


2023 Digital IC Design Homework 4

NAME	林宜謙		
Student ID	N16100250		
Simulation Result			
Functional simulation	100	Gate-level simulation	100
<pre>#----- # START!!! Simulation Start #----- # Layer 0 output is correct ! # Layer 1 output is correct! #----- #----- S U M M A R Y ----- # Congratulations! Layer 0 data have been generated successfully! The result is PASS!! # Congratulations! Layer 1 data have been generated successfully! The result is PASS!! # terminate at 46087 cycle #----- # ** Note: \$finish : C:/Users/lin/Documents/modelsim/verilog_practice/HW4/testfixtu</pre>		<pre>#----- # START!!! Simulation Start #----- # Layer 0 output is correct ! # Layer 1 output is correct! #----- #----- S U M M A R Y ----- # Congratulations! Layer 0 data have been generated successfully! The result is PASS!! # Congratulations! Layer 1 data have been generated successfully! The result is PASS!! # Congratulations! Layer 1 data have been generated successfully! The result is PASS!! # terminate at 46087 cycle #----- # ** Note: \$finish : C:/Users/lin/Documents/modelsim/verilog_practice/HW4/testfixtu</pre>	
Synthesis Result			
Total logic elements	232		
Total memory bits	0		
Embedded multiplier 9-bit elements	0		
Total cycle used	46087		
Flow Summary			
 <<Filter>>			
Flow Status	Successful - Thu May 18 17:27:41 2023		
Quartus Prime Version	20.1.1 Build 720 11/11/2020 SJ Lite Edition		
Revision Name	ATCONV		
Top-level Entity Name	ATCONV		
Family	Cyclone IV E		
Device	EP4CE55F23A7		
Timing Models	Final		
Total logic elements	232 / 55,856 (< 1 %)		
Total registers	99		
Total pins	82 / 325 (25 %)		
Total virtual pins	0		
Total memory bits	0 / 2,396,160 (0 %)		
Embedded Multiplier 9-bit elements	0 / 308 (0 %)		
Total PLLs	0 / 4 (0 %)		

Description of your design

程式流程說明：

設計的思路主要是將每一個畫素(pixel)對應 9 次的 kernel filter 的計算，而每讀取一次畫素值就更新一次圖片的位址，而為減少 cycle 數給入新的位址並同時計算由舊的位址得出的值進行卷積計算，每 9 次計算後就寫回到 Layer0 的記憶體中，並比較計算的值取較大的值，在每做完 4 次的 layer0 寫回後，將陣列中最大值取整數並寫回 layer1 的記憶體中，每 4 個原圖的畫素做一次這樣的流程直到所有畫素完成計算，而根據設計的流程可總共分成 6 個狀態，分別為：

CHECK_IMG_RD：確認圖片是否完成載入到 tb 圖片的記憶體中，有的話將 busy 訊號線拉高，開始之後的運算。

GET_DATA_FROM_MEM：將中間的的畫素值位址傳送給 tb。

CONVOLUTION：每一個畫素值由上一個狀態輸入欲讀取的位址後在這一個狀態可以得到中間的畫素值，根據 filter 的編號進行 shift 計算，並將每一次的計算值加總，並且更新 filter 的編號更新下一個要讀取的畫素值。

WRITE_RELU_LAYER0：在前一狀態後會得到當前畫素值的卷積值，並加上 bias，將寫回的訊號拉高並寫回 layer0 中，而 Relu 的判斷方式則是判斷 sign bit 將正數保留，反之為 0，並且將值暫存與卷積總和歸回 bias。

WRITE_LAYER1：當寫入 layer0 4 次後，將最大值取整寫回 layer1 的記憶體中，而若完成兩列的畫素值計算，更新中間的畫素值完成一次 4 個原圖片畫素值的計算。

RESULT：若做完所有的計算將 busy 訊號拉低，使 tb 確認。

變數說明：

sum_conv：卷積計算的總和值

image_mem_idx：輸入至 iaddr 每一次需要讀位址的變數，透過組合電路更新值

layer1_mem_idx：寫回 layer1 位址的變數，每做完一次 maxpooling 就更新+1 一次

current_pixel：儲存在 layer0 做卷積計算中間的位址，kernel filter 的 padding 位址方式會以此變數進行計算

counter_for_9：作為 kernel filter 計算的順序的計數值

counter_for_4：作為 4 次寫入 layer0 計數值

filter_shift：對應 kernel filter 需要做 shift 的次數

next_mem_offset：做完 maxpooling 後的下一個位址的移動值

bias：卷積計算的 bias 值

max_data：卷積值最大值變數

padding 處理：

非最外兩圈的不會有 padding 的問題，在最外兩圈會有對應的計算方式，以獲取需要 padding 的記憶體位址，kernel filter padding 問題的組合電路計算方式：

首先，kernel filter 計算順序由中間的值 0 索引開始至到 8，示意順序為：

[1] [7] [2]

[6] [0] [5] 雖然比較不直觀的順序，但合成的時候將 4 個角落與 4 個邊計算

[3] [8] [4] 方式排在一起的時候會減少成本

在處理位址計算時牽涉到"行"(column)的計算的時候，考慮位址的後 6-bit，反之，牽涉到"列"(row)的計算的時候，考慮位址的前 6-bit。再來組合電路以 kernel filter 的索引值分成不同的 case 計算，先看到 5678 的索引，代表的是 4 個邊，分別探討：

索引 5 代表右邊那排，以中間的值的位址要+2，但是最右邊那排的數字+0，而倒數右邊一排的數字+1，為"行"的運算，所以在後 6-bit 要創造出分別三種加法的情況，用 2 個 bit 代表，第一個 bit 的條件為小於 62，第二個 bit 的條件為等於 62，所以最右邊那排的數字根據後 6-bit 會得到 2'b00，倒數右邊一排會得到 2'b01，非這兩排，也就是不用 padding 的"行"會得到 2'b10

索引 6 代表左邊那排，以中間的位址要-2，所以同理索引 5 的思路，也是數字的後 6-bit 去判斷出 2, 1, 0 的數字，同樣的第一個 bit 條件變為大於 1，第二個 bit 條件變為等於 1，就可以得到欲求出數字

索引 7 代表上排那排，以中間的值的位址要+128，但第一列的數字+0，而第二列的數字+64，而 0/64/128 的數字分別代表前 6 個 bit 0/1/2，所以邏輯上與索引 6 的方式一樣，只是計算於前 6-bit，後 6-bit 不變，因此索引 8 也就是最下面兩排的數字也是一樣類似索引 5 的處理方式，以此類推。

至於索引 1234，也就是代表著 4 個角落的 4 個數字的處理方式，

索引 1 的數字則為索引 7 與索引 6 的交集，分別處理前 6-bit 與後 6-bit，同樣的以此類推，索引 2 的數字則為索引 7 與索引 5 的交集、索引 3 的數字則為索引 6 與索引 8 的交集、索引 4 的數字則為索引 5 與索引 8 的交集

各狀態(state)說明：

CHECK_IMG_RD: 將 busy 訊號拉高與 ready 訊號一樣

GET_DATA_FROM_MEM: 此狀態要輸入中間的畫素值位址，作為第一次的卷積資料位址輸入，在下一個狀態才會得到資料，將計數器設為 1 為了觸發上述的組合電路更新第二個要讀取的位址

CONVOLUTION: 在進入這個狀態的時候，在前一個狀態可以得到前一個位址的 idata 值，而同時重新輸入至 iaddr 的時候已經是被更新過的”新的”位址了，將得到的 idata 資料進行 shift，也就是乘法的計算，而觀察欲相乘的數字就會發現，利用右 shift 就可以得到計算的結果，但因為是負數(正數*負數)，所以在計算的時候可以轉換成 shift 後再做二補數的計算，中間的值則為原值不需要變動，做完一次的計算後再透過更新計數的變數從而更新下一個位址

WRITE_RELU_LAYER0: 在上一個狀態做完 9 次的計算後，將結果寫回 layer0 與並判斷大小確認是否要更新最大值，而在這之前要先對計算好的結果做 relu 的計算，也就是取正數，所以只要判斷第一個 signed bit 就可以分辨出來，再來是下一個中心畫素位址的更新，以第一組為例，分別要計算 0/1/64/65 的卷積值，所以移動值就會是+1/+63/+1，而要移到下一組的第一個值時，再移動-63

WRITE_RELU_LAYER0: 當做完 4 次卷積計算後就會進行一組的 maxpooling 與取整，並將最大值取整並寫回 layer1，取整的計算方式就是先把後 4-bit 右移 4 位直接捨去，若後 4-bit 不為 0，也就是有任何值的話就會+1，最後再左移 4 位完成計算，而中心值(current_pixel)再做完 32 次後(0~31)就要加一次 128，為下兩列的計算移位

RESULT: 當圖片都完成卷積後，將 busy 訊號拉低，使 tb 確認

*Scoring = (Total logic elements + Total memory bits + 9*Embedded multipliers 9-bit elements) X Total cycle used*

*** Total logic elements must not exceed 1000.**

$$Scoring = 10692184 = (232 + 0 + 9 * 0) * 46087$$