2023 Digital IC Design Homework 4

NAME	葉育賢
Student ID	N96111516

Simulation Result

Functional simulation	Score	Gate-level simulation	Score

```
VSIM 2> run -all
#
# ------------------------------------------------
#
# 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      56326 cycle
# ------------------------------------------------
#
# ** Note: $finish    : C:/Users/dear2/Desktop/IC_hw04/file/testfixture.v(178)
#    Time: 2816300 ns  Iteration: 0  Instance: /testfixture
# 1
# Break in Module testfixture at C:/Users/dear2/Desktop/IC_hw04/file/testfixture.v line 178
```

```
VSIM 5> run -all
#
# ------------------------------------------------
#
# 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      56326 cycle
# ------------------------------------------------
#
# ** Note: $finish    : C:/Users/dear2/Desktop/IC_hw04/file/testfixture.v(178)
#    Time: 2816308127 ps  Iteration: 0  Instance: /testfixture
# 1
# Break in Module testfixture at C:/Users/dear2/Desktop/IC_hw04/file/testfixture.v line 178
```

Synthesis Result

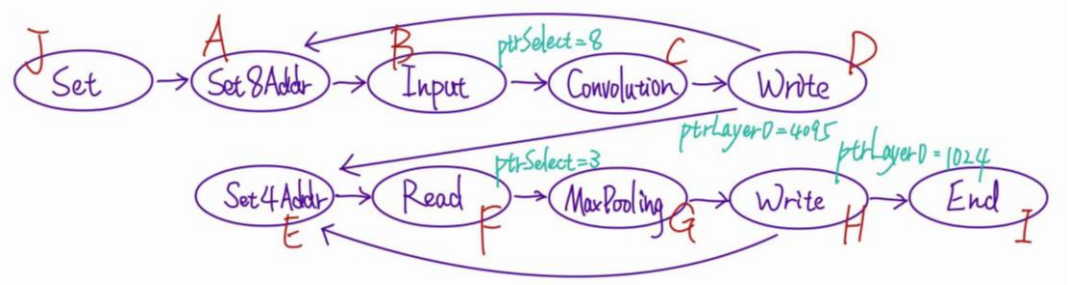
Total logic elements	860
Total memory bits	0
Embedded multiplier 9-bit elements	0
Total cycle used	56326

Flow Summary

<<Filter>>

Flow Status	Successful - Tue May 16 15:03:33 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	860 / 55,856 (2 %)
Total registers	327
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



首先是狀態機的設計，我總共用了 10 個狀態，分別是：

A. 依照當前的 pixel 座標，設定周圍八個 pixel 的序號，並存入 nextAddr 中。

B. 依照前一步存取在 nextAddr 中的序號，使用 iaddr 和 idata 取得資料存入 select。

C. 對 select 中的資料做一次 convolotion，將結果存入 tmpResult。

D. 將 tmpResult 考慮正負後以 caddr_wr 和 cdata_wr 輸出值或 0。

E. 同 A，但只選取 pooling 需要的四個序號。

F. 同 B，但只取四個資料。

G. 將 select 中的 4 個資料做 max-pooling。

H. 將結果無條件進位後輸出。

I. 結束訊號，Layer0 和 Layer1 結束後，將 busy 訊號關閉，驗證結果。

J. 初始狀態，避免再 reset 訊號進來前就影響到 busy 訊號導致 testbench 直接結束，故多設置這個狀態。

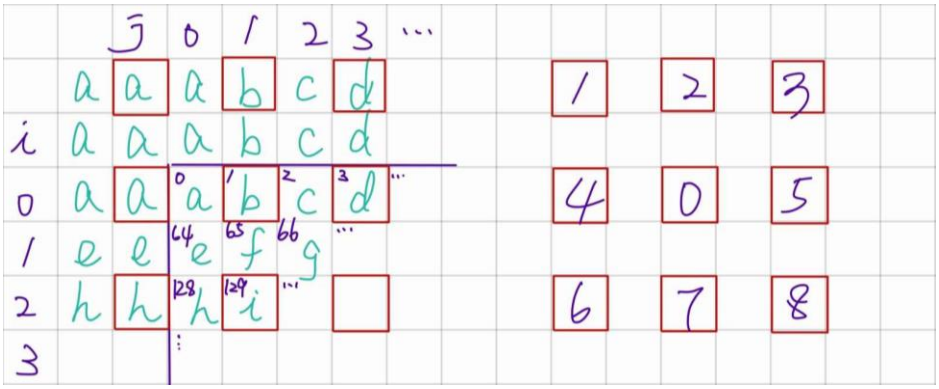
接下來詳細描述程式的內容。

A. 在這個狀態中，我先將 cwr 設為零，避免錯誤的輸出，並在這裡將 busy 訊號打開。開始時的 ptrNextAddr 為 0，我藉此取得中心點的序號。還有將 nextAddr[0]設為下一個中心點的序號，故加一。並計算下一個中心點的座標(i,j)。

```
102             A:begin
103                 cwr <= 1'b0;
104                 busy <= 1'b1;
105                 iaddr <= nextAddr[ptrNextAddr];
106                 ptrNextAddr <= ptrNextAddr + 1'b1;
107                 nextAddr[0] <= nextAddr[0]+1'b1;
108                 //左上角
109                 if(i<12'd2 & j<12'd2)begin
110                     nextAddr[1] <= 12'd0;
111                     nextAddr[2] <= j;
112                     nextAddr[3] <= j+12'd2;
113                     nextAddr[4] <= i*12'd64;
114                     nextAddr[5] <= nextAddr[0]+12'd2;
115                     nextAddr[6] <= 12'd64*i+12'd128;
116                     nextAddr[7] <= nextAddr[0]+12'd128;
117                     nextAddr[8] <= nextAddr[0]+12'd130;
118                 end
```

```
208                 if(j==12'd63)begin
209                     if(i==12'd63)begin
210                         i <= 12'd0;
211                         j <= 12'd0;
212                     end
213                     else begin
214                         i <= i + 1'b1;
215                         j <= 12'd0;
216                     end
217                 end
218                 else j <= j + 1'b1;
219             end
```

最後還要在這個狀態中得到周圍八個 pixel 的序號，我考慮 9 種分類，這裡以圖片的左上角為例。



左邊的圖為左上角 padding 後的資料值，但為了避免使用太多的硬體資源，沒辦法真的做出 padding，但可以依照 padding 的性質去設定周圍八個 pixel 的序號。

以左上角 abcd 四個值而言，設定他們的座標為(i,j)，他們左上方位置[1]的值必然為 a，其座標為(0,0)，代入座標轉換成序號的公式$64\ i+j$ 則序號為 0。同理，正上方[2]的座標應為(0,j)，序號為 j。正右方[5]的座標應為(i,j+2)，序號為 64i+j+2，其中 64i+j 為中心點序號，故可以表示為 nextAddr[0]+2。以此類推，則可以列出九種況各自的 nextAddr。

B. 在狀態 B 中依照 nextAddr 取得需要的九個資料，並存入 select 中。當九個值都輸入完成時將 ptrNextAddr 歸零。

```
220             B:begin
221                 iaddr <= nextAddr[ptrNextAddr];
222                 select[ptrSelect] <= idata;
223                 ptrSelect <= ptrSelect + 1'b1;
224                 if(ptrNextAddr == 4'd8)ptrNextAddr <= 13'd0;
225                 else ptrNextAddr <= ptrNextAddr + 1'b1;
226             end
```

C. 在這個狀態中計算 convolution，並將結果存入 tmpResult 中。因為卷積的計算較大，所以我將它拆成整數和小數，最後再相減取得正確的結果。

```
227             C:begin
228                 tmpResult <= (select[0]
229                                 -{4'b0000,select[1][12:8],4'b0000}
230                                 -{3'b000 ,select[2][12:7],4'b0000}
231                                 -{4'b0000,select[3][12:8],4'b0000}
232                                 -{2'b00  ,select[4][12:6],4'b0000}
233                                 -{2'b00  ,select[5][12:6],4'b0000}
234                                 -{4'b0000,select[6][12:8],4'b0000}
235                                 -{3'b000 ,select[7][12:7],4'b0000}
236                                 -{4'b0000,select[8][12:8],4'b0000})
237                                 -
238                                 ({9'd0,select[1][7:4]}
239                                 +{9'd0,select[2][6:4],1'd0}
240                                 +{9'd0,select[3][7:4]}
241                                 +{9'd0,select[4][5:4],2'd0}
242                                 +{9'd0,select[5][5:4],2'd0}
243                                 +{9'd0,select[6][7:4]}
244                                 +{9'd0,select[7][6:4],1'd0}
245                                 +{9'd0,select[8][7:4]})
246                                 -13'd12;
247             end
```

D. 接下來在狀態 D 中考慮 tmpResult 的正負，輸出 tmpResult 或 0。並將 cwr 打開，以輸出結果。還有將 ptrNextAddr 和 ptrSelect 歸零。若 Layer0 處理完成，則先為狀態 E 設定 i,j 的值。

```
248             D:begin
249                 cwr <= 1'b1;
250                 caddr_wr <= ptrLayer0;
251                 ptrLayer0 <= ptrLayer0 + 1'b1;
252                 ptrNextAddr <= 13'd0;
253                 ptrSelect <= 13'd0;
254                 if(tmpResult[12] == 1'd1) cdata_wr <= 13'd0;
255                 else cdata_wr <= tmpResult;
256 
257                 if(next_state == E)begin
258                     i <= 13'd0;
259                     j <= 13'd1;
260                     ptrLayer0 <= 13'd0;
261                 end
262                 else begin
263                     i <= i;
264                     j <= j;
265                 end
266             end
```

E. 狀態 E 和狀態 A 相同，都是先設定好需要的序號。

```
267             E:begin
268                 csel <= 1'b0;
269                 cwr <= 1'b0;
270                 crd <= 1'b1;
271                 caddr_rd <= nextAddr[ptrNextAddr];
272                 ptrNextAddr <= ptrNextAddr + 1'd1;
273                 nextAddr[0] <= i*13'd128+j*13'd2;
274                 nextAddr[1] <= nextAddr[0] + 1'd1;
275                 nextAddr[2] <= nextAddr[0] + 13'd64;
276                 nextAddr[3] <= nextAddr[0] + 13'd65;
277 
278                 if(j==12'd31)begin
279                     if(i==12'd31)begin
280                         i <= 12'd0;
281                         j <= 12'd0;
282                     end
283                     else begin
284                         i <= i + 1'b1;
285                         j <= 12'd0;
286                     end
287                 end
288                 else j <= j + 1'd1;
289             end
```

F. 類同狀態 B，從 Layer0 中讀取需要的資料。

```
290             F:begin
291                 caddr_rd <= nextAddr[ptrNextAddr];
292                 select[ptrSelect] <= cdata_rd;
293                 ptrSelect <= ptrSelect + 1'b1;
294                 if(ptrNextAddr == 4'd3)ptrNextAddr <= 13'd0;
295                 else ptrNextAddr <= ptrNextAddr + 1'b1;
296             end
```

G. 同狀態 C，只是這裡先進行一次取極值，下個狀態再取出最大值。

```
297             G:begin
298                 csel <= 1'b1;
299                 crd <= 1'b0;
300                 if(select[0] >= select[1])select[0] <= select[0];
301                 else select[0] <= select[1];
302 
303                 if(select[2] >= select[3])select[2] <= select[2];
304                 else select[2] <= select[3];
305             end
```

H. 將結果無條件進位後輸出。

```
306             H:begin
307                 cwr <= 1'b1;
308                 caddr_wr <= ptrLayer0;
309                 ptrLayer0 <= ptrLayer0 + 1'b1;
310                 ptrNextAddr <= 13'd0;
311                 ptrSelect <= 13'd0;
312                 if(select[0] >= select[2])begin
313                     if(select[0][3:0] == 4'b0)cdata_wr <= select[0];
314                     else cdata_wr <= {select[0][12:4] + 1'b1 , 4'b0};
315                 end
316                 else begin
317                     if(select[2][3:0] == 4'b0)cdata_wr <= select[2];
318                     else cdata_wr <= {select[2][12:4] + 1'b1 , 4'b0};
319                 end
320             end
```

I. 完成，將 busy 關閉。

```
321             default:begin
322                 busy <= 1'b0;
323             end
```

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

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