv[1])
height = int(sys.argv[2])
for everyfile in range(3,len(sys.argv)):
    input_file_name = sys.argv[everyfile]

    with open(input_file_name) as open_file:
        image_content = [int(line.strip()[-3:]) for line in open_file.readlines()]

    list_index = 0
    img = Image.new('RGBA', (width,height))
    for y in range(height):
        for x in range(width):
            rgba = (image_content[list_index], image_content[list_index] ,image_content[list_index] ,1)
            img.putpixel((x,y), rgba)
            list_index += 1

    img.show()
    img.save('output_'+input_file_name+'.jpeg')
```

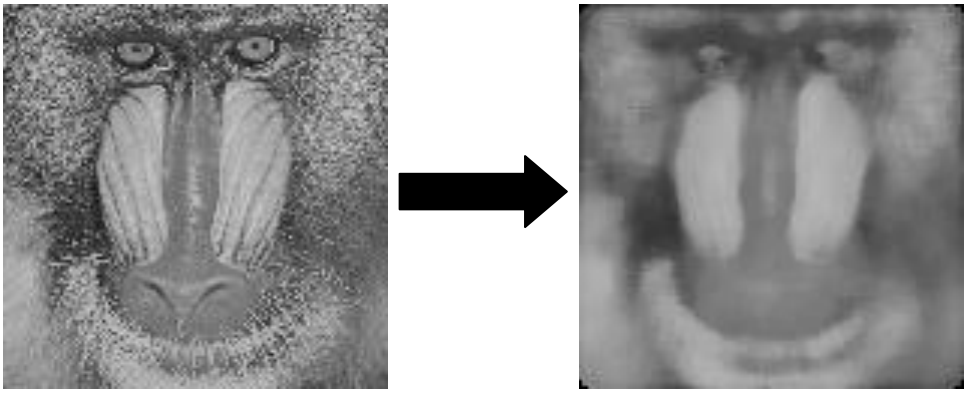
因為輸入要為 RGBA 值，因此我給定 A 值都為 1，然後因為是 pixel 為 Grayscale，因此把 pixel data 的值都都送入 R, G, B 中即可。

(左邊原始 pattern.dat 輸出的圖檔，右邊經過 7x7mask Median Filter 處理後 out.dat 輸出的圖檔)

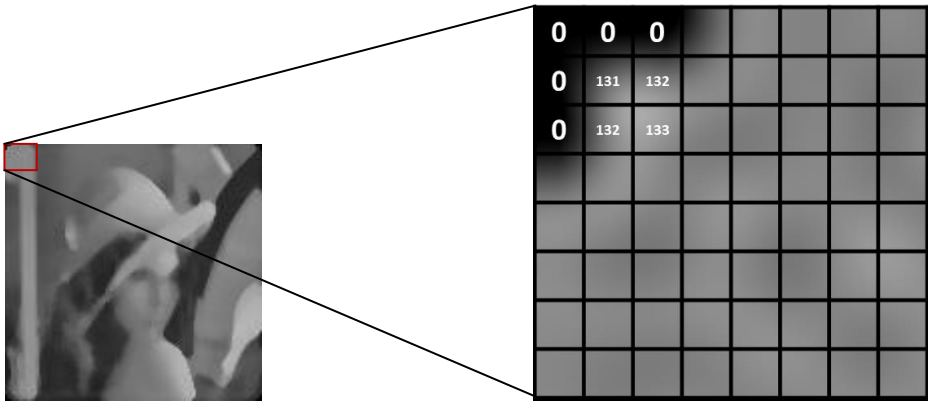
Pattern1: Lena image



Pattern2: Baboon image



細節部分如下：

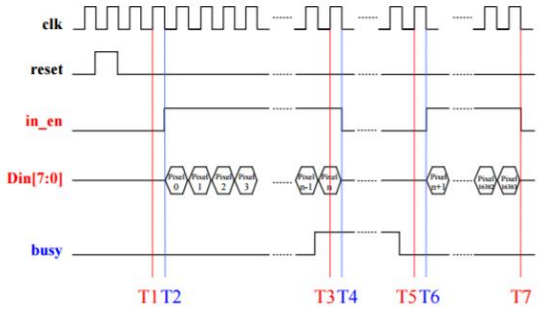


因為 Mask over-covered 的關係，因此邊邊角角都會是全黑(pixel data = 0)

Stimulation

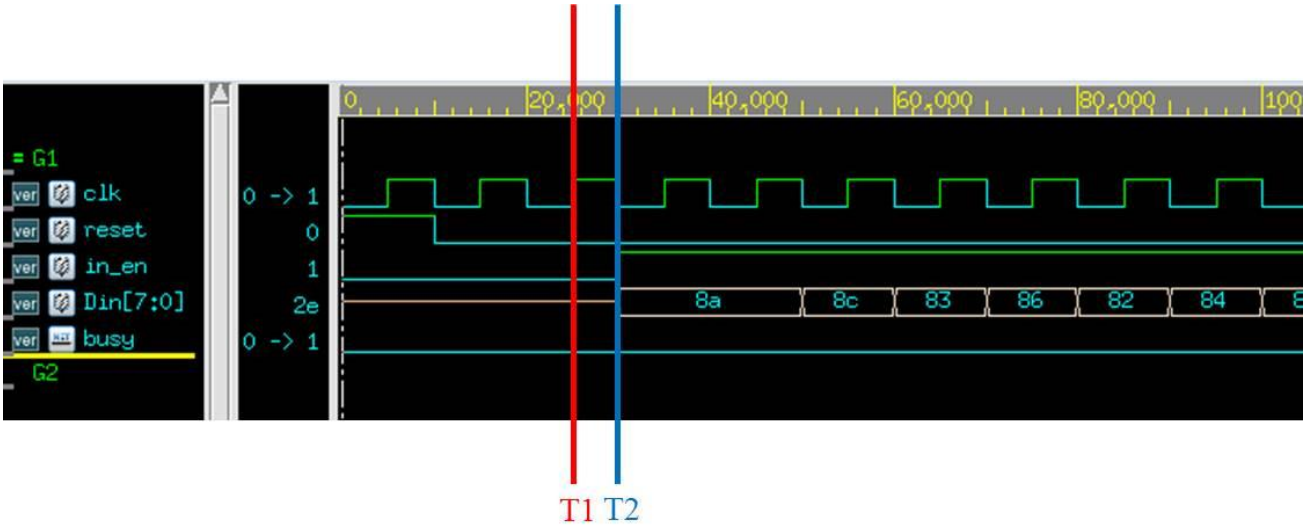
[iclab79@ic21]nWave

下圖為題目提供之 LMFE 電路輸入之時序圖：

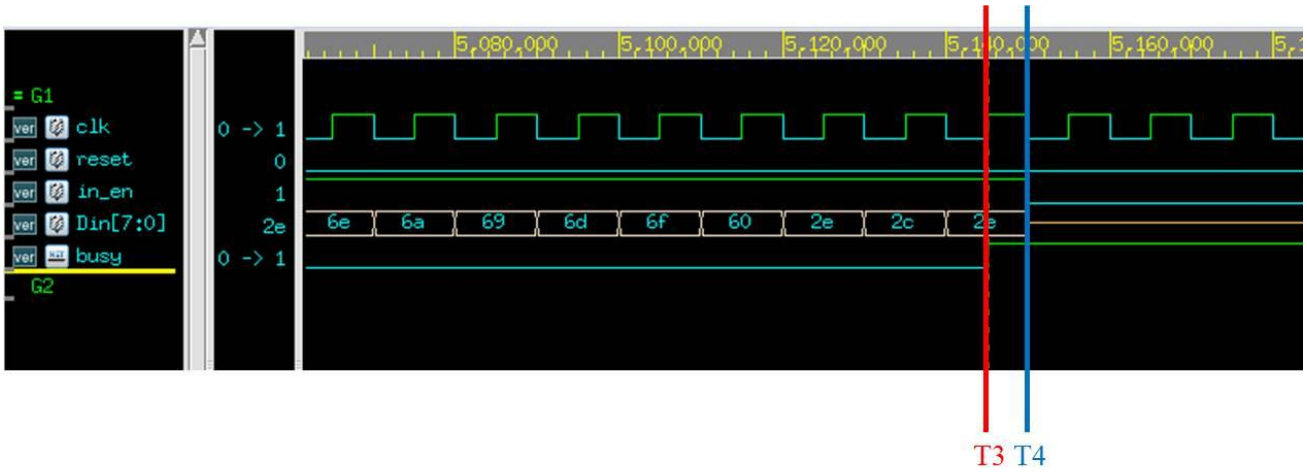


以下為使用 nWave 模擬後的實際電路時序圖：

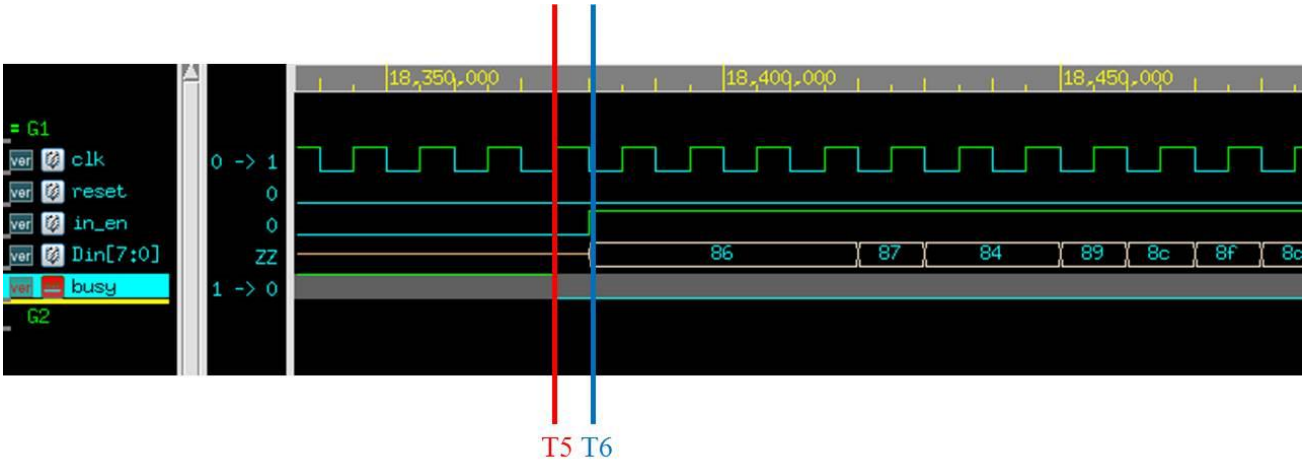
1. LMFE 電路初始化，Reset 一個 Cycle 的時間。
2. T1 時間點，Reset 後等待兩個 Cycle 的時間，Host 於 T1 時間點判斷 busy 訊號為 low，因此 T2 時間點，開始送出第一筆 Pixel 訊號。



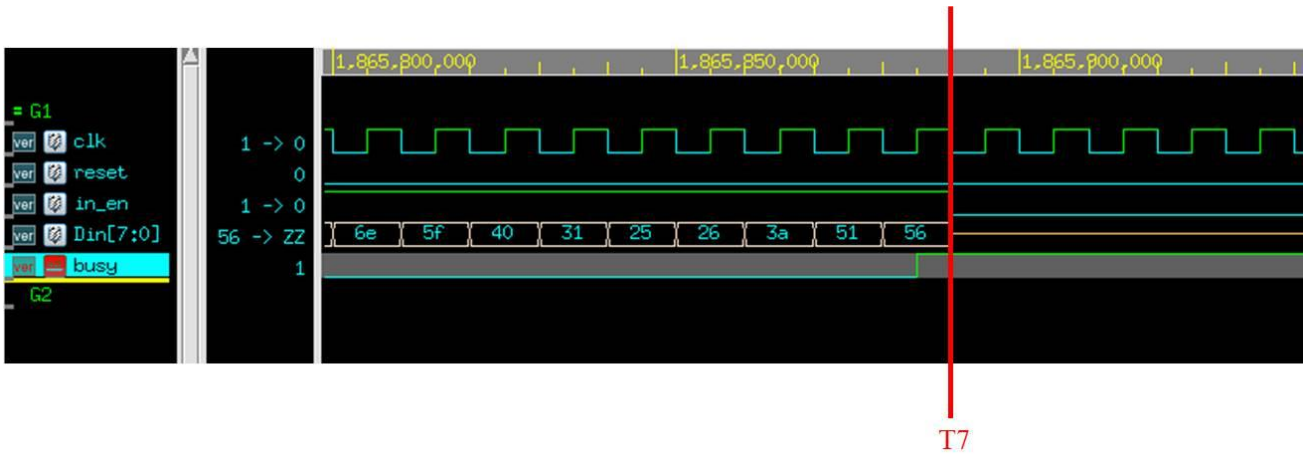
3. T3 時間點，欲暫停影像訊號輸入，將 busy 拉為 High，於 T3 時間點判斷後，T4 時間點便開此暫停影像訊號期間，沒有送訊號期間 in_en 維持為 Low，且 Din 會維持 High Impedance。



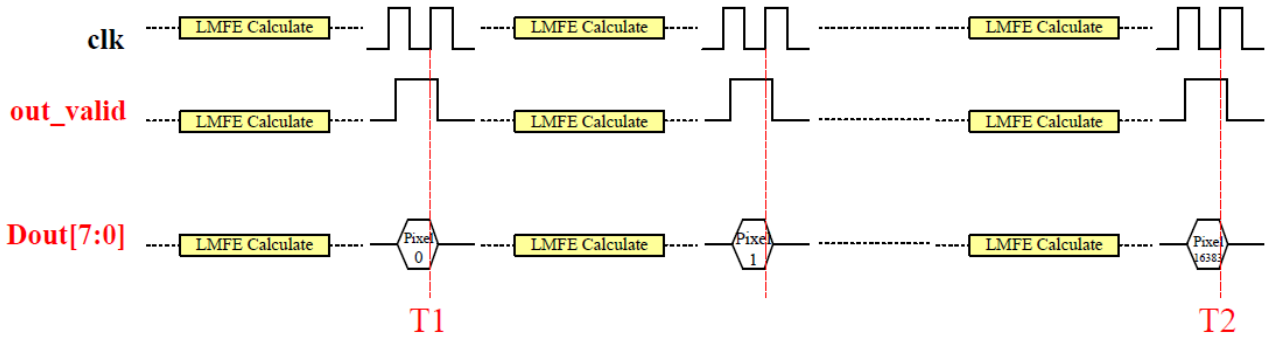
4. T5 時間點，判斷 busy 為 low，於 T6 時間點進緒輸入剩餘的影像訊號。



5. T7 時間點，完成輸入 16384 比影像訊號，T7 時間點以後，Din 會一直維持在 High impedance，in_en 維持在 Low。

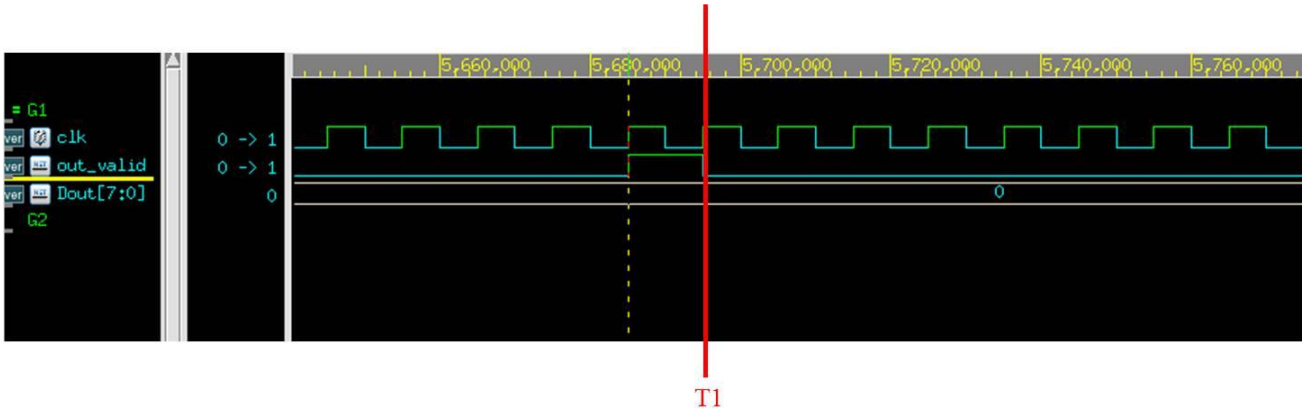


下圖為題目提供之 LMFE 電路輸出之時序圖：

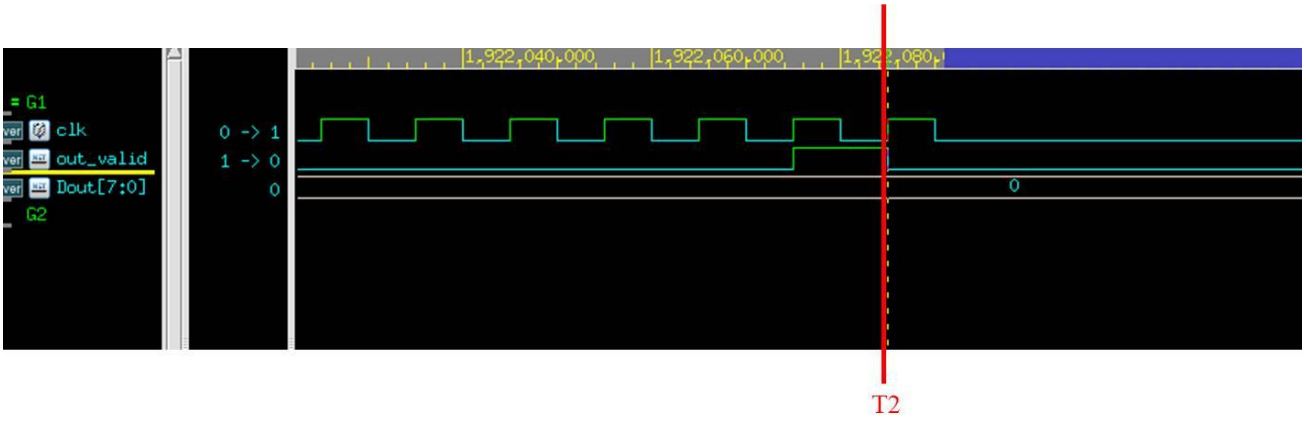


以下為使用 nWave 模擬後的實際電路時序圖：

1. T1 時間點，完成 Pixel0 之計算，輸出 out_valid 訊號為 High。



2. T2 時間點，完成最後一筆 Pixel16383 計算完畢後輸出，模擬立即結束。



Synthesization

[iclab79@ic21]design_vision

》File》Read

完後當前的 current design 位置會不是在頂層，要在 dc_shell 輸入 current_design LMFE

```
Presto compilation completed successfully.
Current design is now '/users/course/2016F/cs5121/iclab79/midterm_project/lmfe_med49.db:lmfe_med49'
Loaded 4 designs.
Current design is 'lmfe_med49'.
design_vision>
Current design is 'lmfe_med49'.
```

輸入完後便可以回到頂層！

```
Presto compilation completed successfully.
Current design is now '/users/course/2016F/cs5121/iclab79/midterm_project/lmfe_med49.db:lmfe_med49'
Loaded 4 designs.
Current design is 'lmfe_med49'.
design_vision>
Current design is 'lmfe_med49'.
design_vision> current_design LMFE
Current design is 'LMFE'.
{LMFE}
Current design is 'LMFE'.
design_vision>
```

》File》Analyze

》File》Elabroate

```
Warning: Design 'LMFE' was renamed to 'LMFE_1' to avoid
        a conflict with another design that has the same name but
different parameters. (LINK-17)
Elaborated 1 design.
Current design is now 'LMFE_1'.
design_vision>
Current design is 'LMFE_1'.
Loading db file '/usr/cad/synopsys/synthesis/cur/libraries/syn/generic.sdb'
```

》Attribute》Specify Clock // for sequential circuits

》Attribute》Optimization Constraints》Timing Constraints // for combinational circuits

從最頂層的 clk 加入就可以了！

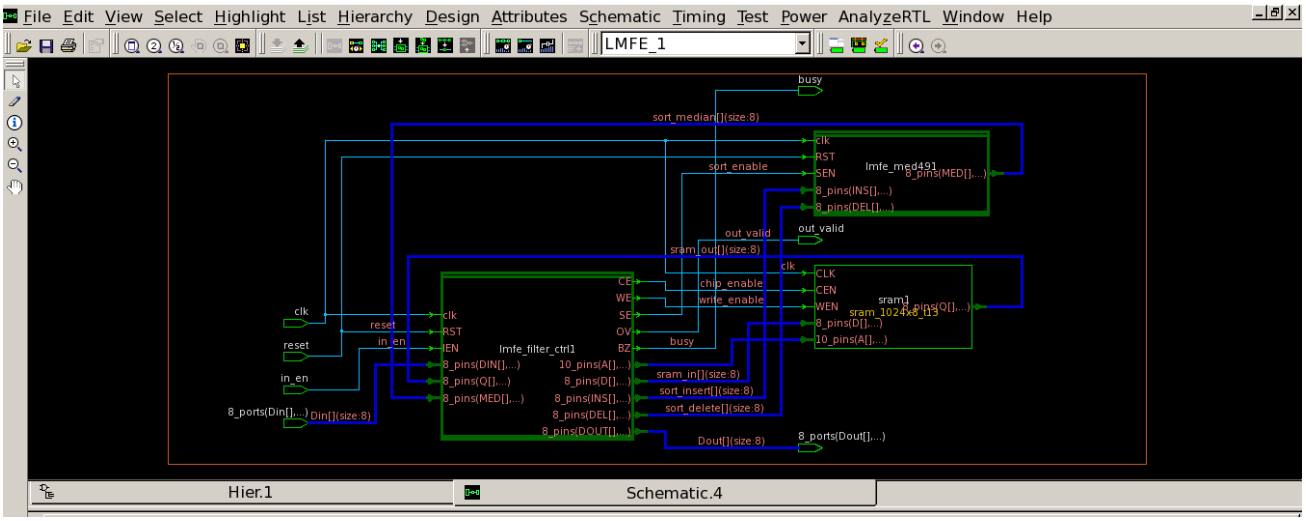
```
Compiling source file /users/course/2016F/cs5121/iclab79/midterm_project/filter_ctrl.v
Presto compilation completed successfully.
design_vision> elaborate LMFE -architecture verilog -library DEFAULT
Running PRESTO HDLC
Presto compilation completed successfully.
Warning: Design 'LMFE' was renamed to 'LMFE_1' to avoid
        a conflict with another design that has the same name but
different parameters. (LINK-17)
Elaborated 1 design.
Current design is now 'LMFE_1'.
design_vision>
Current design is 'LMFE_1'.
Loading db file '/usr/cad/synopsys/synthesis/cur/libraries/syn/generic.sdb'
design_vision> create_clock -name "CLK" -period 10 -waveform { 0 5  }  { clk  }
1
design_vision>
```

》Design》Check Design

》Design》Compile Design

```
  Optimization Complete
  ---------------------
1
Current design is 'LMFE_1'.
design_vision>
```

Schematic View



Report Area

```
Number of cells:                      11094
Number of combinational cells:        10136
Number of sequential cells:             900
Number of macros/black boxes:             1
Number of buf/inv:                     2167
Number of references:                     3

Combinational area:              77009.340039
Buf/Inv area:                     8863.822664
Noncombinational area:           28256.617176
Macro/Black Box area:            69557.296875
Net Interconnect area:      undefined  (No wire load specified)

Total cell area:                174823.254090
Total area:                 undefined
```

Report Power

```
                 Internal         Switching           Leakage            Total
Power Group      Power            Power               Power              Power   (   %    )  Attrs
--------------------------------------------------------------------------------------------------
io_pad             0.0000           0.0000              0.0000            0.0000  (   0.00%)
memory             0.6410        2.3699e-04          2.8000e+07            0.6692  (  26.22%)
black_box          0.0000           0.0000              0.0000            0.0000  (   0.00%)
clock_network      0.0000           0.0000              0.0000            0.0000  (   0.00%)
register           1.7684        2.3069e-03          2.6790e+07            1.7975  (  70.44%)
sequential         0.0000           0.0000              0.0000            0.0000  (   0.00%)
combinational  9.3724e-03        2.2499e-02          5.3364e+07         8.5235e-02  (   3.34%)
--------------------------------------------------------------------------------------------------
Total              2.4187 mW     2.5043e-02 mW     1.0815e+08 pW         2.5519 mW
design_vision>
```

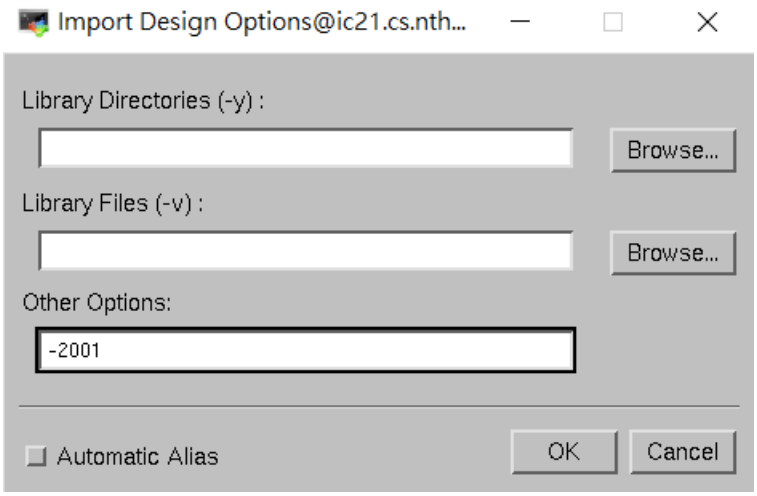
Report Timing Paths

```
  lmfe_filter_ctrl1/A_reg[9]/D (DFFRX1)                   0.00       5.64 f
  data arrival time                                                  5.64

  clock CLK (rise edge)                                  10.00      10.00
  clock network delay (ideal)                             0.00      10.00
  lmfe_filter_ctrl1/A_reg[9]/CK (DFFRX1)                  0.00      10.00 r
  library setup time                                     -0.23       9.77
  data required time                                                 9.77
  --------------------------------------------------------------------------
  data required time                                                 9.77
  data arrival time                                                 -5.64
  --------------------------------------------------------------------------
  slack (MET)                                                       4.13


design_vision>
```

HDL Design Rule Checker

[iclab79@ic21]nLint -gui

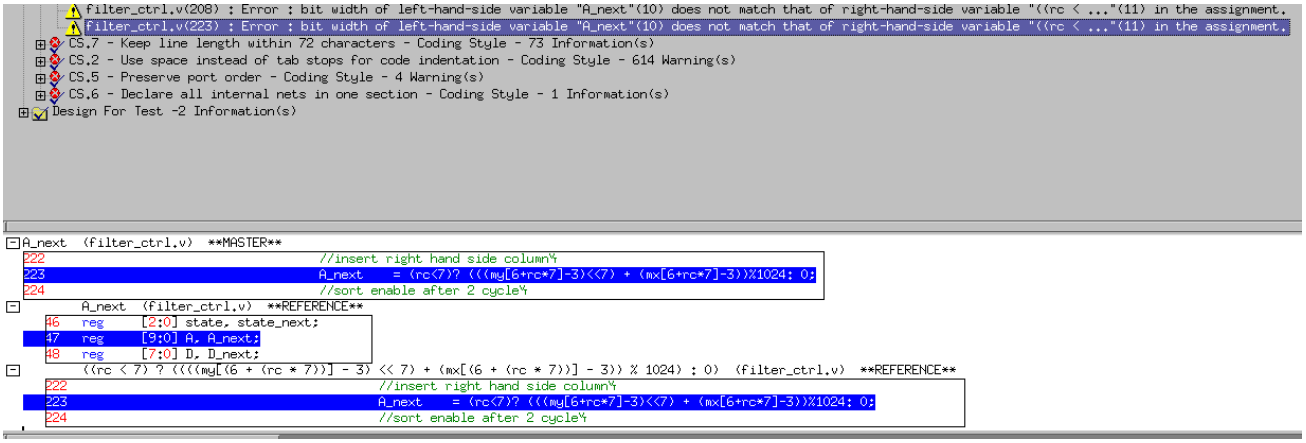
支援 Verilog 2001 因為允許更多新加入的語法，因此 Error 會少掉很多!



大概修正錯誤的地方在：

1. I/O ports 後面要加 comments
2. FSM 裡面給值要對應到宣告 bits 數(舉例來說我宣告 reg [9:0] A，但在初始化 A 的時候卻是給 1'b0)

最後無法修正的錯誤：



解釋理由：

因為我宣告存放 address 的 register 為 10bits，利用 pixel number 本身超出 10bits(即大於 1024)後 overflow 的概念去覆蓋用不到的記憶體位置，因此等號兩邊的值本來就不會相同大小，右邊的數字勢必會超過 1024！

Post-Synthesize Stimulation

RTL 模擬時的 Unix 指令

[iclab79@ic21]ncverilog +nospecify +notimingchecks testfixture1.v LMFE.v filter_ctrl.v lmfe_med49.v sram_1024x8_t13.v +access+r

Gate-Level Post-layout 模擬時的 Unix 指令

[iclab79@ic21]ncverilog testfixture1.v LMFE_syn.v -v /theda21_2/CBDK_IC_\Contest/cur/Verilog/tsmc13.v +define+SDF +access+r

執行完結果如下:

```
--------------------------------------------------

Congratulations! All data have been generated successfully!

-------------------------PASS-------------------------

Simulation complete via $finish(1) at time 1920801 NS + 0
./testfixture1.v:135        #(`CYCLE/2); $finish;
ncsim> exit
[iclab79@ic21 ~/midterm_project]$
```

Project Review

以前的合成與 APR 都是只有用 vivado 去做，而不是使用 Design Vision 去操作，這次的 project 讓我學到了 Vavido 只要按個按鍵的東西在實際上用 EDA tool 去做大概是怎樣的步驟！而還有使用其他 EDA tool 去跑驗證、跑報告，也算是學習了很多！特別感謝吳岳騏助教讓我去實驗室問問題(不過我猜想可能是臉書被打擾太多次了 XD)。

在 coding style 的部分上網查了很多資料，自己原本寫了一個 Bubble Sort 想說應該有辦法合成，只是兩個 for loop 的複雜度 O(N)讓合成遲遲無期，好像有查到硬體是用去展開 for loop 的方式來進行合成，因此這樣會合成合到死(抱歉我原本想用委婉點的說法但找不到更適合的了！)，最後在網路上找到的 median caculator 實際去跑過模擬參透他的用法後就拿來用了！這次讓我更學到關於 FSM 的使用法，FSM 使用續向電路去執行每次值的更新，而更新後的值再去跑組合電路運算出下次要送入的值，這樣就是偏硬體的 coding style，不然以前接觸到的都是 C++居多，總會用很軟體的方式去寫，然後就不能合成，大概是這次最大的收穫吧！

最後用一張我寫半天但卻無法順利合成的 Bubble Sort 畫下句點，旁邊是可以合成的 Median Caculator！

```
always @ * begin
    if(state == SORT) begin
        for(m = 49; m > 0; m = m - 1) begin
            for(n = 0; n < m; n = n + 1) begin
                if(buffer[n] < buffer[n + 1]) begin
                    temp = buffer[n];
                    buffer[n] = buffer[n + 1];
                    buffer[n + 1] = temp;
                end
            end
        end
    end
end
```