2023 Digital IC Design Homework 4

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| NAME | 林宜謙 | | | | |
| Student ID | N16100250 | | | | |
| **Simulation Result** | | | | | |
| Functional simulation | | 100 | | Gate-level simulation | 100 |
|  | | | |  | |
| **Synthesis Result** | | | | | |
| Total logic elements | | | 232 | | |
| Total memory bits | | | 0 | | |
| Embedded multiplier 9-bit elements | | | 0 | | |
| Total cycle used | | | 46087 | | |
|  | | | | | |
| **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layer04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**